UNIVERSITY OF SORBONNE PARIS-NORD

Mixed-Integer Linear Programming: Polyhedral Aspects and Interplay between Lagrangian Relaxation and Machine Learning

Habilitation à Diriger des Recherches

Defended by

MATHIEU LACROIX

Reviewers:

- Mourad Baïou, Directeur de Recherches au CNRS, Université de Clermont-Ferrand,
- Marco Lübbecke, Professor of Operations Research, RWTH Aachen University,
- Miguel Anjos, Professor and Chair of Operational Research, University of Edinburgh.

Contents

Introduction

This document presents some of my research works that I have been conducting since my Ph.D. My dissertation was mostly focused on polyhedra stemming from combinatorial optimization, and I have never stopped studying such beautiful objects. The first part of this document is then naturally dedicated to this topic. Some years ago, I started considering problems related to natural language processing. Such problems are usually solved using machine learning together with combinatorial optimization. I address these problems from a mathematical programming's point of view: one the one hand, by using integer linear programming to model them and on the other hand by using Lagrangian relaxation based heuristics to solve the associated inference problems. Starting from these projects, I have been working on the interactions between machine learning and mixed-integer linear programming. The second part of this document is then dedicated to my contributions on these interactions.

These two parts can be read independently. Both start with preliminary definitions and notions that are necessary to read the corresponding chapters. These chapters give a concise description of my works without proofs nor experimental results. The interested reader should refer to the corresponding papers, which can be found in the Appendix, for more details.

Polyhedra and box-TDIness

A polyhedron is the intersection of a finite number of half-spaces. As half-spaces are defined by linear inequalities, the solutions to a linear problem form a polyhedron. For integer linear problems, one has to consider the set of integer points lying in the polyhedron defined by the constraints. However, this solution set may be described as the set of integer points of different polyhedra. Among them, a fundamental one is the minimum inclusionwise polyhedron, corresponding to the convex hull of the integer solutions. This polyhedron is crucial because optimizing a linear function over this convex hull with the simplex method returns an integer optimal solution. In other words, if the linear inequalities describing the convex hull are known, the integer linear problem may be cast as a linear one that corresponds to optimizing the original objective function over the convex hull. Otherwise, even a partial description may be used to add strengthening inequalities in order to reduce the solution space and speed up the resolution of the integer linear problem. This leads to cutting-plane based methods, which are a core component of the performance of mixed-integer linear solvers.

In Chapter 2, we give the linear description of the convex hull of bonds and st-bonds in series-parallel graphs, where a bond is a minimal inclusionwise cut. We also provide the linear description of the lexicographical polyhedron that corresponds to the convex hull of the points of a hypercube that are lexicographically between two given points. We say that a point p is lexicographically less than or equal to a point q if $p = q$ or the first different coordinate of p is less than the one of q.

Box-total dual integrality of a polyhedron is a strong geometric integrality property. Indeed, a polyhedron P is *box-Total Dual Integral* (box-TDI) if for every dilation of P corresponding to an integer polyhedron, its intersection with any integer hypercube remains an integer polyhedron. Box-TDI polyhedra usually provide min/max relations like the famous max-flow/min-cut theorem (Ford and Fulkerson, 1956).

In Chapter 3, we first answer two complexity questions related to box-TDI polyhedra. First, we prove that, even if box-TDI polyhedra have strong integrality properties, solving an integer linear problem when the linear inequalities define a box-TDI polyhedron is NP-hard. Moreover, we show that determining whether a rational linear system defines a box-TDI polyhedron is co-NP-complete, answering a 40 years old question asked in (Schrijver, 1986). We end this chapter by characterizing the box-TDIness of several polyhedra.

Interaction between Machine Learning and Lagrangian Relaxation

The second part of this document is related to my works lying at the intersection of machine learning and mathematical programming, especially dealing with Lagrangian relaxation.

In structured learning, the machine learning model has to predict a solution following some combinatorial structure like predicting a tree or a path of an input graph. This may be done in two steps: first predicting scores using machine learning and then determining the combinatorial solution having the best score. This second step is known as the inference problem and is nothing but a combinatorial optimization one.

In Chapter 5, we consider two inference problems arising in syntactic parsing, a fundamental natural language processing task aiming at determining the syntactic structure of a sentence. We tackle these two problems using integer linear programming and Lagrangian relaxation. For the first problem, which consists in computing a dependency tree with additional requirements, we formulate the set of solutions as the integer points of a polyhedron defined by an exponential number of inequalities. We also consider a Lagrangian relaxation of the problem obtained by dualizing an exponential number of inequalities. The resulting Lagrangian relaxed problem is then nothing but computing a spanning arborescence. We construct a heuristic based on this Lagrangian relaxation that is competitive with the state-of-the-art while ensuring that the returned solution satisfies all the additional requirements. The second problem consists in constructing a parse tree based on spinal TAG formalism. We reformulate this problem as a generalized spanning arborescence problem and we build a heuristic based on a Lagrangian decomposition of our formulation. Our heuristic compares favorably with the state-of-the-art both in terms of accuracy and of running time.

Mathematical programming solvers, even exact ones, rely on quite a lot of heuristic decisions that may dramatically affect their performance. These heuristics are usually handcrafted but recently, machine learning has been used to design or parametrize them.

In Chapter 6, we use machine learning to fasten the resolution of the Lagrangian dual problem. More specifically, we design a model to predict near-optimal Lagrangian multipliers. Our model can be used as-is to heuristically solve the Lagrangian dual problem in replacement to traditional iterative sub-gradient based algorithms, or the prediction may be used to initialize such algorithms, especially the bundle method whose performance heavily relies on the initial point used for stabilization. Moreover, since Lagrangian relaxation is used to provide dual bounds inside algorithms like branch-and-bound ones, our model may be used inside such algorithms to derive dual bounds. In particular, we predict Lagrangian multipliers and we solve the associated Lagrangian relaxed problem only once. We propose a generic model that can be applied to any mixed-integer linear problem and any Lagrangian relaxation providing tighter bounds than the continuous relaxation. Indeed, our model only uses as input the objective function, the constraint matrix and the right-hand side as well as the information relative to an optimal solution to the continuous relaxation in order to make the predictions.

The presented contributions give an overview of the research I have conducted so far but, since research never ends, the last chapter of this document presents some research perspectives. They correspond to some of my current and future works.

Part I

Mixed-Integer Linear Programming: Polyhedra and Box-Total Dual Integrality

Chapter 1

Preliminaries

In this chapter, we give some notions related to graphs and polyhedra that are necessary to present the research of this part.

1.1 Preliminaries on Graphs

In these works, we consider undirected graphs. We refer to (Bondy and Murty, 2008) for classic definitions and notation which are not presented in here.

Graphs A graph is a pair (V, E) where V is a finite set and E is a family of unordered pairs of V. The elements of V are called *nodes* whereas those of E are called *edges*. The two nodes of an edge are its *extremities*. An edge with extremities u and v is denoted by uv. Two nodes u and v are *adjacent* if uv is an edge. A node u and an edge e are *incident* if u is an extremity of e. Similarly, two edges e and f are incident if they share a same extremity. The *degree* of a node is the number of edges that are incident to it. An edge whose extremities coincide is a *loop*. A graph with no loop is said *loopless*. Two edges having the same extremities are *parallel*. A loopless graph with no parallel edges is said *simple*. A simple graph (V, E) is *complete* all pairs of distinct nodes are adjacent.

Given a node subset $W \subseteq V$, $E[W]$ denotes the set of edges having both extremities in W. Similarly, for $F \subseteq E$, $V[F]$ corresponds to the set of nodes that are the extremities of an edge of F.

Subgraphs A graph $G' = (V', E')$ is a *subgraph* of a $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. G' is a *spanning* subgraph of G is $V' = V$. For $W \subseteq V$, $(W, E[W])$ is the subgraph *induced* by W.

Bipartite graphs A graph $G = (V, E)$ is *bipartite* if its node set V can be partitioned into two sets U and W such that each edge has one extremity in U

and the other in W. If E contains an edge uw for each $u \in U$ and each $w \in W$, then G is a *complete bipartite graph* and is denoted $K_{|U|,|W|}$.

Minors The *contraction* of an edge $uv \in E$ consists in removing it, and identifying nodes u and v. A graph H is a minor of G if H may be obtained from G by contracting edges and removing nodes and edges.

Planarity and dual graphs A graph is *planar* if it can be drawn on the plane in such a way that no edges cross each other. Such a drawing is called a *planar embedding* of the graph. A planar embedding of a graph G divides the plane into a set of regions called *faces*. Among them, there is only one infinite face. The edges delimiting a face F and their extremities form the *boundary* of F. The boundary of the infinite face is the *outer boundary*.

Planar graphs are characterized by minors as stated in the following theorem, where K_n denotes the complete graph on n nodes.

Theorem 1.1 (Wagner, 1937)**.** *A graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.*

Let $G = (V, E)$ be a planar graph and fix a planar embedding of G. The *dual graph $G^\star$* of G with respect to this planar embedding is the graph having a node for each face of G, and an edge $e^\star$ for each edge $e \in E$ such that the extremities of $e^\star$ correspond to the faces having e in their boundary.

Note that $G^\star$ is planar. Moreover, $(G^\star)^\star$ is isomorphic to G if and only if G is connected.

Outerplanar and series-parallel graphs A well-known subclass of planar graphs is the class of series-parallel graphs which can be defined as follows. A 2-connected graph is *series-parallel* if it can be built from the circuit of length two, that is the graph corresponding to two nodes linked by two parallel edges, by repeatedly applying the following operations:

- *parallelization*: add a parallel edge to an existing one,
- *subdivision*: replace an edge by a path of length two.

A graph is *series-parallel* if all its 2-connected components are. These graphs have been characterized by forbidden minors, as stated in the following theorem.

Theorem 1.2 (Duffin, 1965)**.** *A graph is series-parallel if and only if it does not contain K_4 as a minor.*

Consider a planar embedding of a planar graph G. By definition of minors and dual graphs, the dual of a minor of G is a minor of $G^\star$. The graph K_4 being its own dual, we get the following result.

Observation 1.1. *A planar graph G is series-parallel if and only if, given a planar embedding of G, the dual graph is series-parallel.*

A subclass of series-parallel graphs is the class of outerplanar graphs. A graph is *outerplanar* if it can be drawn in such a way that all nodes belong to the outer boundary. Theorem 1.3 gives a characterization of these graphs in terms of forbidden minors.

Theorem 1.3 (Chartrand and Harary, 1967)**.** *A graph is outerplanar if and only if it contains neither K_4 nor $K_{2,3}$ as a minor.*

Circuits, cuts, and bonds A set of edges $C \subseteq E$ is a *cycle* of G if it induces a graph in which each node has an even degree. A cycle is a *circuit* if it is empty or induces a connected graph in which every node has degree two. The graph $(V[E], E)$ where E is a circuit and $|E| = k$ is denoted by C_k.

A *multicut* is the set of all the edges between different classes of some partition $\{V_1, \dots, V_k\}$ of the node set V, and it is denoted by $\delta(V_1, \dots, V_k)$. For every multicut M, there exists a unique partition $\{V_1, \dots, V_\ell\}$ of V such that $M = \delta(V_1, \dots, V_\ell)$ and $G[V_i]$ is connected for all $i = 1, \dots, \ell$. The number of classes ℓ in this partition is called the *order of M* and is denoted by d_M. A set F of edges is a *cut* if $F = \delta(X, V \setminus X)$ for some $X \subseteq V$, that is, there exists a node set $X \subseteq V$ such that F is the set of edges having exactly one extremity in X. The cut F is denoted by $\delta(X)$. The node set X and its complement $V \setminus X$ are the *shores* of F. A cut is a *bond* if it is empty or it does not strictly contain any nonempty cut. By definition, each shore of a nonempty bond induces a connected graph. A bond and a circuit intersect an even number of times. Moreover, for series-parallel graphs, it has been shown that a bond and a circuit cannot intersect more than twice. These are stated in the following observation and theorem.

Observation 1.2. *A bond and a circuit intersect an even number of times.*

Theorem 1.4 (Chakrabarti, Fleischer, and Weibel, 2012)**.** *In a series-parallel graph, a bond and circuit intersect zero or twice.*

Note that a bond (resp. circuit) in a planar graph corresponds to a circuit (resp. bond) in the dual graph, as stated in the following.

Observation 1.3. *The bonds of a planar graph are in bijection with the circuits of the dual graph.*

1.2 Preliminaries on Polyhedra

A polyhedron is the intersection of a finite set of halfspaces. Each halfspace being defined by a linear inequality, a polyhedron P is the set of points satisfying a given set of inequalities, that is, $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. When A and b are rational, the polyhedron is *rational.* In this document, we will only consider rational polyhedra. When $b = \mathbf{0}$, the polyhedron is a *(polyhedral) cone.*

The *convex hull* of a set of points $S = \{p^1, \dots, p^k\}$ of $\mathbb{R}^n$, denoted by *conv*(S), is the set of points that are convex combination of $p^1, \dots, p^k$, that is:

$$conv(S) = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{k} \lambda_i p^i, \lambda \geq \mathbf{0}, \sum_{i=1}^{k} \lambda_i = 1\}.$$

The *conic hull* of a nonempty set of points $S = \{p^1, \dots, p^k\}$ of $\mathbb{R}^n$, denoted by $cone(S)$, is the set of points that are conic combination of $p^1, \dots, p^k$, that is:

$$cone(S) = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{k} \lambda_i p^i, \lambda \geq \mathbf{0}\}.$$

By convention, $cone(\emptyset) = \{\mathbf{0}\}$.

A polyhedron can be described either by linear inequalities or as the Minkowski sum of a convex hull and a cone, where the *Minkowski sum* of two sets X and Y is $X + Y = \{x + y \mid x \in X, y \in Y\}$. This result is known as the Minkowski-Weyl's Theorem (see Theorem 11.10 of (Conforti, Cornuéjols, and Zambelli, 2010) for a proof):

Theorem 1.5. *Let P be a subset of $\mathbb{R}^n$. P is a polyhedron, that is, $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ if and only if there exist points $p^1, \dots, p^k$ and $r^1, \dots, r^\ell$ of $\mathbb{R}^n$ such that $P = conv(\{p^1, \dots, p^k\}) + cone(\{r^1, \dots, r^\ell\})$.*

A bounded polyhedron P, that is, a polyhedron that can be described as $P = conv(S)$ for some (finite) set $S \subseteq \mathbb{R}^n$, is called a *polytope.* A *conic polyhedron* is a rational translation of a cone, that is, a set of the form $w + \{x \in \mathbb{R}^n \mid Ax \leq \mathbf{0}\}$ for some $w \in \mathbb{Q}^n$.

Rays, faces and facets Let C be a cone and $r \in C \setminus \{0\}$. The set $cone(r) = \{\lambda r : \lambda \geq 0\}$ is a *ray* of C. With a slight abuse of notation, r is sometimes called a ray and refers in this case to $cone(r)$. Two rays r^1 and r^2 of $C \setminus \{0\}$ are distinct if there does not exist $\mu \in \mathbb{R}_+$ such that $r^1 = \mu r^2$. A ray r of $C \setminus \{0\}$ is *extreme* if there does not exist distinct rays r^1 and r^2 such that $r = r^1 + r^2$.

A *face* F of $P = \{x \mid Ax \leq b\}$ is the polyhedron obtained from P by setting some inequalities of $Ax \leq b$ to equality. A *proper face* of P is a nonempty face different from P. The *dimension* of a face is the dimension of its affine hull, that is, the number of affinely independent points of F minus one. A *facet* is a face that is not contained in any other proper face and a *minimal face* is a face containing no other proper face. A minimal face of dimension zero is a *vertex.*

An inequality $a^\top x \leq b$ is *tight* for a face F if every point of F satisfies this inequality with equality. Given a face of a polyhedron $P = \{x \mid Ax \leq b\}$, the *tangent cone* of a face F is the conic polyhedron described by the inequalities of $Ax \leq b$ that are tight for F. The tangent cone associated with a minimal face of P is a *minimal tangent cone of P.*

A polyhedron P is *integer* if all its faces contain an integer point.

Polar cones Let C be a polyhedral cone of $\mathbb{R}^n$. The *polar cone* of C, denoted C^*, is the cone defined by:

$$C^* = \{y \in \mathbb{R}^n \mid y^\top x \leq 0 \text{ for all } x \in C\}.$$

The polar of the polar cone of C is C itself, that is, $C^{**} = C$. Moreover, if $C = \{x \mid Ax \leq 0\}$, then C^* is the cone generated by the transposes of the rows of A.

1.2.1 Extended Formulations

Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be a polyhedron. A polyhedron

$$Q = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^p \mid Bx + Cy \leq d \right\}$$

is an *extended formulation* of P if its projection onto the x-variables

$$proj_x(Q) = \left\{ x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^p \text{ s.t. } \begin{pmatrix} x \\ y \end{pmatrix} \in Q \right\}$$

is equal to P.

Extended formulations are useful for optimizing over P since the two linear problems provide the same optimum:

$$\max \left\{ c^\top x \mid x \in P \right\} = \max \left\{ c^\top x + 0^\top y \mid \begin{pmatrix} x \\ y \end{pmatrix} \in Q \right\}.$$

Sometimes, a polyhedron may be easier to be described in a higher dimension, that is, it needs fewer inequalities to be described when adding variables. The minimum number of inequalities in an extended formulation of a polytope is called its *extension complexity.*

In the example of Figure 1.1[1], the hexagon needs six inequalities to be described in two dimensions but only five in three dimensions.

The difference between the minimum number of inequalities to describe a polytope and its complexity extension may be dramatic. Indeed, several polytopes needs an exponential number of inequalities in their original space but have a polynomial complexity extension. This means that they can be described by a polynomial number of variables and constraints. For instance, the cut polytope for graphs with no K_5-minor contains an exponential number of inequalities (Barahona and Mahjoub, 1986) when considering a variable per edge but has a polynomial compact extension (Barahona, 1993). We prove that this also holds for the circuit and bond polytopes in series-parallel graphs (Borne et al., 2015).

[1]This figure is to be credited to Roland Grappe.

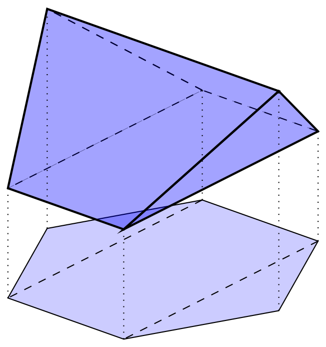

Figure 1.1: The hexagon as the projection of a prism.

Projection cone Let $Q = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^p \mid Ax + By \leq b \right\}$ be a polyhedron with $A \in \mathbb{Q}^{m \times n}$, $B \in \mathbb{Q}^{m \times p}$ and $b \in \mathbb{Q}^m$. To obtain the linear description of the projection of Q onto the x variables, one may consider the projection cone $C = \{u \in \mathbb{R}^m \mid u^\top B = \mathbf{0}, u \geq \mathbf{0}\}$. Indeed, for any $u \in C$, the inequality $u^\top Ax \leq u^\top b$ is valid for $proj_x(Q)$. Moreover, the set of such inequalities gives the description of the projection of Q, as stated in the following theorem (a proof can be found in (Conforti, Cornuéjols, and Zambelli, 2013)).

Theorem 1.6. *For a polyhedron* $Q = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^p \mid Ax + By \leq b \right\}$*, we have*

$$proj_x(Q) = \{x \in \mathbb{R}^n \mid u^\top Ax \leq u^\top b \text{ for all extreme rays } u \in C\},$$

where $C = \{u \in \mathbb{R}^m \mid u^\top B = \mathbf{0}, u \geq \mathbf{0}\}$.

Fourier's method To project a polyhedron, one can iteratively use the Fourier's method that eliminates a variable[2]. This method can be described as follows.

Let $Q = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R} \mid A^+x + y \leq b^+, A^-x - y \leq b^-, A'x \leq b' \right\}$ with $A^+ \in \mathbb{Q}^{m^+ \times n}$, $A^- \in \mathbb{Q}^{m^- \times n}$ and $A' \in \mathbb{Q}^{m' \times n}$. Note that any rational polyhedron can be put in this form by multiplying each inequality by the appropriate value such that the coefficient of y is either 1, 0 or -1.

Let $A''x \leq b''$ be the system given by adding a constraint of $A^+x + y \leq b^+$ with a constraint of $A^-x - y \leq b^-$, that is, $A''x \leq b''$ corresponds to the set of constraints:

$$A_i^+x + A_j^-x \leq b_i^+ + b_j^- \text{ for all } i \in \{1, \ldots, m^+\}, j \in \{1, \ldots, m^-\},$$

where for some matrix A, A_i denotes the ith row of A.

[2]This method is also known as the Fourier-Motzkin's elimination method.

The projection of Q onto the x variables is given by the new inequalities as well as those of Q not containing y, as stated by the following theorem.

Theorem 1.7 (Fourier, 1826)**.** *The projection of Q onto the x variables equals:*

$$proj_x(Q) = \{x \in \mathbb{R}^n \mid A'x \le b', A''x \le b''\}.$$

Union of polytopes Balas (1998) gives a way to obtain a linear description in a higher dimensional space of the union of a finite number of nonempty polytopes from their respective linear description. We restate his theorem for the union of two nonempty polytopes in the following theorem.

Theorem 1.8 (Balas, 1998)**.** *Let $P_1 = \{x \in \mathbb{R}^n \mid A_1 x \le b_1\}$ and $P_2 = \{x \in \mathbb{R}^n \mid A_2 x \le b_2\}$ be two nonempty polytopes. Then, $conv(P_1 \cup P_2) = proj_x(Q)$, where Q is the following polytope:*

$$Q = \left\{ \begin{pmatrix} x \\ x_1 \\ x_2 \\ \lambda \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \;\middle|\; \begin{array}{l} x = x_1 + x_2, \\ A_1 x_1 \le \lambda b_1, \\ A_2 x_2 \le (1-\lambda) b_2, \\ 0 \le \lambda \le 1 \end{array} \right\}.$$

1.2.2 Total Dual Integrality

A rational system of linear inequalities $Ax \le b$ is *totally dual integral (TDI)* if the minimization problem in the linear programming duality:

$$\max\{c^\top x \mid Ax \le b\} = \min\{y^\top b \mid y^\top A = c, y \ge \mathbf{0}\}$$

admits an integer optimal solution for each integer vector c such that the maximum is finite. Edmonds and Giles (1977) provide a sufficient condition for a TDI system to describe an integer polyhedron, as stated in the following theorem.

Theorem 1.9 (Edmonds and Giles, 1977)**.** *If a rational system $Ax \le b$ with b integer is TDI, it describes an integer polyhedron.*

Giles and Pulleyblank (1979) prove that every rational polyhedron can be described by a TDI system. Moreover, every integer polyedron can be described by a TDI system with an integer right-hand side.

Schrijver system A rational system $Ax \le b$ is *minimally TDI* if any proper subsystem of $Ax \le b$ which defines the same polyhedron as $Ax \le b$ is not TDI.

Theorem 1.10 (Schrijver, 1981)**.** *If P is a full dimensional polyhedron, there exists a unique minimally TDI system $Ax \le b$ describing P with A integer. Moreover, b is integer if and only if P is integer.*

When P is a full dimensional polyhedron, the minimally TDI system $Ax \le b$ describing P is called the *Schrijver system* of P.

Total dual integrality and Hilbert bases A set of vectors $v^1, \dots, v^k$ of $\mathbb{R}^n$ forms a *Hilbert basis* if each integer of $cone(\{v^1, \dots, v^k\})$ can be expressed as the integer nonnegative sum of $v^1, \dots, v^k$, that is:

$$cone(\{v^1, \dots, v^k\}) \cap \mathbb{Z}^n = \{x \in \mathbb{Z}^n \mid x = \sum_{i=1}^{k} \lambda_i v^i, \lambda \in \mathbb{Z}^k_+\}.$$

Theorem 1.11 (Giles and Pulleyblank, 1979)**.** *The rational system $Ax \leq b$ is TDI if and only if for each face F of the polyhedron $P = \{x \mid Ax \leq b\}$, the rows of A associated with the inequalities that are tight for F form a Hilbert basis.*

Theorem 1.11 can be restricted to minimal faces. This gives the following characterization of TDI systems describing polyhedral cones.

Corollary 1.1. *A rational system $Ax \leq 0$ is TDI if and only if the rows of A form a Hilbert basis.*

1.2.3 Box-Total Dual Integrality

A stronger property than total dual integrality is box-total dual integrality, where a system $Ax \leq b$ is *box-total dual integral (box-TDI)* if

$$Ax \leq b, \quad \ell \leq x \leq u$$

is TDI for all rational vectors ℓ and u (with possible infinite components). On the contrary to TDIness, all polyhedra cannot be described by box-TDI systems. Moreover, the box-TDIness is a property relative to polyhedra, and not to systems since Cook (1986) proves the following theorem.

Theorem 1.12 (Cook, 1986)**.** *If a system is box-TDI, then all TDI systems describing the same polyhedron are box-TDI.*

A polyhedron that can de described by a box-TDI system is called a *box-TDI polyhedron.*

Chervet, Grappe, and Robert (2021) show that the box-TDIness of a cone is preserved by polarity, as stated in the following theorem.

Theorem 1.13 (Chervet, Grappe, and Robert, 2021)**.** *A polyhedral cone is box-TDI if and only if its polar cone is.*

There is a close relation between the box-TDIness of a polyhedron and the box-TDIness of its tangent cones. This result has been formally stated in (Chervet, Grappe, and Robert, 2021) as follows.

Theorem 1.14. *A polyhedron is box-TDI if and only if all its minimal tangent cones are.*

The *dominant* of a polyhedron P is the polyhedron given by $\{x \mid \exists z \in P \text{ with } z \leq x\}$. The box-TDIness of a polyhedron is transmitted to its dominant, as stated in the following theorem.

Theorem 1.15. *The dominant of a box-TDI polyhedron is box-TDI.*

Matricial characterization of box-TDIness Chervet, Grappe, and Robert (2021) propose a characterization of the box-TDIness of a polyhedron from a matricial point of view. We first introduce some definitions before stating their results.

Let $P \subseteq \mathbb{R}^n$ be a polyhedron and F be a face of P. A full row rank matrix $A \in \mathbb{R}^{m\times n}$ is *face-defining* for F if $\text{aff}(F) = \{x \in \mathbb{R}^n : Ax = b\}$ for some $b \in \mathbb{R}^m$.

A sufficient condition for a matrix to be face-defining is the following.

Observation 1.4. *Let $M \in \mathbb{Q}^{r\times n}$ be a full row rank matrix. Let P be a polyhedron of $\mathbb{R}^n$ and F be a face of P. If $F = \{x \in P \mid Mx = d\}$ for some d and $dim(F) = dim(\{x \in \mathbb{R}^n \mid Mx = d\})$, then M is face-defining for F.*

A matrix $A \in \mathbb{R}^{m\times n}$ is *equimodular* if it is full row rank and all its $m\times m$ non-zero determinants have the same absolute value. A matrix is *totally equimodular* if every set of linear independent rows of A forms an equimodular matrix.

Theorem 1.16 (Chervet, Grappe, and Robert, 2021)**.** *Let P be a polyhedron. The following statements are equivalent:*

(i) P is box-TDI.

(ii) Every face-defining matrix of every non-empty face of P which is not full-dimensional is equimodular.

(iii) Every non-empty face of P which is not full-dimensional has an equimodular face-defining matrix.

As a consequence of Theorem 1.16, totally equimodular matrices describe box-TDI polyhedra, as stated by the following corollary. Note that this result is already proved in (Chervet, Grappe, and Robert, 2021) but their proof does not rely on Theorem 1.16 on the contrary to ours.

Corollary 1.2 (Chervet, Grappe, and Robert, 2021)**.** *A rational matrix $A \in \mathbb{Q}^{m\times n}$ is totally equimodular if and only if the polyhedron $\{x \colon Ax \le b\}$ is box-TDI for all $b \in \mathbb{Q}^m$.*

Proof. ($\Rightarrow$) Suppose that A is totally equimodular. Let $P = \{x \mid Ax \le b\}$ for some b, and F be any non-empty face of P which is not full dimensional. F admits a face-defining matrix B which is submatrix of A. By definition of total equimodularity, B is equimodular. Hence, by Theorem 1.16, P is box-TDI.

($\Leftarrow$) Suppose that A is not totally equimodular. Hence, $A = \begin{pmatrix} A^= \\ A^\le \end{pmatrix}$ with $A^= \in \mathbb{Q}^{r\times n}$ being a non-equimodular full row rank matrix, and $A^\le \in \mathbb{Q}^{r'\times n}$. Let $x^1, \dots, x^{n-r+1}$ be affinely independent points of $\{x \in \mathbb{R}^n : A^= x = \mathbf{1}\}$. Let $b^\le \in \mathbb{Q}^{r'}$ be such that $b^\le_i > \{\max a_i x^j \mid j \in \{1, \dots, n-r+1\}\}$, where a_i denotes the i^{th} row of $A^\le$. Set $b = \begin{pmatrix} \mathbf{1} \\ b^\le \end{pmatrix}$. Clearly, $x^1, \dots, x^{n-r+1} \in F = \{x \mid A^= x = \mathbf{1}, A^\le x \le b^\le\}$ so, by Observation 1.4, $A^=$ is a face defining matrix of the face F of $P = \{x \mid Ax \le b\}$. By Theorem 1.16, P is not box-TDI. ☐

Note that Corollary 1.2 is the polyhedral counterpart of a well-known result establishing the box-TDIness of a system with respect to the total unimodularity of a matrix, as stated in the following theorem. Recall that a matrix is *totally unimodular* if all its subdeterminants are equal to 0, -1 or 1.

Theorem 1.17 (Schrijver, 1986)**.** *A matrix A of $\mathbb{Z}^{m\times n}$ is totally unimodular if and only if the system $Ax \leq b$ is box-TDI for all $b \in \mathbb{Q}^m$.*

Chapter 2

Linear Description of the Convex Hull of the Solutions to Mixed-Integer Linear Problems

This chapter presents some of my works related to the characterization of the convex hull of the solutions to some combinatorial optimization problems.

For a mixed-integer linear problem, having a linear description of the convex hull of its solutions permits to cast this problem as a (continuous) linear one: optimizing the objective function subject to the inequalities describing the convex hull. Even a partial description of the convex hull may be helpful to solve the problem. For instance, the efficiency of the concorde solver (Applegate et al., 2006) is mainly due to the partial description of the convex hull of the incidence vectors of the Hamiltonian circuits of a graph.

Obtaining a description of the convex hull of the solutions to an integer linear problem is usually challenging, even for problems that can be solved in polynomial time. A way to do it is to add variables since in higher dimension, the convex hull may be easier to describe. In this case, one obtains an extended formulation of the convex hull which can be projected to obtain a linear description in the original space. This is what has been done to obtain the description of the circuit polytope, that is the convex hull of the incidence vectors of all the circuits of a graph, for series-parallel graphs (Borne et al., 2015). The description of the convex hull of the vectors that are lexicographically between two given vectors has also been obtained using extended formulations (Barbato, Grappe, Lacroix, and Pira, 2018).

It is also possible to characterize a convex hull by providing a set of valid inequalities forming a polyhedron whose integer points are the solutions to the integer linear problem, and by proving that this polyhedron is integer. This is the approach we use to characterize the st-bond polytope when the graph is 2-connected and outerplanar (Grappe and Lacroix, 2018), where the st-bond polytope is the convex hull of the incidence vectors of minimally inclusionwise cuts separating two given nodes s and t.

In Section 2.1, we provide a linear description of the circuit polytope and the bond polytope when the graph is series-parallel. These descriptions come from a joint work with S. Borne, P. Fouilhoux, R. Grappe and P. Pesneau (Borne et al., 2015). We also give the linear description of the st-bond polytope for the same class of graphs. This result stems from a joint work with R. Grappe (Grappe and Lacroix, 2018). In Section 2.2, we provide a linear description of the lexicographical polytope, issued from a joint work with M. Barbato, R. Grappe and C. Pira (Barbato, Grappe, Lacroix, and Pira, 2018).

2.1 The Circuit and Bond Polytopes of a Series-Parallel Graph

In (Borne et al., 2015), we give the description of the bond polytope, that is, the convex hull of the incidence vectors of bonds, for series-parallel graphs.

A bond being a cut, the bond polytope is contained in the cut polytope, that is, the convex hull of the incidence vectors of cuts. In series-parallel graphs, and more generally in graphs with no K_5-minor, the cut polytope is the set of points satisfying the following inequalities (Barahona and Mahjoub, 1986):

$$x(F) - x(C \setminus F) \leq |F| - 1 \qquad \begin{array}{l}\text{for all circuits } C,\\ \text{for all } F \subseteq C \text{ with } |F| \text{ odd,}\end{array} \tag{2.1}$$

$$x \geq \mathbf{0}. \tag{2.2}$$

By Theorem 1.4, for series-parallel graphs, a bond intersects zero or twice each circuit. Hence, we have the following valid inequalities for the bond polytope:

$$x(C) \leq 2 \quad \text{for all circuits } C. \tag{2.3}$$

Inequalities (2.3) imply that each inequality (2.1) associated with an edge subset F containing more than one edge is redundant. Hence, only the following subset of inequalities (2.1) has to be considered for describing the bond polytope in series-parallel graphs:

$$x_e \leq x(C \setminus e) \quad \text{for all circuits } C, \text{ for all } e \in C. \tag{2.4}$$

However, inequalities (2.2), (2.3) and (2.4) are not sufficient to describe the bond polytope in series-parallel graphs. Indeed, Figure 2.1 shows a graph for which the polytope defined by (2.2), (2.3) and (2.4) has $\frac{1}{2}\chi^{E\setminus\{v_1v_2\}}$ as a non-integer vertex. This point satisfies all the inequalities, and it is the unique solution to the system defined by the following tight constraints: (2.2) associated with the edge v_1v_2, (2.3) associated with each circuit not containing v_1v_2, and

(2.4) associated with each circuit containing v_1v_2 and each edge of the circuit different from v_1v_2.

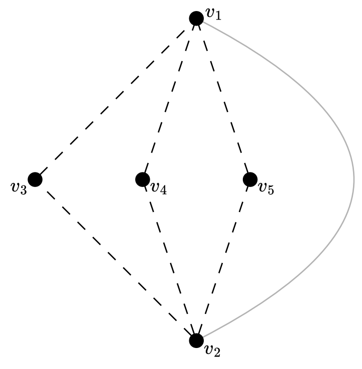

Figure 2.1: A fractional vertex of $\{x \in \mathbb{R}^8 : x$ satisfies (2.2), (2.3) and (2.4)$\}$: it has value 0 for the gray edge, and a value $\frac{1}{2}$ for each dotted edge.

In (Borne et al., 2015), we define a new family of valid inequalities based on ear decompositions. This family contains the circuit inequalities (2.3). They are sufficient to describe, together with (2.2) and (2.4) the bond polytope in series-parallel graphs.

Our proof relies on extended formulations. Indeed, by definition, a 2-connected series-parallel graph is obtained from C_2 by a series of parallelizations and subdivisions. One can describe in a straightforward way the set of bonds of G from those of H, when G is obtained from H by either adding a parallel edge or by subdividing an edge. Using the theorem of Balas on the union of polyhedra, we obtain the description of the bond polytope of G from the one of the bond polytope of H. By repeatedly applying this, we obtain a compact extended formulation for the bond polytope in series-parallel graphs. We then project the extended formulation at each step using Fourier's method and show that it always gives inequalities of type (2.2) or (2.4), or the new family defined on ear decompositions.

The bond polytope has been recently described for 3-connected graphs with no $(K_5 \setminus e)$-minor in (Chimani, Juhnke-Kubitzke, and Nover, 2023). For such graphs, inequalities (2.3) are not all valid since a bond may intersect some circuits more than twice. The authors characterize the subset of these inequalities that remain valid, and show that together with (2.2) and (2.4), it describes the bond polytope. This work does not intersect ours since series-parallel graphs are not 3-connected.

By Observation 1.3, given a planar embedding of a series-parallel graph G, the bonds are in bijection with the circuits of the dual graph $G^\star$. The dual graph of a series-parallel graph being series-parallel by Observation 1.1, we get the description of the circuit polytope, that is, the convex hull of the incidence vectors of circuits, for series-parallel graphs.

In (Grappe and Lacroix, 2018), we consider the *st*-bond polytope, that is, the convex hull of the set of bonds separating two given nodes s and t. We give its linear description for series-parallel graphs. For this, we start by showing that every nontrivial simple 2-connected series-parallel graph has at least two nodes of degree two. Moreover, when it has exactly two such nodes, it is outerplanar. We then describe the *st*-bond polytope in 2-connected outerplanar graphs and use this graph characterization to extend the description to series-parallel graphs.

Generally, the intersection of integer polyhedra is not integer. However, we show that the *st*-bond polytope in series-parallel graphs is nothing the but intersection of the bond polytope and the dominant of the cut polytope.

2.2 Lexicographical Polytopes

In (Barbato, Grappe, Lacroix, and Pira, 2018), we consider the lexicographical polytope. This latter was described in (Gupte, 2016) but we provide another shorter proof of this result using extended formulations.

Given two points x and y of $\mathbb{Z}^n$, x is *lexicographically less than or equal to* y, denoted by $x \preccurlyeq y$, if $x = y$ or the first coordinate of $x - y$ is negative. Given $\ell, u \in \mathbb{Z}^n$, the *lexicographical polytope* $P_{\ell,u}^{r\preccurlyeq s}$ is the convex hull of the integer points within $[\ell, u]$ that are lexicographically between r and s, that is,

$$P_{\ell,u}^{r\preccurlyeq s} = conv\{x \in \mathbb{Z}^n : \ell \leq x \leq u, r \preccurlyeq x \preccurlyeq s\}.$$

For simplicity, we write $P_{\ell,u}^{r\preccurlyeq s}$ as $P_{\ell,u}^{\preccurlyeq s}$ (resp. $P_{\ell,u}^{r\preccurlyeq}$) when $r = \ell$ (resp. $s = u$).

Figure 2.2 depicts the two dimensional lexicographical polytope $P_{\ell,u}^{r\preccurlyeq s}$ where $\ell = \begin{pmatrix}2\\2\end{pmatrix}$, $u = \begin{pmatrix}5\\5\end{pmatrix}$, $r = \begin{pmatrix}2\\3\end{pmatrix}$ $s = \begin{pmatrix}4\\4\end{pmatrix}$.

In our proof, we first consider the set $X_{\ell,u}^{\preccurlyeq s}$ of componentwise maximal integer points of $P_{\ell,u}^{\preccurlyeq s}$, that is the set of points $p^i = (s_1, \dots, s_{i-1}, s_i - 1, u_{i+1}, \dots, u_n)$, for $i = 1, \dots, n+1$ such that $s_i > \ell_i$ ($p^{n+1} = s$ by definition)[1]. We represent this set $X_{\ell,u}^{\preccurlyeq s}$ as the set of *st*-paths of a given acyclic digraph. This gives an extended formulation of the convex hull of $X_{\ell,u}^{\preccurlyeq s}$. We project it to obtain a linear description of $conv(X_{\ell,u}^{\preccurlyeq s})$ in the natural space. We then prove that $P_{\ell,u}^{\preccurlyeq s}$ is equal to the submissive of $conv(X_{\ell,u}^{\preccurlyeq s})$ intersected with $\{x \in \mathbb{R}^n : x \geq \ell\}$.

[1]p^n is not a componentwise maximal point of $P_{\ell,u}^{\preccurlyeq s}$ but it is still added to $X_{\ell,u}^{\preccurlyeq s}$ for simplification of the proof.

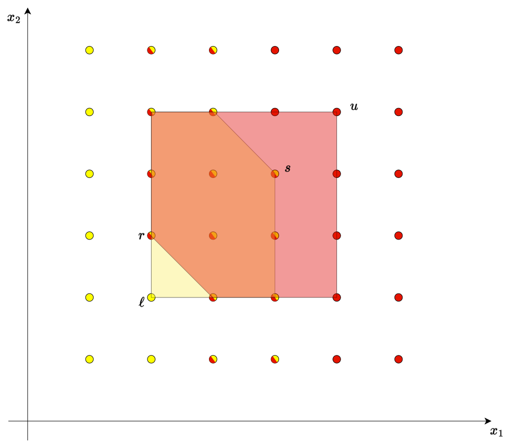

Figure 2.2: Yellow points are lexicographically less than or equal to s whereas red ones are lexicographically greater than or equal to ℓ. $P_{\ell,u}^{\preccurlyeq s}$ and $P_{\ell,u}^{r\preccurlyeq}$ are the yellow and red polytopes and their intersection $P_{\ell,u}^{r\preccurlyeq s}$ is in orange.

By symmetry, we obtain the description of $P_{\ell,u}^{r\preccurlyeq}$. We get the description of $P_{\ell,u}^{r\preccurlyeq s}$ by proving that it is the intersection of $P_{\ell,u}^{\preccurlyeq s}$ and $P_{\ell,u}^{r\preccurlyeq}$.

Chapter 3

Polyhedra and Box-Total Dual Integrality

This chapter presents my research works related to the box-TDIness of polyhedra and systems. Box-TDIness is a strong integrality property which is usually behind min/max relations such as the well-known max-flow/min-cut theorem (Ford and Fulkerson, 1956). Finding box-TDI systems and polyhedra is a tough task, and a research area consists in exhibiting such polyhedra and systems.

In Section 3.1, we study the complexity of some problems related to box-TDI polyhedra. We prove that optimizing a linear function over the integer points of a box-TDI polyhedra is NP-hard. We also show that determining whether a linear system describes a box-TDI polyhedra is a co-NP-complete problem. This answers Question (90) in (Schrijver, 1986) that was open for almost forty years. These results come from our characterization of the total equimodularity of the incidence matrix of a graph. These results are published in a joint work with P. Chervet, R. Grappe, F. Pisanu and R. Wolfler Calvo (Chervet, Grappe, Lacroix, et al., 2023). Section 3.2 is devoted to the study of box-TDIness of systems and polyhedra related to combinatorial objects defined on graphs. We characterize the box-TDIness of these systems and polyhedra with respect to series-parallel graphs. In Section 3.2.1, we show that the systems describing the cycle cone, the T-join polytope and its dominant are box-TDI if and only if the graph is series-parallel. These systems were known to be TDI (Schrijver, 2003) but we extend these results by proving that they are box-TDI whenever they are TDI. These results come from a joint work with D. Cornaz and R. Grappe (Cornaz, Grappe, and Lacroix, 2019). In Section 3.2.2, in a joint work with M. Barbato, R. Grappe, E. Lancini and R. Wolfler Calvo (Barbato, Grappe, Lacroix, Lancini, and Wolfler Calvo, 2022) we provide the Schrijver system of the flow cone when the graph is series-parallel. In Section 3.2.3, we study the k-edge connected spanning subgraph polyhedron. We prove that it is box-TDI if and only if the graph is series-parallel. Moreover, we provide a box-TDI system describing it for such graphs. These results stem from a joint work with M. Barbato, R. Grappe, and E. Lancini (Barbato, Grappe, Lacroix, and Lancini, 2023).

3.1 Hard Problems on Box-Totally Dual Integral Polyhedra

In (Chervet, Grappe, Lacroix, et al., 2023), we characterize when the incidence matrix of a graph is totally equimodular. We use this characterization to prove the NP-hardness of two problems related with box-TDI polyhedra.

Given a graph $G = (V, E)$, let A_G denote the *edge-vertex incidence matrix*, that is, the matrix whose rows are the characteristic vectors of the edges of G, where the *characteristic vector of an edge* $e = uv$ is the vector $\chi^e \in \{0,1\}^V$ with $\chi^e_w = 1$ if $w \in \{u, v\}$ and $\chi^e_w = 0$ otherwise. The *vertex-edge incidence matrix* is the transpose of A_G.

Hoffman and Kruskal characterize when the vertex-edge matrix incidence is totally unimodular, as stated in the following theorem.

Theorem 3.1 (Hoffman and Kruskal, 2010)**.** *The vertex-edge incidence matrix of a graph is totally unimodular if and only if the graph is bipartite.*

Since, by definition, the transpose of a totally unimodular matrix is also totally unimodular, and a totally unimodular matrix is totally equimodular, it follows that when G is bipartite, A_G is totally equimodular. On the contrary to total unimodularity, we prove that A_G remains totally equimodular when G is not bipartite.

Theorem 3.2. *The edge-vertex incidence matrix of a graph is totally equimodular.*

Given an undirected graph, a *stable set* is a set of pairwise nonadjacent nodes. The stable set polytope of a graph is the convex hull of the incidence vectors of its stable sets. The polytope $\{x \in \mathbb{R}^V \mid A_G x \leq \mathbf{1}, x \geq \mathbf{0}\}$ is called the *edge relaxation of the stable set polytope* of G and its integer points are precisely the incidence vectors of the stable sets of G. Since the intersection of a box-TDI polyhedron with $\mathbb{R}^V_+$ is box-TDI, Theorem 1.2 and Theorem 3.2 imply that the edge relaxation of the stable set polytope is a box-TDI polyhedron. Since the stable problem is NP-hard (Karp, 1972), we get the following NP-hardness result.

Corollary 3.1. *Given a box-TDI polyhedron P and a cost vector c, finding an integer point x maximizing $c^\top x$ over P is NP-hard.*

By Theorem 3.1, $A_G^\top$ is totally equimodular for bipartite graphs. If G is an odd hole, then $A_G^\top$ is also the edge-vertex incidence matrix of an odd hole, and hence it is totally equimodular by Theorem 3.2. We prove that these are the only cases when $A_G^\top$ is totally equimodular.

Theorem 3.3. *The vertex-edge incidence matrix of a connected graph $G = (V, E)$ is totally equimodular if and only if G is an odd hole or a bipartite graph.*

The problem of deciding whether a system $Ax \leq b$ with A integer is TDI is in co-NP (Schrijver, 1986). Ding, Feng, and Zang (2008) show that this problem is actually co-NP complete.

For this, the authors define the class of quasi-bipartite graphs, where a graph is *quasi-bipartite* if for each odd circuit C of G, the graph $G \setminus V(C)$ has at least one isolated node. They prove that determining whether a graph is quasi-bipartite is a NP-complete problem. They also prove the following equivalence.

Theorem 3.4 (Ding, Feng, and Zang, 2008)**.** *Given a connected graph G, the system $A_G^\top x \geq \mathbf{1}, x \geq \mathbf{0}$ is box-TDI if and only if G is a quasi-bipartite graph different from K_4.*

Given a graph $G = (V, E)$, an *edge cover* is a set of edges covering each node. The polyhedron $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}, x \geq \mathbf{0}\}$ is called the *edge relaxation of the edge cover dominant* of G and its binary points are precisely the characteristic vectors of the edge covers of G.

When G is an odd hole, the edge relaxation of the edge cover dominant is box-TDI by Theorems 3.3 and 1.2, since when a polyhedron is box-TDI, so is its intersection with $\mathbb{R}_+^E$. We prove that this is the only case where this polyhedron is box-TDI but the system $A_G^\top x \geq \mathbf{1}, x \geq \mathbf{0}$ is not TDI, as stated in the following theorem.

Theorem 3.5. *The edge relaxation of the edge cover dominant of a connected graph G is box-TDI if and only if G is an odd hole or a quasi-bipartite graph different from K_4.*

The proof of Theorem 3.5 mainly consists in proving that when there exists an odd circuit C such that $G \setminus V(C)$ has no isolated node, the matrix M corresponding to $A_G^\top$ restricted to the rows associated with nodes of $V(C)$ is a face defining matrix of the edge relaxation of the edge cover which is not equimodular.

Cook (1986) shows that, given a rational system, the problem of determining whether it describes a box-TDI polyhedron is in co-NP. As a corollary of Theorem 3.5, since recognizing whether a graph is quasi-bipartite is an NP-complete problem (Ding, Feng, and Zang, 2008), we have the following result.

Corollary 3.2. *Determining whether a rational system describes a box-TDI polyhedron is a co-NP-complete problem.*

3.2 Series-Parallel Graphs and Box-TDIness

This section presents several works related to the study of the box-TDIness of some polyhedra and systems in series-parallel graphs. These works lead to several characterizations of series-parallel graphs.

3.2.1 Characterization of Series-Parallel Graphs in terms of Box-TDI Systems

Series-parallel graphs are known be the class of graphs for which a certain number of linear systems are TDI. In (Cornaz, Grappe, and Lacroix, 2019), we strengthen this result by showing that series-parallel graphs are the class of graphs for which these systems are box-TDI. We start this section by presenting the different systems.

Seymour (1979) proves that the *cycle cone* of G, that is, the set of nonnegative combinations of cycles of G, is described by the following set of inequalities.

$$\text{(Cycle cone)}\begin{cases} x(\delta(U)\setminus\{e\}) - x_e \geq 0 & \text{for each } U\subseteq V \text{ and each } e\in\delta(U),\\ x\geq \mathbf{0}. \end{cases}$$

For a graph $G=(V,E)$ and an even subset $T\subseteq V$, a *T-join* is a set of edges inducing a graph in which the set of nodes of odd degree is T. Note that a $\emptyset$-join is a cycle. The *T-join polytope* of G is the convex hull of its T-joins. Seymour (1981) proves that it is described by the following set of inequalities.

$$(T\text{-join})\begin{cases} x(F) - x(\delta(U)\setminus F) \leq |F|-1 & \begin{array}{l}\text{for each } U\subseteq V,\ F\subseteq\delta(U)\\ \text{with } |U\cap T|+|F| \text{ odd,}\end{array}\\ \mathbf{0}\leq x\leq \mathbf{1}. \end{cases}$$

The T-join dominant of G is described by the following set of inequalities, see Corollary 29.2b in (Schrijver, 2003), where a T-cut is a cut $\delta(U)$ with $|T\cap U|$ odd.

$$(T\text{-join dominant})\begin{cases} x(C)\geq 1 & \text{for each } T\text{-cut } C,\\ x\geq\mathbf{0}. \end{cases}$$

Schrijver (2003) shows that Systems (Cycle cone), (T-join) and (T-join dominant) are TDI if and only if the graph is series-parallel, see Corollary 29.9c.

Theorem 3.6 (Schrijver, 2003)**.** *For any graph $G=(V,E)$, the following statements are equivalent:*

(i) G is series-parallel,

(ii) System (T-join dominant) is TDI for each choice of T,

(iii) System (T-join) is TDI for each choice of T,

(iv) System (T-join) is TDI for some choice of T,

(v) System (Cycle cone) is TDI.

In (Cornaz, Grappe, and Lacroix, 2019), we strengthen this result by showing that the aforementioned systems are actually box-TDI in series-parallel graphs.

Theorem 3.7. *For any graph $G = (V, E)$, the following statements are equivalent:*

(i) G is series-parallel,

(ii) System (T-join dominant) is box-TDI for each choice of T,

(iii) System (T-join) is box-TDI for each choice of T,

(iv) System (T-join) is box-TDI for some choice of T,

(v) System (Cycle cone) is box-TDI.

By Theorem 3.7, it follows that for series-parallel graphs the T-join polytope, its dominant and the cycle cone are box-TDI polyhedra. However, it does not answer whether these polyhedra remain box-TDI whenever the graph is not series-parallel, as these systems are no more TDI in this case by Theorem 3.6.

Chervet, Grappe, and Robert (2021) almost answer this question by proving that the cycle cone is not box-TDI whenever the graph is not series-parallel. Schrijver (2003) shows that the minimal tangent cones of the T-join polytope are nothing but the cycle cone up to translations and axial rotations. Since these operations preserve box-TDIness, by Theorem 1.14, the T-join polytope is not box-TDI for each T whenever the graph is not series-parallel. We complement the answer by showing that when G is not series-parallel, the T-join dominant is not box-TDI for all $T \subseteq V$. This proof is similar to the ones given in (Chervet, Grappe, and Robert, 2021) and (Barbato, Grappe, Lacroix, and Lancini, 2023) for proving the non box-TDIness of the flow cone and the k-edge-connected subgraph polyhedron for $k \geq 2$.

Proposition 3.1. *For a non series-parallel graph $G = (V, E)$, there exists $T \subset V$ with T even such that the T-join dominant is not box-TDI.*

Proof. Since G is not series-parallel, there exist node sets $V_1, \dots, V_4$ such that $G[V_i]$ is connected for $i \in \{1, \dots, 4\}$, and $V_i \cap V_j = \emptyset$ and $\delta(V_i) \cap \delta(V_j) \neq \emptyset$ for $i \neq j \in \{1, \dots, 4\}$. Let $v_i \in V_i$ for $i = \{1, \dots, 4\}$ and set $T = \{v_1, \dots, v_4\}$. Let $e_1, \dots, e_6$ be edges belonging to distinct $\delta(V_i) \cap \delta(V_j) \neq \emptyset$ for $i \neq j \in \{1, \dots, 4\}$ as given in Figure 3.1 (we suppose that $e_k \in E_k$ for $k = 1, \dots, 6$). We show that the T-join dominant is not box-TDI by exhibiting a non equimodular matrix face-defining matrix, see Theorem 1.16.

Consider the matrix $M \in \{0, 1\}^{3 \times |E|}$ whose rows are the transpose of $\chi^{\delta(V_i)}$ for $i = 1, 2, 3$. M is full row rank by construction. Note that $\delta(V_i)$ is a T-cut for $i = 1, 2, 3$, and let F be the face of the T-join dominant given by setting to equality the constraints $x(\delta(V_i)) \geq 1$ for $i = 1, 2, 3$. We now exhibit $|E| - 2$ affinely independent points of F. Let $B = E \setminus \cup_{i=1}^{4} \delta(V_i)$. The points $p_1 = \chi^{B \cup \{e_1, e_6\}}$, $p_2 = \chi^{B \cup \{e_2, e_5\}}$, $p_3 = \chi^{B \cup \{e_3, e_4\}}$ and $\chi^{B \cup \{e_4, e_5, e_6\}}$ belong to F and are affinely independent. For each $e \in E_k \setminus e_k$, consider the point given by $p - \chi^{e_k} + \chi^{e}$ where p is the point among p_1, p_2 and p_3 having a one in the coordinate associated with e_k. Finally, for each $e \in B$, consider the point

$p_1 + \chi^e$. All these points are affinely independent and belong to F. Hence, by Observation 1.4, M is face defining for F.

The matrix M restricted to columns associated with $e_1, \ldots, e_6$ has the following form:

$$\begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline \chi^{\delta(V_1)} & 1 & 1 & 0 & 1 & 0 & 0 \\ \chi^{\delta(V_2)} & 1 & 0 & 1 & 0 & 1 & 0 \\ \chi^{\delta(V_3)} & 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

The first three columns form a matrix with a determinant equal to two whereas the last three ones form a matrix whose determinant is one. Hence, M is not equimodular, which ends the proof. □

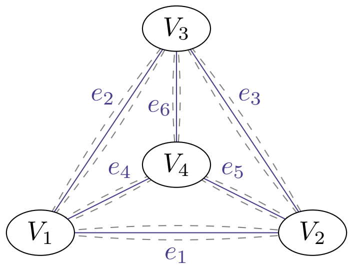

Figure 3.1: $E_1, \ldots, E_6$ are non-empty edge sets. Each edge $e_k \in E_k$ for $k = 1, \ldots, 6$ is represented with a solid line.

3.2.2 The Schrijver System of the Flow Cone in Series-Parallel Graphs

The cut cone of a graph, that is, the conic hull of the incidence vectors of its cuts, is box-TDI if and only if the graph is series-parallel (Chervet, Grappe, and Robert, 2021). The polar cone of its opposite is the flow cone. Since polarity preserves box-TDIness, the flow cone is box-TDI if and only if the graph is series-parallel. For such graphs, we provide the Schrijver system of the flow cone.

Given a graph $G = (V, E)$, a *flow* of G is a couple (C, e) with C a circuit of G and e an edge of C. In a flow (C, e), the edge e represents a demand and $C \setminus e$ represents the path satisfying this demand. The *incidence vector* of a flow (C, e) is the $0/\pm 1$ vector $\chi^{C \setminus e} - \chi^e$. The *flow cone* of G is the cone generated by the flows of G and the unit vectors χ^e of $\mathbb{R}^E$. Seymour (1981) shows that the flow cone is the polar of the opposite cone of the cut cone if and only if G has no K_5-minor. As a consequence, we have the following:

Corollary 3.3. *The flow cone of a graph G is*

$$\{x \in \mathbb{R}^E \mid x(D) \geq 0 \textit{ for all cuts } D \textit{ of } G\}$$

if and only if G has no K_5-minor.

Chervet, Grappe, and Robert (2021) show that the flow cone associated with a graph G is box-TDI if and only if G is series-parallel. Since this cone is full dimensional as it contains $\mathbf{0}$ and χ^e for all $e \in E$, by Theorem 1.10, the flow cone can be described by a box-TDI Schrijver system when G is series-parallel. We exhibit such a system in this case.

Note that the system:

$$x(D) \geq 0 \quad \text{for all cuts } D \text{ of } G,$$

which describes the flow cone for series-parallel graphs is not a Schrijver system since it is not TDI for this class of graph. Indeed, if $G = K_3$, for an objective function equal to $-\mathbf{1}$, the (linear programming) dual is:

$$\begin{aligned} &\min \mathbf{0} \\ &y_1 + y_2 = 1, \\ &y_1 + y_3 = 1, \\ &y_2 + y_3 = 1, \\ &y \geq \mathbf{0}, \end{aligned}$$

and its unique solution is $y = \frac{\mathbf{1}}{\mathbf{2}}$. However, Chervet, Grappe, and Robert (2021) prove that dividing each inequality by two yields a box-TDI system describing the flow cone when G is series-parallel:

$$\frac{1}{2}x(B) \geq 0 \quad \text{for all bonds } B \text{ of } G.$$

The incidence vector of a multicut $M = \delta(V_1, \ldots, V_k)$ being the half sum of the incidence vectors of the cuts $\delta(V_i)$ for $i = 1, \ldots, k$, it follows that the flow cone is described by:

$$x(M) \geq 0 \quad \text{for all multicuts } M \text{ of } G. \tag{3.1}$$

In (Barbato, Grappe, Lacroix, Lancini, and Wolfler Calvo, 2022), we prove that the incidence vectors of multicuts form a Hilbert Basis if and only if G is series-parallel. By Corollary 1.1, this implies that inequalities (3.1) defines a TDI system if and only if G is series-parallel.

A multicut $M \subseteq E$ is *chordal* if contracting the edges of $E \setminus M$ yields a 2-connected chordal graph. In (Barbato, Grappe, Lacroix, Lancini, and Wolfler Calvo, 2022), we prove that in series-parallel graphs, a multicut is the union of multicuts if and only if it is not chordal. As a consequence, it is sufficient to only consider the inequalities associated with chordal multicuts in the TDI system describing the flow cone, as stated as follows.

Corollary 3.4 (Barbato, Grappe, Lacroix, Lancini, and Wolfler Calvo, 2022)**.** *The Schrijver system for the flow cone of a series-parallel graph G is the following:*

$$x(M) \geq 0 \quad \textit{for all chordal multicuts } M \textit{ of } G. \tag{3.2}$$

Moreover, this system is box-TDI.

3.2.3 Edge Connectivity

In (Barbato, Grappe, Lacroix, and Lancini, 2023), we provide a new characterization of series-parallel graphs related to the box-TDIness of the k-edge-connected spanning subgraph polyhedron. We also provide two integer box-TDI systems describing this polyhedron in series-parallel graphs, depending on the parity of k.

A *k-edge-connected spanning subgraph* of a graph $G = (V, E)$ is a graph $H = (V, F)$, with F a multiset of elements of E, that remains connected after the removal of any $k-1$ edges. The *k-edge-connected spanning subgraph polyhedron* of G, hereafter denoted by $P_k(G)$, is the convex hull of all the k-edge-connected spanning subgraphs of G.

Note that the 1-edge-connected spanning subgraph polyhedron is the dominant of the spanning tree polytope. This latter being box-TDI, $P_1(G)$ is box-TDI by Theorem 1.15 (see (Lancini, 2019) for such a proof).

Didi Biha and Mahjoub (1996) give a description of the k-edge-connected spanning subgraph polyhedron in series-parallel graphs for all k.

Theorem 3.8 (Didi Biha and Mahjoub, 1996)**.** *Let G be a series-parallel graph and h be a positive integer. Then $P_{2h}(G)$ is described by:*

$$(3.3)\begin{cases} x(D) \geq 2h & \text{for all cuts } D \text{ of } G, & (3.3a)\\ x \geq \mathbf{0}, & & (3.3b)\end{cases}$$

and $P_{2h+1}(G)$ is described by:

$$(3.4)\begin{cases} x(M) \geq (h+1)d_M - 1 & \text{for all multicuts } M \text{ of } G, & (3.4a)\\ x \geq \mathbf{0}. & & (3.4b)\end{cases}$$

Chen, Ding, and Zang (2009) prove that dividing by two the cut inequalities (3.3a) in System (3.3) gives a system for $P_2(G)$ that is box-TDI if and only if G is series-parallel.

Theorem 3.9 (Chen, Ding, and Zang, 2009)**.** *The system*

$$(3.5)\begin{cases} \frac{1}{2}x(D) \geq 1 & \text{for all cuts } D \text{ of } G, & (3.5a)\\ x \geq \mathbf{0}, & & (3.5b)\end{cases}$$

is box-TDI if and only if G is series-parallel.

Theorem 3.9 implies that $P_2(G)$ is a box-TDI polyhedron when G is series-parallel. For even $k \geq 4$, by Theorem 3.8, $P_k(G)$ is obtained by multiplying the right-hand side of each inequality of System (3.5) by $\frac{k}{2}$. Since this operation preserves box-TDIness, it follows that $P_k(G)$ is a box-TDI polyhedron when G is series-parallel and k is even. We extend this result by proving the following.

Theorem 3.10. *For $k \geq 2$, $P_k(G)$ is box-TDI if and only if G is series-parallel.*

The proof of Theorem 3.10 relies on the matricial characterization of box-TDI polyhedra. We first exhibit a non equimodular face defining matrix when G has a K_4-minor. This implies that $P_k(G)$ is not box-TDI whenever G is not series-parallel by Theorems 1.16 and 1.2.

To prove that $P_k(G)$ is box-TDI in series-parallel graphs for k odd, we show that the polyhedron remains box-TDI when adding a parallel edge or when subdividing an edge in G. We also show that the 1-sum of two graphs preserves the box-TDIness of the k-edge connected spanning subgraph polyhedron, where the *1-sum* of two graphs G and H is the graph obtained by identifying a node of G with a node of H. More precisely, we prove that if $P_k(G)$ and $P_k(H)$ are box-TDI for two graphs G and H, the k-edge connected spanning subgraph polyhedron of any graph corresponding to the 1-sum of G and H is also box-TDI. By definition of series-parallel graphs, since $P_k(K_2)$ is box-TDI, the result follows.

We also provide integer TDI systems describing $P_k(G)$, depending on the parity of k, see Theorems 3.11 and 3.12.

Theorem 3.11. *For a positive integer h, the system*

$$(3.6)\begin{cases} x(M) \geq hd_M & \text{for all multicuts } M \text{ of } G, & (3.6a)\\ x \geq \mathbf{0}, & & (3.6b)\end{cases}$$

is box-TDI if and only if G is series-parallel.

The proof of Theorem 3.11 is as follows. By Theorem 3.10, it is sufficient to prove that System (3.6) is TDI when G is series-parallel. For this, we prove that $\{\chi^M \mid M \text{ is a multicut of } G\} \cup \{\chi^e \mid e \in E\}$ form a Hilbert basis which gives the desired result by Theorem 1.11. The proof is done by induction on the number of edges. More precisely, we show that the incidence vectors of the multicuts and edges of a graph G form a Hilbert basis if G is obtained from H by adding a parallel edge or by subdividing one, and if the incidence vectors of the multicuts and edges of H form a Hilbert basis. Similarly, we show that when G is the 1-sum of H_1 and H_2, the incidence vectors of the multicuts and edges of G form a Hilbert basis if those of H_1 and H_2 do.

Theorem 3.12. *For a positive integer h, System* (3.4) *is box-TDI if and only if G is series-parallel.*

Surprisingly, though series-parallel graphs are usually easy to handle, the proof of Theorem 3.12 came out to be very technical. It is based on the following new structural property of series-parallel graphs.

Proposition 3.2. *For a simple nontrivial 2-connected series-parallel graph, at least one of the following holds:*

(a) two nodes of degree 2 are adjacent,

(b) a node of degree 2 belongs to a circuit of length 3,

(c) two nodes of degree 2 belong to a same circuit of length 4.

We prove Theorem 3.12 by considering a minimal counter example, that is, a series-parallel graph, and a vector $c \in \mathbb{Z}^E$ such that:

1. the dual of $\min\{c^\top x \mid x \text{ satisfies } (3.4)\}$ is finite but has no integer optimal solution,
2. G has a minimum number of edges,
3. $\sum_{e \in E} c_E$ is minimum with respect to 2.

We then prove that no condition of Proposition 3.2 holds in G, implying that G is not series-parallel. By Theorem 3.10, this proves Theorem 3.12.

Part II

Mixed-Integer Linear Programming and Machine Learning

Chapter 4

Preliminaries

This chapter is devoted to the presentation of basic notions and definitions that are used in the subsequent chapters. The first section is devoted to Lagrangian relaxation whereas the second one focuses on machine learning.

4.1 Preliminaries on Lagrangian Relaxation

Lagrangian relaxation is a widely used method to obtain dual bounds. We focus here on Lagrangian relaxation for Mixed-Integer Linear Problems (MILP), even if it can applied more generally to convex optimization problems. For more details to Lagrangian relaxation, the interested reader may refer to (Lemaréchal, 2001).

4.1.1 Lagrangian Relaxed Problem and Lagrangian Dual Problem

Consider the following mixed-integer linear problem:

$$(P)\begin{cases} \max c^\top x & (4.1a)\\ Ax \leq b & (4.1b)\\ x \in \mathcal{S}, & (4.1c)\end{cases}$$

where $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, and $\mathcal{S} = \{x \in \mathbb{Z}^n \times \mathbb{R}^p \mid Cx \leq d\}$ is nonempty. Solving (P) is NP-hard in general (Karp, 1972), and providing dual bounds (*i.e.*, upper bounds) helps to prune the solution space. Suppose that removing (4.1b) yields a tractable problem. In this case, one can apply Lagrangian relaxation to obtain an upper bound. This method consists in removing inequalities (4.1b) but penalizing their violation. The trade-off between optimizing the original objective function and maximizing the satisfaction of the dualized constraints is done using Lagrangian multipliers which indicate the importance of each dualized constraint with respect to the objective function.

More formally, for $\pi \in \mathbb{R}_+^m$, the *Lagrangian Relaxed problem* of (P) is:

$$(LR(\pi)) \begin{cases} \max c^\top x + \pi^\top (b - Ax) & (4.2a) \\ x \in \mathcal{S}. & (4.2b) \end{cases}$$

Since every solution to (P) satisfies $Ax \leq b$ and π is nonnegative, its cost in $(LR(\pi))$ is greater than or equal to its cost in (P). Moreover, the solution set of (P) is contained in the one of $(LR(\pi))$ so the latter is a relaxation of the former, that is, $(LR(\pi)) \geq (P)$ for all $\pi \in \mathbb{R}_+^m$. The best Lagrangian dual bound is given by solving the *Lagrangian Dual problem (LD)*:

$$(LD) \quad \min_{\pi \in \mathbb{R}_+^m} LR(\pi). \tag{4.3}$$

The quality of the bound (LD) depends on the constraints that are dualized. There is a trade-off between the quality of this dual bound and the complexity to solve (LD) which depends on the tractability of the Lagrangian relaxed problem. The following theorem, stemming from linear duality, gives a clue on the quality of the bound.

Theorem 4.1 (Geoffrion, 1974)**.** *If the linear problem:*

$$\begin{cases} \max c^\top x & (4.4a) \\ Ax \leq b & (4.4b) \\ x \in conv(\mathcal{S}) & (4.4c) \end{cases}$$

is feasible, then its optimum is equal to the optimum of (LD).

From Theorem 4.1, it follows that (LD) gives a bound that is as tight as the one given by the continuous relaxation. A necessary condition to obtain a better bound than the continuous relaxation is to dualize constraints such that $\mathcal{S}$ is not an integer polyhedron.

4.1.2 Solving the Lagrangian Dual Problem

The cost of each solution $\bar{x} \in \mathcal{S}$ of $(LR(\pi))$ is a linear function of the Lagrangian multiplier π. Then, the *dual function* $LR : \mathbb{R}_+^m \to \mathbb{R}$ is a convex piecewise linear function function defined by $\pi \mapsto LR(\pi)$. It is continuous and convex, but not differentiable on $\mathbb{R}_+^m$ in general. Figure 4.1[1] gives an example of such a function where $\pi \in \mathbb{R}_+$ corresponds to the case where only one constraint has been dualized.

There exist different algorithms to solve the Lagrangian dual problem. We briefly present the algorithms used in the subsequent works. For more details on these methods, one could refer for instance to (Lemaréchal, 2001) or (Bertsekas, 2016, Chapter 7).

[1]This figure and the subsequent ones in this preliminary chapter are to be credited to Francesca Demelas.

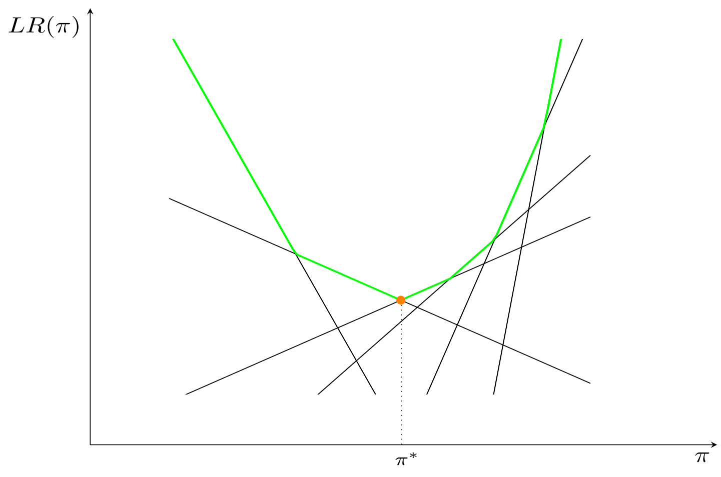

Figure 4.1: Example of a dual function drawn in green and obtained by dualizing only one inequality, that is, $\pi \in \mathbb{R}_+$. Each line corresponds to the cost of a solution to $\mathcal{S}$ with respect to the Lagrangian multiplier π. The value π^* is the one minimizing $LR(\pi)$; the ordinate of the orange point corresponds to the optimum of (LD).

Subgradient based methods

To find its minimum, that is, to solve (LD), one may apply a subgradient method.

A vector $g \in \mathbb{R}^m$ is a *subgradient* of a function $f : X \subseteq \mathbb{R}^m \to \mathbb{R}$ at point $x_0 \in X$ if

$$f(x) \geq f(x_0) + g^\top (x - x_0) \quad \forall x \in X. \tag{4.5}$$

For each $\pi \in \mathbb{R}^m_+$, solving the Lagrangian relaxed problem $LR(\pi)$ gives an optimal solution x^π and a subgradient of LR at point x^π equal to $b - Ax^\pi$.

Starting from an initial point π^0, one may iteratively converge to the minimum of (LD) with the series:

$$\pi^{k+1} = \left[\pi^k - \alpha^k (b - Ax^{\pi^k})\right]^+,$$

where π^k is the Lagrangian multiplier at iteration k, x^{π^k} an optimal solution to $LR(\pi^k)$, $\alpha^k > 0$ is the step-size, and $[x]^+ = \max\{\mathbf{0}, x\}$. Poljak (1987, Chapter 5) shows that convergence holds when the stepsize satisfies the following conditions:

$$\alpha^k > 0, \lim_{k\to\infty} \alpha^k = 0, \sum_{k=1}^{\infty} \alpha^k = +\infty. \tag{4.6}$$

However, stepsizes satisfying those conditions may lead to slow convergence in practice. An active research area consists in finding stepsizes and in adapting the subgradient method to accelerate its convergence (Bragin, 2024).

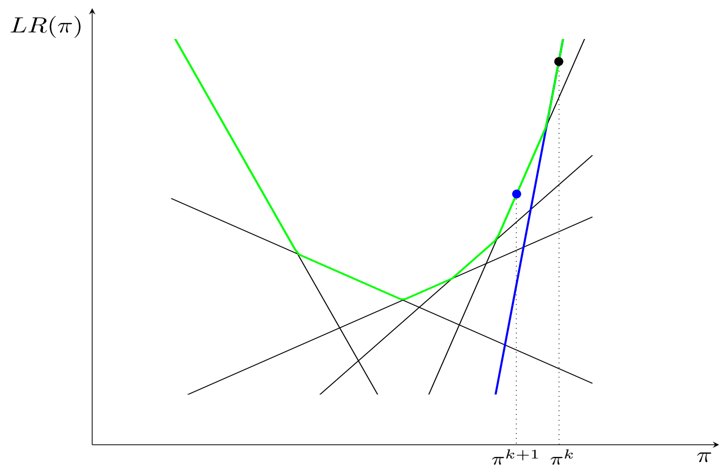

Figure 4.2: Example of an iteration of the subgradient method. The dual function LR is the one of Figure 4.1. The point in black corresponds to $LR(\pi^k)$. The line in blue is $(g^k)^\top(\pi - \pi^k) + LR(\pi^k)$, where g^k is the subgradient at point π^k. The update of the point π^k with respect to subgradient g^k gives the point π^{k+1}.

Figure 4.2 gives an example of an iteration of the subgradient method.

Handling an exponential number of dualized constraints Subgradient methods may be used to solve a Lagrangian relaxation with an exponential number of dualized constraints. In this case, only a small subset of these constraints is considered at the beginning. Then, each time the Lagrangian relaxed problem is solved, a separation phase is performed to detect inequalities violated by the optimal solution to the current Lagrangian relaxed problem, similarly to what is done in cutting-plane algorithms (Dantzig, Fulkerson, and Johnson, 1954; Marchand et al., 2002). These inequalities are added on the fly to the set of dualized inequalities before updating the Lagrangian multipliers. Such a method is called *non delayed Relax-and-Cut algorithm* (Lucena, 2005). There exist variants where violated constraints are not added at each iteration of the subgradient method but consist in solving successive Lagrangian dual problems obtained by adding violated constraints (Lucena, 2006).

Cutting-plane methods

By definition, (LD) can be rewritten as minimizing a value that is greater or equal to the cost of each solution to $\mathcal{S}$ parametrized by Lagrangian multipliers,

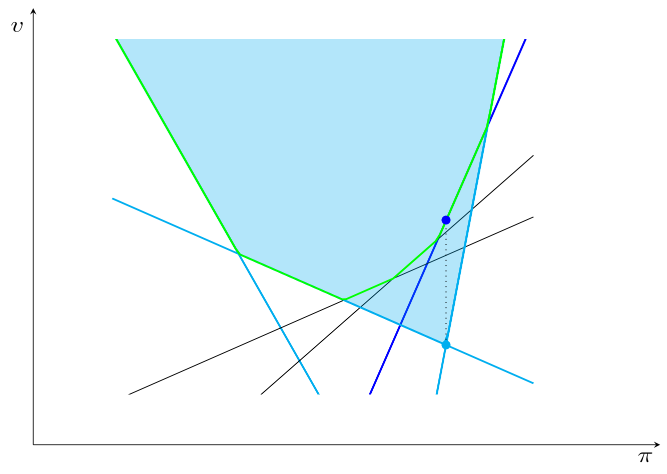

Figure 4.3: Example of an iteration of the cutting-plane algorithm. The dual function LR is the one of Figure 4.1 and is drawn in green. The set of solutions to the current restricted master problem $RMP(\mathcal{S}')$ is drawn in cyan and is defined by three inequalities (4.7b). The resolution of $RMP(\mathcal{S}')$ gives the point $(\bar{\pi}, \bar{v})$ in cyan. The separation problem gives the blue point $(\bar{\pi}, LR(\bar{\pi}))$ and exhibits in blue an inequality (4.7b) violated by the cyan point $(\bar{\pi}, \bar{v})$. This inequality is added to $RMP(\mathcal{S}')$.

that is:

$$(LD)\begin{cases} \min v & (4.7a)\\ v \geq c^\top x + \pi^\top (b - Ax) \quad \forall x \in \mathcal{S}, & (4.7b)\\ \pi \geq \mathbf{0}. & (4.7c)\end{cases}$$

In general, System (4.7) contains an exponential number of inequalities (4.7b), even if this set may be restricted to inequalities associated with extreme points of $conv(\mathcal{S})$. Hence, System (4.7) may be solved by a cutting-plane algorithm (Cheney and Goldstein, 1959; Kelley, 1960) which can be described as follows.

For a subset $\mathcal{S}' \subseteq \mathcal{S}$, let $RMP(\mathcal{S}')$ be the system obtained from System (4.7) by considering only the inequalities (4.7b) associated with $x \in \mathcal{S}'$. Solving $RMP(\mathcal{S}')$ gives a solution $(\bar{\pi}, \bar{v})$. If $(\bar{\pi}, \bar{v})$ violates the inequality (4.7b) associated with some point $x \in \mathcal{S}$, this point is added to $\mathcal{S}'$ and the algorithm iterates. Otherwise, $(\bar{\pi}, \bar{v})$ corresponds to an optimal solution to System (4.7).

The *separation problem*, that is the problem of determining whether there exists an inequality (4.7b) violated by $(\bar{\pi}, \bar{v})$ is solved by computing $LR(\bar{\pi})$. If its optimum is equal to $\bar{v}$, no inequality is violated. Otherwise, $(\bar{\pi}, \bar{v})$ violates the inequality associated with any optimal solution to $LR(\bar{\pi})$.

Figure 4.3 gives an example of an iteration of the cutting-plane algorithm.

Bundle methods

The presented cutting-plane algorithm is known to require a huge number of iterations to converge (Lemaréchal, 2001). Several adaptions of this algorithm have been devised in order to stabilize it. One if these is the *bundle method.* It consists in considering at each iteration a *stabilization point* π^s, and optimizing the current restricted master problem while not being too far from this stabilization point. This is done by adding a quadratic term in the objective function corresponding to the square of the Euclidean distance with respect to this stabilization point. For $\mathcal{S}' \subseteq \mathcal{S}$, $RMP(\mathcal{S}')$ becomes:

$$\begin{cases} \min v + \dfrac{1}{2\eta}\|\pi - \pi^s\|^2 & \text{(4.8a)} \\ v \geq c^\top x + \pi^\top(b - Ax) \quad \forall x \in \mathcal{S}', & \text{(4.8b)} \\ \pi \geq \mathbf{0}, & \text{(4.8c)} \end{cases}$$

where η is the *regularization parameter* used to balance the weight of the norm with respect to the original objective function.

The restricted master problem is not a linear problem anymore but a quadratic one. However, the objective is convex and the constraints remain linear, so it can be solved efficiently (Kiwiel, 1986; Kiwiel, 1989; Frangioni, 1996). Moreover, the additional runtime required to solve this problem in each iteration is offset by the reduction in the number of iterations for the entire algorithm.

Figure 4.4 presents an example of an iteration of the bundle method. Due to the stabilization term in the objective function of $RMP(\mathcal{S}')$, the optimal solution differs from the one of the cutting-plane algorithm given in Figure 4.3, and so does the inequality added to $RMP(\mathcal{S}')$.

4.2 Preliminaries on Machine Learning

In this section, we give a brief overview of machine learning concepts and models used in the subsequent chapters. This presentation is restricted to machine learning based on neural networks. This approach is presented in Section 4.2.1 whereas Section 4.2.2 is dedicated to the presentation of different neural network architectures. For a more detailled review of these concepts, one can refer to (Murphy, 2013; Prince, 2023).

4.2.1 Learning with Neural Networks

A machine learning *task* consists in approximating an unknown mapping from each input of that task to its associated output. The notion of task is quite general and covers several problematics and domain areas like for instance image classification, syntactic parsing in natural language processing, or solution prediction in mathematical programming. The set of inputs of the task is denoted

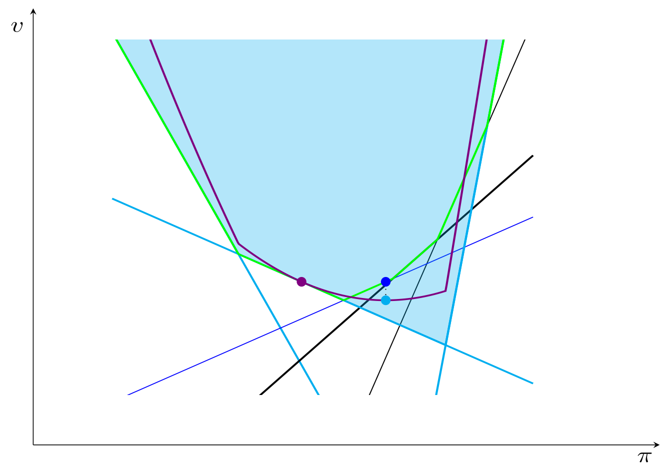

Figure 4.4: Example of an iteration of the bundle algorithm. The dual function LR is the one of Figure 4.1 and is drawn in green. The set of solutions to the current restricted master problem $RMP(\mathcal{S}')$ is drawn in cyan and is defined by three inequalities (4.8b). The point in purple is the stabilization point at the current iteration. The curve in purple represents the value of the objective function (4.8a) with respect to the Lagrangian multiplier π. The minimum is obtained at the cyan point. The separation problem gives the blue point $(\bar{\pi}, LR(\bar{\pi}))$ and exhibits in blue an inequality (4.8b) violated by the cyan point $(\bar{\pi}, \bar{v})$. This inequality is added to $RMP(\mathcal{S}')$.

by X and the set of outputs by Y. Each element $x \in X$ is associated with an output $y \in Y$.

Given a task, machine learning is to learn from data the parameters of a neural network, that is an approximation of the unknown mapping from each input of the task to its associated output. Mathematically speaking, a *neural network* is a mathematical function:

$$f_\theta : X \to Y, \tag{4.9}$$

where θ are its *parameters*. Different architectures exist that correspond to different shapes of f_θ, see Section 4.2.2 for details.

Training a neural network consists in determining the parameters θ such that $f_\theta(x)$ tends to be the closest to the output y of x for every input $x \in X$. Since X is usually infinite, we consider a dataset $\mathcal{D}$ of samples of X for training. For each sample x of $\mathcal{D}$, the dissimilarity between $f_\theta(x)$ and the output y of x is measured using a loss function. This latter and the neural network define a *machine learning model.*

Training a machine learning model is then cast as an optimization problem consisting in determining the parameters that minimize the averaged loss function over the dataset $\mathcal{D}$. If $\mathcal{D}$ correctly represents the input set X, one should expect the model to correctly handle unkown data.

The model depends on the machine learning task and on the data available for training the model. We now present different machine learning paradigms: supervised learning, structured learning and unsupervised learning.

Supervised learning

In supervised learning, the dataset $\mathcal{D}$ is composed of n samples of X and their associated outputs, that is, $\mathcal{D} = \{(x^i, y^i) \mid i = 1, \ldots, n\}$. The outputs y^i for $i = 1, \ldots, n$ are referred to *labels*, and the *loss function* measures the dissimilarity between the prediction $f_\theta(x^i)$ and the label y^i. It is defined as:

$$L(y, y') : Y \times Y \to \mathbb{R}_+. \tag{4.10}$$

The loss function must satisfy $L(y, y) = 0$ and the more similar y and y' are, the smaller should be $L(y, y')$. Then, *training* a model is nothing but determining:

$$\theta^* \in \operatorname*{argmin}_\theta \sum_{i=1}^{n} L(f_\theta(x^i), y^i). \tag{4.11}$$

This problem is untractable as it consists in minimizing a high-dimensional non linear function. Hence, the usual way to tackle this problem is by gradient descent based methods to find a local optimum. For this, the neural network f_θ parametrized by θ and the loss L are designed in such a way that the gradient $\frac{\partial L}{\partial \theta}(f_\theta(x^i), y^i)$ with respect to parameters θ can be efficiently computed for all $i = 1, \ldots, n$. This allows to train the model using a stochastic gradient descent. The dataset is randomly partitionned into *batches*. For each batch B, the gradient

$$\sum_{(x^i,y^i)\in B} \frac{\partial L}{\partial \theta}(f_\theta(x^i), y^i)$$

is computed, and the parameters θ are updated with respect to that gradient, usually using the ADAM rule (Kingma and Ba, 2014).

Note that if there is a single batch containing the whole dataset, the stochastic gradient descent is actually a gradient descent. Experimentally, stochastic gradient descent is usually faster than gradient descent since each parameter update requires less time (Bottou and Bousquet, 2007; Bottou and Bousquet, 2011). Moreover, the randomness introduced by batches allows to escape from local optima, see (Prince, 2023, Section 6.2) for more details.

Generalization and regularization The objective of machine learning is to obtain after training models with a high *generalization capacity*, that is, a model which is able to make accurate predictions on unseen data. To measure this, the dataset is divided into a *train set* used for training the model and a *test set* which corresponds to the unseen instances on which the generalization capacity of the model is measured[2].

In general, the prediction accuracy is worse on the test set than on the train test. This is due, among other things, to the *noise* in the dataset, that is, there are multiple possible outputs for a same input. This may be due for instance to the nature of the learning task itself, to some stochasticity process in the data generation or to some errors in the labels.

Due to the noise in the train set, a mapping of each input of the train set to its corresponding output does not necessarly correspond to the mapping we want to learn. Hence, a model really accurate on the train set may not generalize well. This phenomenon is known as *overfitting*.

To prevent overfitting, that is, to reduce the gap between the accuracy on training and test sets, one may consider *regularization*. This consists in adding in the loss function a term to smooth the neural network. This regularization may be for instance the Eclidean norm of the parameters that has to be minimized in addition to the loss. The objective to minimize when training a model becomes:

$$\sum_{i=1}^{n} L(f_\theta(x^i), y^i) + \lambda\|\theta\|^2,$$

where λ is an hyperparameter used as a ponderation between the regularization term and the original loss.

For a clear understanding of generalization issues and regularization, we refer to (Prince, 2023, Chapters 8 and 9).

[2]One usually also have a *validation set* used to evaluate the generalization capacity of different models or variants (usually defined by different values of hyperparameters like the number of neurons or layers in the architecture) and to select the model with the highest generalization capacity before evaluating it on the test set.

Structured learning

Structured learning is when the output domain Y of a machine learning task is constrained and may be represented as a set of integer points. In this case, the learning framework changes: the prediction y is not directly given by the output $f_\theta(x)$ of the neural network but this latter is used to parametrize the problem of determining the best element of Y with respect to the input x. For instance, in syntactic parsing, machine learning is used to predict the syntactic tree of a sentence. The prediction must follow the structure of a syntactic tree. The output domain is represented as incidence vectors of constrained spanning arborescences in a digraph. A neural network is used to score the arcs of the digraph, and the prediction is the spanning arborescence maximizing the score.

The problem of finding the best element given a network output is called *inference problem*, and this problem may be more or less complex depending on the machine learning task. In structured learning, a model is then defined by a neural network, a loss and an inference model.

Energy based models Structured learning can be tackled using *Energy Based Model (EBM)* framework. In EBMs, a value, referred to as *energy*, is associated with each couple $(x, y) \in X \times Y$, and the smaller the energy is, the more likely is that y is the output of x in the learning task. This energy is defined using a neural network and is then parametrized by θ. It is denoted by $E(x, y, \theta)$. For a given $x \in X$, the inference problem consists in finding the element $y^* \in Y$ minimizing the energy, that is, $y^* = \min_{y \in Y} E(x, y, \theta)$.

In supervised learning, training an EBM consists in determining θ^* that minimizes a loss such that after training, for each $i = 1, \ldots, n$, the energy $E(x^i, y^i, \theta^*)$ is lower than $E(x^i, y, \theta^*)$ for all $y \neq y^i$. In that case, solving the inference problem associated with x^i and θ^* gives the correct output y^i. The interested reader can refer to (Le Cun et al., 2006) for a thorough description of EBMs.

Different loss functions have been defined in the litterature. One of the most famous is the *perceptron loss* defined by:

$$L(x^i, y^i) = E(x^i, y^i, \theta) - \min_{y \in Y} E(x^i, y, \theta).$$

The name perceptron loss stems from the structured perceptron (Collins, 2002) considered as one of the first EBM and which can be described as follows. Consider a learning task where $Y \subseteq \{0, 1\}^d$ and $X \in \mathbb{R}^{f \times d}$, that is each $x \in X$ is a matrix whose i^{th} column is a vector of dimension f representing component $i \in \{1, \ldots, d\}$. The *structured perceptron* is an EBM parametrized by a vector $\theta \in \mathbb{R}^f$, its energy is defined as $E(x, y, \theta) = (\theta^\top x)y$, and the loss function is the perceptron loss.

Unsupervised learning and latent variables

Up to now, we have considered supervised learning where the dataset contains for each input its associated label, and the loss function measures the dissimilarity between the prediction from the input and the associated label. In unsupervised learning, there is no label in the dataset.

Unsupervised learning is usually based on a *latent representation* of the input. The latent space, denoted by Z, is much smaller than the input domain X, and the latent representation z of an input x is intended to capture the important underlying properties of x while being easier to handle. In some way, z can be considered as a "compressed version" of x.

Machine learning with latent variables is based on the *encoder-decoder* framework (Sutskever, Vinyals, and Le, 2014; Cho et al., 2014; Kalchbrenner and Blunsom, 2013). The *encoder* is a neural network that maps the input $x \in X$ to its latent representation $z \in Z$. This latent representation z is then used as input for the *decoder* which is a neural network whose design depends on the learning task. Figure 4.5 gives an illustration of an encoder-decoder.

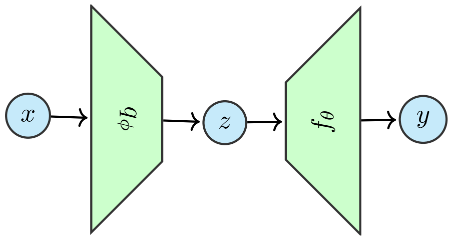

Figure 4.5: Visualization of an encoder-decoder. The encoder q_ϕ is a neural network parametrized by ϕ that takes some $x \in X$ as input and outputs its latent representation z. The decoder f_θ is a neural network parametrized by θ using the latent representation z for input and outputing the overall prediction $y \in Y$.

A *generative model* is a model that may generate different outputs for a same input. When using an encoder-decoder, a way to do it is to use a *probabilistic encoder* that outputs parameters of a probabilistic distribution over the latent space like the mean and variance of a Gaussian distribution for instance. Then, several samples are drawn from the latent space using this distribution, and each sample is used as input for the decoder producing as many outputs as drawn samples.

A well-known type of probabilistic encoder-decoder is *variational autoencoder* (Kingma and Welling, 2014; Pinheiro Cinelli et al., 2021) that is designed to learn a latent representation of the input. In that case, the output domain is X, and the model is trained to be able to retrieve the input from the latent representation.

4.2.2 Neural Network Architectures

Deep neural networks are built by stacking different "simple" neural networks called *layers*. In a deep neural network with ℓ layers, the output h^k of layer number k is the input of layer number $k+1$ for $k = 1, \dots, \ell - 1$. The input h^0 of the first layer is the machine learning task x and the output h^ℓ of the final layer corresponds to y or is used to parametrize the inference problem giving y. Figure 4.6 provides a visualization of layers constituing a deep neural network.

$$\begin{tikzcd}
h^0 \arrow[r, dash] & \boxed{\text{Layer } 1} \arrow[r, dash, "h^1"] & \boxed{\text{Layer } 2} \arrow[r, "h^2"] & \cdots \arrow[r, "h^{\ell-1}"] & \boxed{\text{Layer } \ell} \arrow[r] & h^\ell
\end{tikzcd}$$

Figure 4.6: Stacking layers to build deep neural networks.

From a mathematical point of view, stacking layers is composing functions. For $k = 1, \dots, \ell$, let $f^k_{\theta^k}$ denote the function parametrized by θ^k that corresponds to layer k. The deep neural network obtained by stacking the ℓ layers is equivalent to:

$$f_\theta(x) = f^\ell_{\theta^\ell} \circ f^{\ell-1}_{\theta^{\ell-1}} \circ \cdots \circ f^1_{\theta^1}(x)$$

and θ is the concatenation of parameters $\theta^1, \dots, \theta^\ell$.

Note that as already mentionned, a crucial point to train a neural network using gradient descent is to be able to efficiently compute the value $\frac{\partial f}{\partial \theta}$. Using the chain rule, one can compute it from the values $\frac{\partial f^k}{\partial \theta^k}$ and $\frac{\partial f^k}{\partial h^{k-1}}$. This calculus can be done efficiently by dynamic programming (by reusing partial computations) starting from the last layer and iteratively computing the values for a layer k using the already computed values for layer $k+1$. This algorithm is referred to as *backpropagation* (Rumelhart, Hinton, and Williams, 1986; Rumelhart and McClelland, 1987) and is implemented in modern machine learning frameworks such as Pytorch (Paszke et al., 2019) or Flux (Innes et al., 2018; Innes, 2018). The implementation is done to be efficiently parallelized on GPU cores.

In the rest of this section, we review the different layers used to build deep neural networks in the subsequent works. For a layer, we denote by x its input, y its output and θ its parameters. When this layer is used as the k^{th} layer of a deep neural network, x, y and θ correspond to h^{k-1}, h^k and θ^k, respectively. Depending on the layer, x and y may be vectors or matrices. The function $\sigma : \mathbb{R} \to \mathbb{R}$ will denote a non linear function, usually called *activation function*. This function can be a rectified linear unit (ReLU) function (He, Zhang, et al., 2015), the softplus (Zheng et al., 2015), the hyperbolic tangent, and the sigmoid function. When z is a vector or a matrix, $\sigma(z)$ consists in applying the activation function over each component of z.

This review only covers a small part of the layers used in deep learning in general. We refer to (Zhang, Lipton, et al., 2024; Prince, 2023) for more details on the different layers.

Dense layers and feed forward neural networks

A *dense layer* takes as input a vector $x \in \mathbb{R}^{d_x}$ and consists in an affine map followed by a nonlinear function on every component of the resulting vector, that is:

$$y = \sigma(Wx + b). \tag{4.12}$$

The parameter θ of a dense layer is given by W and the *bias* b.

A neural network composed of dense layers is called a *Feed Forward Neural Network (FFNN)*. They are also referred to *multilayer perceptron* since a dense layer where the output $y \in$ is a scalar was one of the first machine learning model called *perceptron* (Rosenblatt, 1958).

Recurrent neural layers

Recurrent neural layers are designed to predict sequences. Given an input $x = (x_1, \dots, x_n)$, one may have to predict an output $y = (y_1, ..., y_n)$ where the prediction y_t for some step $t = 1, \dots, n$ depends on x_t but also on the past, that is, on the previous inputs $x_{t'}$ and predictions $y_{t'}$ for $t' = 1, \dots, t-1$. This happens for instance in Part-of-Speech tagging (POS), a NLP task where given a sentence, one has to tag each of its words with a part-of-speech tag like "noun", "verb" or "determiner".

Recurrent neural layers may also be used to handle inputs of varying length. Indeed, feed forward neural networks need inputs of the same length, and artificially getting inputs of equal length by filling missing values does not perform well in general. It may be better to use a recurrent neural layer to process sequentially each component and using for output the one of the last element.

In the following, we restrict our description to long short-term memory layers which correspond to one of the most popular recurrent neural layers.

Long Short-Term Memory *Long Short-Term Memory (LSTM)* (Hochreiter and Schmidhuber, 1997) is a special kind of recurrent neural layer designed to avoid the exploiding or the vanishing of the gradient during backpropagation due to long sequences of multiplications.

A vector $s_t \in \mathbb{R}^{d_y}$ is used to represent the state of the LSTM. This vector is propagated at each step t of the prediction. Parameter θ of an LSTM is composed of four matrices θ^f, θ^i, θ^o and θ^a of $\mathbb{R}^{d_y \times 2d_x}$ and of four vectors b^f, b^i, b^o and b^a of $\mathbb{R}^{d_y}$.

For each step $t = 1, \dots, n$, four vectors are computed using θ, x_t and y_{t-1}. These four vectors f_t, o_t, i_t and a_t are obtained by applying a dense layer[3] on the vector $(x_t \parallel y_{t-1})$ which corresponds to the concatenation of x_t and y_{t-1}[4]:

$$f_t = \sigma(\theta^f(x_t \parallel y_{t-1}) + b_f), \tag{4.13}$$

$$i_t = \sigma(\theta^i(x_t \parallel y_{t-1}) + b_i), \tag{4.14}$$

$$o_t = \sigma(\theta^o(x_t \parallel y_{t-1}) + b_o), \tag{4.15}$$

$$a_t = \sigma(\theta^a(x_t \parallel y_{t-1}) + b_a). \tag{4.16}$$

$$\tag{4.17}$$

[3]Usually, the activation function used in (4.16) is the hyperbolic tangent whereas the other activation functions are sigmoids.

[4]y_0 is initialized with $y_0 = \mathbf{0}$.

Vector f_t, referred to *forget gate*, is used to "remove information from the past state" whereas vectors i_t and a_t, referred to as *input gate* and *cell activation*, are used to "add new information in the state". Mathematically, these operations are given by:

$$s_t = f_t \odot s_{t-1} + i_t \odot a_t, \tag{4.18}$$

$$\tag{4.19}$$

where $\odot$ denotes the Hadamart product, that is, the element-wise product. Finally, the output y_t is computed from the current state s_t using the *output gate* o_t as follows:

$$y_t = o_t \odot \sigma(s_t). \tag{4.20}$$

Bidirectional LSTM In a LSTM, the prediction y_t is made using information from the past thanks to both s_t and y_{t-1}. However, no information from the future, that is no input $x_{t'}$, state $s_{t'}$ or output $y_{t'}$ associated with some $t' > t$ is used. When this information from the future matters, one can use *Bidirectional LSTM (BiLSTM)*. This layer consists in two disjoint LSTM, say LSTM$^{\mathrm{F}}$ and LSTM$^{\mathrm{R}}$, such that LSTM$^{\mathrm{F}}$ processes the sequence $(x_1, \ldots, x_n)$ from the beginning to the end (as presented above), whereas LSTM$^{\mathrm{R}}$ processes the sequence in reverse sense, that is, from the end to the beginning. The output y_t^F and y_t^R at step $t = 1, \ldots, n$ of both LSTM are concatenated to form the output y_t of the BiLSTM.

Graph convolutional layers and networks

Sometimes, the input of a learning task is or may be represented by a graph. This is the case for instance when the task consists in detecting fraud in financial networks (Van Belle et al., 2022; Motie and Raahemi, 2024) or in predicting the citation count of scientific papers in academic networks (Dong, Johnson, and Chawla, 2015; He, Xue, et al., 2023). Graph neural networks are designed to handle such inputs (Zhou, Cui, et al., 2020). Different types of graph neural networks exist (Zhou, Zheng, et al., 2022; Corso et al., 2024) but we restrict our presentation to the graph convolutional ones (Kipf and Welling, 2017).

The input graph is used inside a graph convolutional layer to compute the output of a node using its input and those of its neighbors. Mathematically speaking, a *Graph Convolutional layer* is defined as follows.

Let $G = (V, E)$ be an undirected graph. The *adjacency matrix* $A \in \{0,1\}^{|V| \times |V|}$ of G has entry A_{ij} equal to 1 if nodes i and j are adjacent, and 0 otherwise. Let $\tilde{A} = A + I_{|V|}$ where $I_{|V|}$ is the identity matrix of size $|V| \times |V|$, and define the diagonal matrix $\tilde{D}$ such that $\tilde{D}_{ii} = \sum_{j=1}^{|V|} \tilde{A}_{ij}$ for $i \in V$.

The input x of this layer is a matrix of $|V|$ rows and d_x columns. The v^{th} row of x corresponds to the vector input of node $v \in V$. $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} x$ gives a linear map where the v^{th} row is computed from the rows of x associated with v

and its neighbors[5]. The matrix is then multiplied by the parameter matrix θ of the layer and a nonlinear function is applied to each element in order to obtain a matrix $y \in \mathbb{R}^{|V| \times d_y}$ where the v^{th} row corresponds to the output of node v. We have:

$$y = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} x \theta), \tag{4.21}$$

where the parameter θ is a matrix of $\mathbb{R}^{d_x \times d_y}$.

Graph convolutional networks consist in stacking several graph convolutional layers. This allows to propagate information through the graph as the output of a node v in a graph convolutional network with ℓ layers is computed from the input of all the nodes at distance no more than ℓ from v.

Biaffine attention layers

Attention (Bahdanau, Cho, and Bengio, 2014; Kim et al., 2017) is a core mecanism of modern deep learning architectures. It mainly offers a way to focus on some parts of the inputs by learning a ponderation when aggregating data. It is used in several neural networks such as transformers (Vaswani et al., 2017) or graph attention networks (Veličković et al., 2018).

The input is composed of a set of couples *key/value* $(k_i, v_i) \in \mathbb{R}^{d_k} \times \mathbb{R}^{d_v}$ for $i = 1, \ldots, m$, as well as a set of *queries* $q_j \in \mathbb{R}^{d_q}$ for $j = 1, \ldots, n$. The output is a representation $y_1, \ldots, y_n$ by vectors of $\mathbb{R}^{d_y}$. Each representation y_j is computed from the query q_j and from k_i, v_i for all $i = 1, \ldots, m$.

Let $a_\theta(k, q) : \mathbb{R}^{d_k} \times \mathbb{R}^{d_q} \rightarrow \mathbb{R}$ be a learnable function parametrized by θ. Given a query, this function is used to gather information from the values by computing for each key the proportion of the associated value used in the representation. More formally, for $j = 1, \ldots, n$, the output y_j is computed as follows:

$$y_j = \sum_{i=1}^{m} \alpha(k_i, q_j) v_i, \tag{4.22}$$

where

$$\alpha(k_i, q_j) = \frac{exp(a_\theta(k_i, q_j))}{\sum_{i'=1}^{m} exp(a_\theta(k_{i'}, q_j))}. \tag{4.23}$$

Note that $(\alpha(k_1, q_j), ..., \alpha(k_m, q_j))$ is obtained by a softmax transformation and satisfies $\alpha(k_i, q_j) \geq 0$ for $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \alpha(k_i, q_j) = 1$. Hence, y_j is a convex combination of $v_1, \ldots, v_m$ where the coefficients of this convex combination are given by the learnable function a_θ.

Different functions a_θ may be used for attention mecanism. In a *biaffine attention layer*, a_θ is of the form:

$$a_\theta(k, q) = k^\top W q + b^\top k, \tag{4.24}$$

and θ corresponds to $W \in \mathbb{R}^{d_k \times d_q}$ and $b \in \mathbb{R}^{d_k}$.

[5]The matrix $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is a renormalization of $\tilde{A}$ aimed at balancing the influence across all nodes irrespective to their degree.

Chapter 5

Using Mixed-Integer Linear Programming for Inference in Machine Learning

This chapter is dedicated to my research relative to the use of mixed-integer linear programming and Lagrangian relaxation to solve inference problems in machine learning relative to syntactic parsing.

Syntactic parsing is a fundamental task in Natural Language Processing (NLP) that reveals the syntactic structure of a sentence. It can be used for instance for grammar checking, or as a component in more complex NLP tasks (Drozdov et al., 2023; Yin et al., 2022). Syntactic parsing is cast as finding a discrete structure maximizing a score function or a likelihood, the score being predicted via a neural network. The type of the discrete structure and the requirements it must satisfy depend on the considered grammatical formalism.

In section 5.1, we first give a brief description of dependency trees, and present different requirements that a dependency tree may follow. We then consider the dependency parsing problem where the dependency tree must be well-nested and satisfy block degree requirements. We formulate this problem as an integer linear problem and present a Lagrangian relaxation based heuristic to solve it. This approach comes from a joint work with C. Corro, J. Le Roux, A. Rozenknop and R. Wolfer Calvo (Corro, Le Roux, Lacroix, et al., 2016).

In Section 5.2, we consider constituency parsing using spinal TAG. We formulate this parsing problem as a generalized spanning arborescence problem. We also propose a Lagrangian Decomposition based heuristic. These results stem from a joint work with C. Corro and J. Le Roux (Corro, Le Roux, and Lacroix, 2017).

Note that this chapter is not intended to be an in-depth presentation of syntactic parsing. Only the NLP material necessary to present the aforementioned works is given. For a thorough presentation of parsing or NLP in general, one may refer for instance to (Jurafsky and Martin, 2025).

5.1 Dependency Parsing with Bounded Block Degree and Well-nestedness

We present dependency trees and its variants in Section 5.1.1. Section 5.1.2 is dedicated to the presentation of a parsing algorithm based on Lagrangian relaxation.

5.1.1 Dependency Trees

In dependency grammars, the syntactic structure of a sentence is described as a set of *syntactic relations* between couples of words. Each relation emphasizes a *head* word with a *dependent* word that modifies the head. This relation is labeled with the grammatical function of the dependent with respect to the head. For instance in "the algorithm", the word "algorithm" is the head word whereas "the" is the dependent one. The relation is labeled with "determiner".

Syntactic relations of a sentence are usually represented using graphs. Given a sentence $w = w_0, w_1, \dots, w_n$, where w_0 is a dummy root word, we associate a node with each word w_i[1], and denote by $V = \{w_0, \dots, w_n\}$ this set of nodes. A syntactic relation with head w_i and dependent w_j is represented with an arc (w_i, w_j)[2]. Each arc is labeled with the associated grammatical function.

With this graph based representation, a (syntactic) *dependency tree (DT)* of a sentence is defined as a set of syntactic relations such that their representation by labeled arcs forms a spanning arborescence rooted at w_0, that is, T is a set of arcs such that:

- w_0 has no entering arc,
- w_i has one entering arc for each $i = 1, \dots, n$,
- the graph (V, T) contains no circuit.

In the following, a dependency tree will also refer to the associated spanning arborescence rooted at w_0. Figure 5.1 gives an example of a dependency tree.

Projectivity Some grammar formalisms consider that the dependency tree must satisfy additional requirements. The most known requirement is *projectivity* (Nivre, 2003). It consists in considering only dependency trees having no crossing pairs of arcs, that is, the underlying spanning tree obtained by removing the orientation of the arcs does not contain edges $w_i w_k$ and $w_j w_\ell$ with $i < j < k < \ell$.

Projective trees are the spanning arborescences that can be obtained, starting from a forest containing $n + 1$ arborescences with one node, by iteratively merging two consecutive arborescences, where *merging* two arborescences consists in adding an arc linking their root[3], and two arborescences are *consecutive*

[1] w_i refers both to the i^{th} word of the sentence and its associated node.
[2] Note that the head word w_i is not the head of the arc but its tail.
[3] If w_0 is one of the roots, the arc must leave w_0.

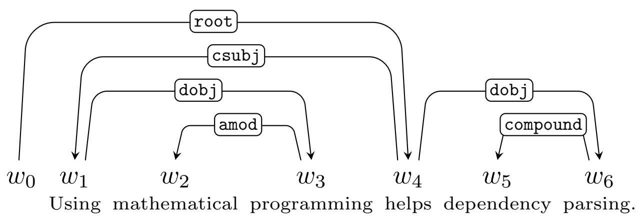
root
csubj
dobj
amod
dobj
compound
w_0 w_1 w_2 w_3 w_4 w_5 w_6
Using mathematical programming helps dependency parsing.

Figure 5.1: Example of a dependency tree of the sentence "*Using mathematical programming helps dependency parsing.*". Syntactic relations are given as labels on the arcs: `dobj` for direct object, `csubj` for clausal subject, `amod` for adjectival modifier and `compound` for nominal compounds (the label `root` is of course not a syntactic relation since w_0 is a dummy word).

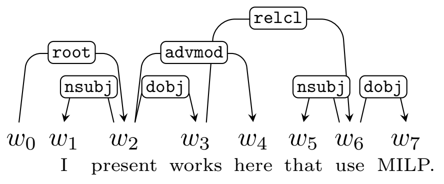
relcl
root
advmod
nsubj
dobj
nsubj
dobj
w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7
I present works here that use MILP.

Figure 5.2: Example of a non-projective dependency tree.

if there exists $j \in \{0, \dots, n-1\}$ such that one arborescence contains w_j while the other contains w_{j+1}.

A *constituent* is a set of words that form a meaningful part of the sentence from a grammatical point of view. A constituent is said *discontinous* if the words are not consecutive in the sentence. Since constituents usually induce subarborescences in dependency trees, requiring projective dependency trees is usually too restrictive (Nivre, 2006). For instance, the dependency tree given in Figure 5.2 is not projective due to the arcs (w_2, w_4) and (w_3, w_6). However, imposing no constraints on the spanning arborescence rooted at w_0 may be too permissive. Hence, different requirements have been considered that are less restrictive than projectivity. Among them are the k-block degree requirement and well-nestedness.

Block degree requirement Some grammatical formalisms impose some continuity on the words covered by subarborescences that can be expressed as follows. For a dependency tree T, a node t is *reachable* from a node s if $s = t$ or there exists a path from s to t in T. The *yield* of a node $v \in V$ corresponds to the set of nodes reachable from v with respect to T.

The *block degree* of a node set $W \subseteq V \setminus \{w_0\}$ is the number of nodes of W without their predecessor inside W, where the *predecessor* of a node w_i is the node w_{i-1}. The *block degree* of a node is the block degree of its yield and the *block degree* of T is the maximum among the block degrees of $w_1, \ldots, w_n$. T is *k-bounded block degree* (*k-BBD* for short) if its block degree is less than or equal to k. In Figure 5.2, the block degree of node w_3 is two since its yield contains w_3 and w_5 but not w_2 nor w_4. Since the block degree of every other node of $V \setminus \{w_0\}$ is one, the block degree of T is two.

Well-nestedness Two disjoint node subsets I_1 and I_2 of $V \setminus \{0\}$ *interleave* if there exist $i, j \in I_1$ and $k, \ell \in I_2$ such that $i < k < j < \ell$. A dependency tree is *well-nested* if there do not exist two nodes whose yields are disjoint and interleave. In Figure 5.2, the dependency tree is well-nested as the yield of every node of size greater than one contains w_7.

5.1.2 Dependency Parsing Algorithm

The *dependency parsing problem* consists, given a sentence $w = w_0, w_1, \ldots, w_n$ where w_0 is a dummy root word, in determining the dependency tree of that sentence. Since the 1990's, it is cast as an optimization problem consisting in returning the most probable dependency tree. For this, machine learning is used to score parts of the DT and the parsing problem is to find the DT with the highest score. The dependency parsing problem depends on the parts of the DT that are scored as well as the grammar formalism used to characterize the set of valid DT.

In (Corro, Le Roux, Lacroix, et al., 2016), a score is assigned to each couple of words that represents the probability that the syntactic relation of this couple appears in the DT (the chosen label is the most probable one). The score of a DT is then the sum of the scores of its arcs. Such type of score is referred to in the literature as *arc factored score*. There exist other scoring functions such as second-order scores (Eisner, 1997) which in addition consider for instance scores associated with pairs of arcs having the same tail, or scores associated with paths of length two.

With arc factored scores and no additional requirements on the DT, the parsing problem is a maximum spanning arborescence problem. Let $D = (V, A)$ be a directed graph where $V = \{w_0, w_1, \ldots, w_n\}$ is the set of nodes associated with the words of the sentence, and A is the set of possible syntactic relations[4]. Let $s \in \mathbb{R}^A$ be the scores associated with each possible syntactic relation. The parsing problem consists in determining a spanning arborescence T rooted at w_0 and maximizing $\sum_{a \in T} s_a$ (McDonald, Pereira, et al., 2005). This problem can be solved in $O(|V|^2)$ using Tarjan's algorithm (Tarjan, 1977). When restricting the set of valid DT to projective ones, dependency parsing can be done in $O(|V|^3)$ using Eisner algorithm (Eisner, 1996).

In (Corro, Le Roux, Lacroix, et al., 2016), we consider for valid DT those that are well-nested and k-BDD, where k is a given constant depending on the language[5]. While there exists a dynamic programming based parsing algorithm

[4]A contains no arc entering w_0 since it is a dummy word.

[5]We also perform some experiments in which we drop wellnestedness.

for such DT, its complexity is in $O(n^{3+2k})$ where n is the number of words in the sentence (Gómez-Rodríguez, Weir, and Carroll, 2009). Hence, it is too ineffective to be used for parsing large data.

We formulate this parsing problem as an integer linear problem. Before presenting the formulation, we give a graph characterization of well-nested and k-BDD dependency trees that are used to derive valid inequalities. Let $\mathcal{W}^{\geq k}$ be the family of node subsets of $V \setminus \{w_0\}$ with block degree greater than or equal to k. We denote by $\mathcal{I}$ the family of couples of disjoint interleaving node subsets of $V \setminus \{0\}$. The following two propositions characterize whether a dependency tree is k-BDD or well-nested.

Proposition 5.1. *A dependency tree T is not k-BBD if and only if there exists a node subset $W \in \mathcal{W}^{\geq k+1}$ whose cut $\delta(W)$ contains a unique arc of T.*

Proposition 5.2. *A dependency tree T is not well-nested if and only if there exists $(I_1, I_2) \in \mathcal{I}$ such that $\delta(I_1) \cap T$ and $\delta(I_2) \cap T$ are singletons.*

From Propositions 5.1 and 5.2, one can derive the following formulation of the dependency parsing problem:

$$\begin{cases} \max w^\top z & & \text{(5.1a)} \\ z(\delta^{\text{in}}(v)) = 1 & \forall v \in V \setminus \{w_0\}, & \text{(5.1b)} \\ z(\delta^{\text{in}}(W)) \geq 1 & \forall W \subseteq V \setminus \{w_0\}, & \text{(5.1c)} \\ z(\delta(W)) \geq 2 & \forall W \subseteq \mathcal{W}^{\geq k+1}, & \text{(5.1d)} \\ z(\delta(I_1)) + z(\delta(I_2)) \geq 3 & \forall (I_1, I_2) \in \mathcal{I}, & \text{(5.1e)} \\ z \in \{0,1\}^A. & & \text{(5.1f)} \end{cases}$$

By (5.1b), every node but w_0 has one entering arc. By (5.1c), z is the incidence vector of a set of arcs inducing a connected graph. Since G has no arc entering w_0, z is the incidence vector of a dependency tree. By Proposition 5.1 and (5.1d), this DT is k-BDD. By Proposition 5.2 and (5.1e), it is also well-nested.

We propose a Lagrangian based heuristic to solve System (5.1). Dualizing inequalities (5.1d) and (5.1e), the Lagrangian relaxed problem reduces to computing a maximum spanning arborescence rooted at w_0 where the arc costs are parametrized by the Lagrangian multipliers associated with (5.1d) and (5.1e). The Lagrangian relaxed problem can be solved in $O(n^2)$ using Tarjan's algorithm (Tarjan, 1977). Since the number of dualized constraints is exponential, we use a non delayed relax-and-cut algorithm to solve the Lagrangian dual problem, see Section 4.1.2. At each iteration of this variant of the subgradient algorithm, the separation of Inequalities (5.1d) and (5.1e) is done in $O(n^2)$ using the algorithms described in (Möhl, 2006) and (Havelka, 2007), respectively. Moreover, at each iteration, a step of variable fixing is performed to reduce the size of the problem. Our heuristic returns the best solution found after a fixed number of iterations of the non delayed relax-and-cut algorithm, and if no solution is found, it returns the best projective tree using the Eisner algorithm (Eisner, 1996).

Note that we also embed our Lagrangian heuristic inside a branch-and-bound tree to derive an exact method to this parsing problem.

To learn arc scores, we use a structured perceptron model. Given a sentence, a high-dimensional sparse feature vector is extracted for each word based on linguistic properties. A feature vector $f_a \in \mathbb{R}^d$ for each arc $a = (w_i, w_j) \in A$ is obtained by concatenating the feature vectors of w_i and w_j and another vector containing features involving both words (McDonald, Crammer, and Pereira, 2005). The score of arc a is given by $\theta^\top f_a$, where $\theta \in \mathbb{R}^d$ is the parameter of the structured perceptron model. The energy of a DT is the sum of the scores of its arcs. During training, the inference problem is given by removing block degree requirements and wellnestedness, that is, the inference problem amounts to computing a w_0-spanning arborescence.

From an experimental point of view, our LR heuristic is competitive with projective tree parsing (Eisner), non projective tree parsing (Maximum spanning arborescence) and a mixture of them (compute the max spanning arborescence and if it is not valid, return the best projective tree). Moreover, it always returns a valid dependency tree on the contrary to the non projective tree parsing.

5.2 Discontinuous Constituency Parsing

In this section, we first introduce parse trees and present discontinous spinal TAG formalism. We then present a Lagrangian decomposition based heuristic for the parsing problem under this formalism.

5.2.1 Parse Trees and Discontinuous Spinal TAG

Context free grammar is a grammar formalism that consists in a set of rules for constituent formation and merging, where a *constituent* (or *phrase*) is a set of words that behave as a single unit from a grammatical point of view. Each rule $\ell \to r$ contains a non-terminal symbol ℓ and a list r of symbols that can replace ℓ. The symbols in r may be terminal (*i.e.*, words) or non-terminal. Figure 5.3 gives examples of such rules.

$$\begin{array}{rcl} \text{S} & \to & \text{NP} \\ \text{NP} & \to & \text{Det Noun} \\ \text{Det} & \to & \textit{the} \\ \text{Noun} & \to & \textit{algorithm} \end{array}$$

Figure 5.3: Example of a set of rules of a context free grammar.

The rules can be combined to generate the sentence from the initial non-terminal symbol S (for sentence). Such a construction is represented as a tree and is called *parse tree*. Figure 5.4 provides a parse tree obtained with the rules given in Figure 5.3, whereas Figure 5.5 gives a parse tree of a complete sentence.

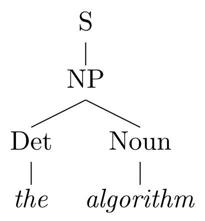

Figure 5.4: Example of a parse tree constructed from the set of rules of Figure 5.3.

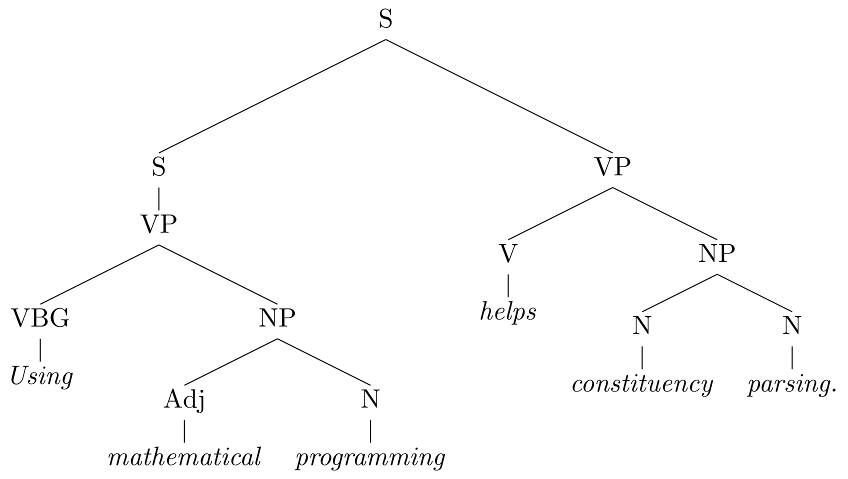

Figure 5.5: A parse tree of the sentence "*Using mathematical programming helps constituency parsing.*".

Shen and Joshi (2005) introduce spinal TAG formalism. A *spine* is a sequence of non-terminal symbols ending with a terminal one. The first non-terminal symbol of the spine is its *head*. A parse tree is obtained by associating a spine to each word and linking these spines via adjunctions, where an *adjunction*[6] consists in attaching the head of a spine to a non-terminal symbol of another spine. This non-terminal symbol is the *adjunction site*. Since constituents are usually words covered by subarborescences in dependency trees, the spine adjunctions must form a DT, that is, we consider an arc (w_i, w_j) when the head of a spine with terminal symbol w_j is attached to a spine with terminal symbol w_i (independently to the adjunction site it is attached to), and this set of arcs must form a dependency tree (see Section 5.1.1 for details on DT). Figure 5.6 gives an example of a parse tree under spinal TAG formalism.

[6]Following TAG formalism, the adjunction may be regular or sister and must be labelled accordingly; in our presentation, we do not differ both types of adjunctions.

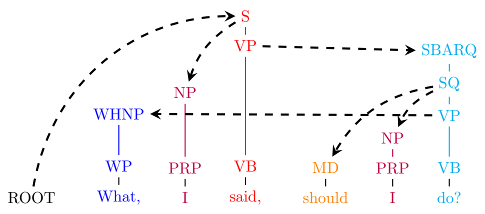

Figure 5.6: A parse tree with spines and adjunctions (dashed arrows). Each color corresponds to a spine. The spine `WHNP-WP-What` with head `WHNP` is adjoined to the spine `SBARQ-SQ-VP-VB-do?` and the adjunction site is `VP`. The induced dependency tree is non-projective.

5.2.2 Parsing Problem for Discontinous Spinal TAG

The parsing problem consists, given a sentence $w = w_0, w_1, \dots, w_n$ where w_0 is a dummy root word, in determining a spine s_i associated with each word w_i, $i = 0, \dots, n$, a DT T of w, and for each syntactic relation (w_i, w_j) of T the adjunction site of the spine associated with w_i used for adjoining w_j to w_i.

In our approach, the determination of the adjunction sites is done in a postprocessing step that is not described here. The parsing problem then corresponds to determining the spines and the DT. Moreover, the set of possible spines for each word w_i of w is restricted to the most probable m spines $S_i^1, \dots, S_i^m$ that are extracted in a preprocessing step not described here[7].

The parsing problem is cast as an optimization problem consisting in finding the most probable parse tree. The probability of a parse tree is simplified to be decomposable by arcs of the associated DT. More precisely, the probability of arc (w_i, w_j) is expressed as $P_\alpha((w_i, w_j)|w) * P_\nu(s_j|(w_i, w_j), w)$ where $P_\alpha((w_i, w_j)|w)$ is the probability of having the syntactic relation (w_i, w_j) in the DT knowing the sentence w, and $P_\nu(s_j|(w_i, w_j), w)$ is the probability of having spine s_j associated with word w_j knowing the sentence w and that (w_i, w_j) is a syntactic relation of the associated DT. The probability of a parse tree is then defined as:

$$\prod_{(w_i, w_j) \in T} P_\alpha((w_i, w_j)|w) * P_\nu(s_j|(w_i, w_j), w).$$

Probabilities $P_\alpha((w_i, w_j)|w)$ and $P_\nu(s_j|(w_i, w_j), w)$ are predicted by two separate neural networks. A context sensitive representation of each word is first computed using bidirectional recurrent networks. Then, the probabilities are predicted from this word representation using a feed forward neural network followed by a biaffine attention layer, see (Corro, Le Roux, and Lacroix, 2017) for more details.

In (Corro, Le Roux, and Lacroix, 2017), we reformulate the parsing problem as a generalized maximum spanning arborescence problem. Indeed, we consider a digraph $D = (V, A)$ where each node of V is associated with a spine for

[7]For the dummy root node, the spine is unique and fixed.

each word. A partition $\pi = \{V_0, \ldots, V_n\}$ of V is obtained by considering for $i = 0, \ldots, n$ a class V_i containing all the spines associated with word w_i. The arc set A consists in all arcs (k, ℓ) where k and ℓ belong to different classes of the partition and ℓ does not belong to V_0. The score $\phi_{(k,\ell)}$ of each arc (k, ℓ) with $k \in V_i$ and $\ell \in V_j$ is equal to $\log(P_\alpha((w_i, w_j)|w) * P_\nu(s|(w_i, w_j), w))$ where s corresponds to the spine associated with node ℓ. A solution to the generalized maximum spanning arborescence problem, that is, a *Generalized Spanning Arborescence (GSA)*, is a couple (W, T) where W is a set of $n + 1$ nodes intersecting once each class of π, and T is an arborescence spanning W and rooted at the node v_0 of V_0. The generalized maximum spanning arborescence consists in determining a GSA (W, T) such that $\phi(T)$ is maximum.

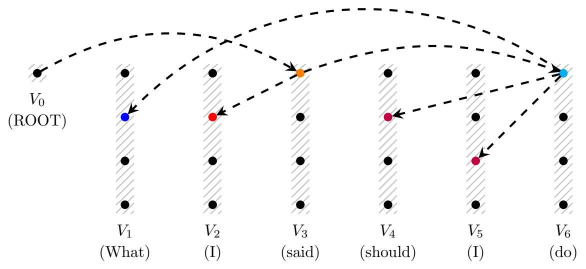

Figure 5.7: The generalized spanning arborescence inducing the parse tree in Figure 5.6.

Let D^π be the graph obtained from D by identifying every cluster to a single node. Since every cluster is a stable set, there is a one-to-one correspondence between the arcs of D and those of D^π. It follows that for a node set W intersecting each cluster once, and a set of arcs T, (W, T) is a GSA if and only if $T \subseteq A[W]$ and T is a spanning arborescence rooted at v_0 in D^π.

We provide an integer linear programming formulation for the generalized maximum spanning arborescence problem. A GSA (W, T) is encoded using the following variables. Let $x \in \{0, 1\}^V$ be such that x_v equals 1 if v belongs to W, and 0 otherwise. Similarly, let $y \in \{0, 1\}^A$ be such that y_a equals 1 if $a \in T$, and 0 otherwise. The generalized maximum spanning arborescence can be formulated as follows.

$$\begin{cases} \max \phi^\top y & & \text{(5.2a)} \\ x(V_k) = 1 & \forall k = 0, \dots, n, & \text{(5.2b)} \\ y(\delta^{\text{in}}(v)) = x_v & \forall v \in V \setminus V_0, & \text{(5.2c)} \\ y(\delta^{\text{out}}(v) \cap \delta^{\text{in}}(V_k)) \leq x_v & \forall k = 0, \dots, n, \forall v \in V \setminus V_k, & \text{(5.2d)} \\ y(\delta^{\text{in}}(\underset{V_k \in \pi'}{\cup} V_k)) \geq 1 & \forall \pi' \subseteq \pi \setminus \{V_0\} & \text{(5.2e)} \\ x_v \in \{0,1\} & \forall v \in V & \text{(5.2f)} \\ y_a \in \{0,1\} & \forall a \in A & \text{(5.2g)} \end{cases}$$

Let $(\bar{x}, \bar{y})$ be a solution to (5.2) and let $W = \{v \in V \mid \bar{x}_v = 1\}$ and $T = \{a \in A \mid \bar{y}_a = 1\}$. Equations (5.2b) imply that W contains one node per cluster of π. By constraints (5.2c) and (5.2d), T only contains arcs of $A[W]$. By (5.2b) and (5.2c), T enters once every node of D^π but v_0. Inequalities (5.2e) ensure that T induces a connected graph in D^π, so T is a v_0-spanning arborescence of D^π (Schrijver, 2003). Hence, (W, T) is a GSA.

We propose a Lagrangian decomposition based heuristic built upon the following reformulation of (5.2). It consists in replacing variables y by three copies y^0, y^1, y^2, and ensuring that these copies have the same value using extra variables z variables. The choice of the copy y^i used to replace y inside each set of constraints is done to obtain decomposable subproblems in the Lagrangian relaxed problem. We get the following reformulation:

$$\begin{cases} \max \frac{1}{3}\phi^\top (y^0 + y^1 + y^2) & & \text{(5.3a)} \\ x(V_k) = 1 & \forall k = 0, \dots, n, & \text{(5.3b)} \\ y^1(\delta^{\text{in}}(v)) = x_v & \forall v \in V \setminus V_0, & \text{(5.3c)} \\ y^2(\delta^{\text{out}}(v) \cap \delta^{\text{in}}(V_k)) \leq x_v & \forall k = 0, \dots, n, \forall v \in V \setminus V_k, & \text{(5.3d)} \\ y^0(\delta^{\text{in}}(\underset{V_k \in \pi'}{\cup} V_k)) \geq 1 & \forall \pi' \subseteq \pi \setminus \{V_0\}, & \text{(5.3e)} \\ y^0(\delta^{\text{in}}(V_k)) = 1 & k = 1, \dots, n & \text{(5.3f)} \\ y^i = z & \forall i = 0, 1, 2, & \text{(5.3g)} \\ x_v \in \{0,1\} & \forall v \in V, & \text{(5.3h)} \\ y^i_a \in \{0,1\} & \forall a \in A, \forall i = 0, 1, 2. & \text{(5.3i)} \end{cases}$$

First, note that equations (5.3f) are redundant constraints obtained from (5.2) by summing for each $k = 1, \dots, n$ equations (5.2c) for all $v \in V_k$ and the (5.2b) associated with k. These redundant constraints strengthen the following decomposition.

By dualizing equations (5.3g), we obtain a Lagrangian relaxed problem that can decomposed into disjoint subproblems. Note first that since z appear in no constraint of this relaxed problem, the latter is finite only when the coefficient in the objective function associated with z equals $\mathbf{0}$, implying that the sum of the dual variables equal $\mathbf{0}$, and variable z vanishes. A first subproblem involves variables y^0 and constraints (5.3e) and (5.3f). Solving this subproblem reduces to computing a v_0-spanning tree in D^π and can be done in $O(n^2)$.

Since y^1 and y^2 are only linked by variables x, the remaining part can be decomposed by cluster V_k for $k = 0, \ldots, n$. The problem associated with V_k reduces to selecting for each node $v \in V_k$ its entering arc with the highest cost and for every cluster V_ℓ with $\ell \neq k$, to selecting the outgoing arc of v entering V_ℓ with the highest cost if it is positive. The sum over the costs of these arcs corresponds to cost of a node, and the problem is nothing but choosing the node in each cluster with the maximum cost. Solving the subproblems associated with all clusters can then be done in $O(|A|)$.

Hence solving the Lagrangian relaxed problem can be done in an efficient way.

The Lagrangian dual problem is solved using a projected subgradient descent[8]. At each iteration of this subgradient algorithm, a variable fixing scheme is applied to reduce the size of the problem, and a reweighting of the cost is performed to help the convergence of the subgradient descent. Moreover, at each iteration, we also determine a GSA from the current solution $\bar{x}, \bar{y}$ to the Lagrangian relaxed problem by computing a spanning arborescence rooted at v_0 on the graph induced by the nodes $v \in V$ for which $\bar{x}_v = 1$. The algorithm returns the best GSA found after 500 iterations of the subgradient algorithm.

From an experimental point of view, our algorithm outperforms the state-of-the-art both in terms of accuracy and of running time. Moreover, with our method we can obtain a certificate of optimality of the solutions given by our heuristic for more than 99% of the sentences of the datasets.

[8]It is a projected gradient descent to ensure that at each iteration, the sum of the dual variables is zero, see (Corro, Le Roux, and Lacroix, 2017) for more details.

Chapter 6

Computing Dual Bounds to Mixed-Integer Linear Problems using Machine Learning

This chapter presents my research relative to the incorporation of machine learning inside mathematical programming algorithms to enhance their performance. Indeed, mathematical programming solvers usually rely on heuristic decisions which may dramatically affect their performance, even when these algorithms are exact ones. Well-known examples are branch-and-bound based methods where the choice of the node to process (*aka,* node selection) or the variable to branch on (*aka,* variable selection) have a huge impact on the size of the branch-and-bound tree, and hence on the overall running time. A vast research litterature exists on handcrafted heuristics but since a few years, machine learning is used instead to tune such decisions (Bengio, Lodi, and Prouvost, 2021; Zhang, Liu, et al., 2023; Scavuzzo et al., 2024). The aim is to learn from data underlying properties of an instance and use them to drive the heuristic decisions.

A key component for the efficiency of branch-and-bound methods is the tightness of dual bounds. When this bound is provided by the continuous relaxation, strengthening inequalities may be incorporated to the relaxation using a cutting-plane scheme, leading to branch-and-cut algorithms (Mitchell, 2009). The heuristics devised for generating, selecting and removing cuts at each iteration are fundamental for such algorithms (Marchand et al., 2002; Dey and Molinaro, 2018; Wesselmann and Suhl, 2012) and machine learning is now at the core of recent research questions and advances (Deza and Khalil, 2023).

Lagrangian relaxation is another way to provide high-quality dual bounds for mixed-integer linear problems. It consists in dualizing some linear inequalities, that is, removing them but penalizing their violation in the objective function. For a trade-off between the original objective and the dualized inequalities' satisfaction, a weighted sum of the violation is done. The weight associated with each inequality is the Lagrangian multiplier and each feasible Lagrangian multiplier vector provides a dual bound. To obtain accurate bounds, one usually applies iterative subgradient based methods such as the subgradient algorithm or the bundle method, see Section 4.1 for more details. However, these iterative methods may be time-consuming. To encompass this problem, we propose to use machine learning to predict Lagrangian multipliers. Assuming that instances follow some unknown patterns, our aim is that machine learning catches such patterns to make predictions for new instances sharing these patterns.

This approach stems from a joint work with F. Demelas, J. Le Roux and A. Parmentier (Demelas et al., 2024).

6.1 Learning Framework

Providing dual bounds based on Lagrangian relaxation can be interpreted as a machine learning task. Our objective is to provide a generic tool where genericity is intented as being able to apply it on a large range of mixed-integer linear problems and Lagrangian relaxations. For this, the learning task input only consists of a mixed-integer linear problem (P) and the set R of the linear constraints of (P) that are dualized in the Lagrangian relaxation; it is denoted by $\iota = (P, R)$. The output is the set of Lagrangian multipliers associated with the dualized constraints. The dual bound is then obtained by computing with a combinatorial algorithm the Lagrangian relaxed problem parametrized by the predicted Lagrangian multipliers. By Lagrangian duality, the optimum of the Lagrangian relaxed problem gives a dual bound to the input problem. The learning task is to predict Lagrangian multipliers providing a bound approaching the value of the Lagrangian dual problem, see Section 4.1 for more details on Lagrangian relaxation.

To have a complete representation of the input, the objective function, the right-hand side and the constraint matrix defining the input problem must be given explicitly. This implies that our approach only handles compact mixed-integer linear problems, that is, problems with a polynomial number of variables and constraints.

We focus on Lagrangian relaxations such that the Lagrangian dual problem provides a tighter bound that the one of the continuous relaxation. This assumption is not too restrictive as the continuous relaxation of a compact formulation can be efficiently solved using a linear solver. This assumption offers two advantages. Firstly, given an input $\iota = (P, R)$, the continuous relaxation of (P) can be solved to get an optimal pair of primal and dual solutions x^ι and λ^ι, and these solutions can be used to extract initial features, say χ^ι. Secondly, instead of directly predicting a Lagrangian multiplier for each dualized constraint $r \in R$, we predict its deviation with respect to the value λ^ι_r of λ^ι associated with r, following the same idea as (He, Zhang, et al., 2016) for residual connections.

Since optimal Lagrangian multipliers may not be unique, we build our model under unsupervised learning framework, that is, we do not try to predict Lagrangian multipliers that are close to specific optimal ones considered as labels. Instead, to measure the quality of our predicted Lagrangian multipliers, we use the value of the Lagrangian relaxed problem parametrized by our prediction.

We use a latent space to represent the dualized constraints. More specifically, we use a probabilistic encoder-decoder scheme. The probabilistic encoder q_ϕ, parametrized by ϕ, outputs parameters of a normal distribution over the latent space for each dualized constraint. By sampling, we get a latent representation z_r of each dualized constraint $r \in R$. This value z_r is used as input for the deterministic decoder f_θ, parametrized by θ, that predicts the deviation of the Lagrangian multiplier π_r with respect to λ^ι_r. Since inequalities are dualized, the Lagrangian multipliers must be nonnegative. Then, $\pi_r = [\lambda^\iota_r + f_\theta(z_r)]_+$ for each dualized constraint $r \in R$, where $[\cdot]_+$ denotes the softplus function[1]. The (Lagrangian) dual function is used as the loss function.

Given a training dataset $\mathcal{I}$, the learning task is to find parameters ϕ^* and θ^* such that:

$$\phi^*, \theta^* = \arg\min_{\phi,\theta} \sum_{\iota \in \mathcal{I}} \mathbb{E}_{z \sim \mathcal{N}(q_\phi(\chi^\iota))} \left[LR([\lambda^\iota_{|R} + f_\theta(z)]_+; \iota) \right],$$

where $z \in \mathbb{R}^{d \times R}$ is a matrix whose column $z_r \in \mathbb{R}^d$ is the latent representation of constraint r obtained by sampling over a normal distribution parametrized by $q_\phi(\chi^\iota)$, $\lambda^\iota_{|R}$ is the restriction of λ^ι to the coordinates associated with R, and $f_\theta(z)$ is the vector obtained by contatenating the value $f_\theta(z_r)$ for all $r \in R$.

6.2 Neural Network Architecture

In this section, we describe each part of the probabilistic encoder-decoder. Its overall architecture is depicted in Figure 6.1.

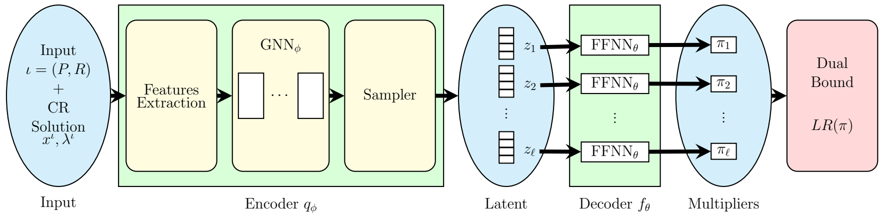

Figure 6.1: Probabilistic encoder-decoder designed to predict a Lagrangian dual bound.

[1]On the contrary to $\max\{0, \cdot\}$ function, the soft version is differentiable at 0 and the gradient for nonzero values is more informative as it is not only 0 or 1.

Feature extraction Given an input $\iota = (P, R)$, a feature vector is obtained for each variable and each constraint of (P) by using information from the objective function and the constraints of (P), from the set R of dualized constraints, and from an optimal pair of primal and dual solutions to the continuous relaxation of (P)[2]. Each vector is used as input of a feed forward neural network to get a high-dimensional initial feature vector for each variable and each constraint.

Graph neural network A graph neural network is used to compute meaningful features for the dualized constraints from the high-dimensional initial features. For this, (P) is represented as a bipartite graph following (Gasse et al., 2019) that will be used to perform graph convolutions.

The bipartite graph $G = (U \cup V, E)$ is constructed as follows. The node sets U and V contain a node per variable and per onstraint of (P), respectively. For all $u \in U$ and $v \in V$, there is an edge uv in E if the variable associated with u has a nonzero coefficient in the constraint associated with v.

The graph neural network is composed of ℓ layers in order to refine the high-dimensional initial node features. Each layer is a graph neural layer as represented in Figure 6.2. This layer is based on a graph convolutional layer as described in Section 4.2.2 but both operations forming this layer, namely the linear convolution and the non-linear transformation[3], are preceeded by a normalization layer (Ba, Kiros, and Hinton, 2016), followed by a dropout layer (Srivastava et al., 2014), and sorrounded by a residual connection that adds the input to the output (He, Zhang, et al., 2016), similarly to what is done in the transformer architecture (Vaswani et al., 2017).

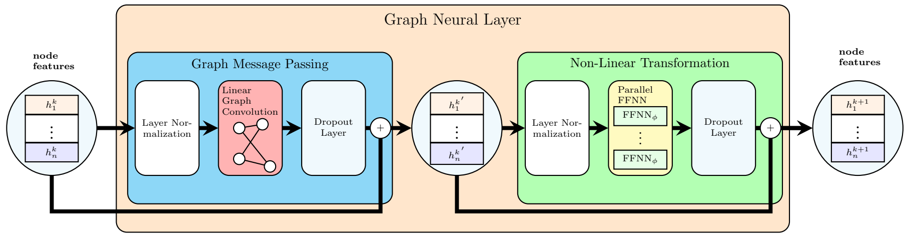

Figure 6.2: A visual representation of the Graph Neural Layer used in the probabilistic encoder.

Sampler The graph neural network computes node embeddings for each variable and each constraint of (P). The ones associated with the dualized constraints of R are used to sample a latent representation of these constraints.

To enable backpropagation, we use the reparametrization trick (Kingma and Welling, 2014): for each $r \in R$, z_r is the output of a deterministic function taking

[2]See (Demelas et al., 2024) for a precise description of the initial features.

[3]The linear convolution corresponds to the operations inside parenthesis whereas the non-linear transformation corresponds to σ in Equation (4.21).

some paremeters predicted by the encoder as well as a random noise following a normal distribution, that is:

$$z_r = \mu_r + exp(\tau_r) \odot \xi,$$

where ξ is a standard normal random vector, that is, each of its components ξ_i is given by $\xi_i \sim \mathcal{N}(0, 1)$. The value $(\mu_r \parallel \tau_r)$ corresponds to the embedding associated with the node in the last layer of our GNN associated with constraint $r \in R$. Following (Chua et al., 2018), we clip the variance $exp(\tau_r)$ to a safe interval using the softplus function.

Decoder The deterministic decoder f_θ is a single feed forward neural network taking as input a latent representation z_r of a constraint $r \in R$ and outputing the deviation of the Lagrangian multiplier π_r associated with r with respect to λ_r^ι, that is $\pi_r = [\lambda_r^\iota + f_\theta(z_r)]_+$.

6.3 Evaluation

Our approach is evaluated on two classical operations research problems that are briefly described in the next two paragraphs. For more details on the problems, their formulation and the considered Lagrangian relaxations, the interested reader should look at the Appendix of (Demelas et al., 2024).

Multi Commodity Capacitated Network Design (MC) In this problem, one has a capacitated digraph and a set of commodities to route at minimum cost from their origin to their destination while not exceeding arc capacities. The cost of a solution is given by the the sum of the fixed costs of the arcs used to route commodities plus the sum over these arcs of the routing cost that linearly depends on the amont of commodities carried on the arc. Note that each commodity may be routed on several paths.

The problem may be formulated as a mixed-integer linear problem by considering continuous flow variables for each commodity and a binary variable for each arc to indicate whether or not an arc is used to route commodities. This formulation contains three families of linear constraints: the flow conservation equations to ensure having a flow for each commodity, the capacity inequalities to force the amount of flow routed through any arc to be no more than its capacity, and bound inequalities.

A classic Lagrangian relaxation is obtained by dualizing the flow conservation constraints[4]. The Lagrangian relaxed problem then decomposes per arc and corresponds for each arc to a continuous knapsack problem.

[4]Since the dualized constraints are equations, the Lagrangian multipliers are not required to be nonnegative. Then, in our model, the prediction is given by $\pi_r = \lambda_r^\iota + f_\theta(z_r)$ for each constraint $r \in R$.

Generalized Assignment (GA) In this problem, there is a set of items and a set of bins, each with different capacities. The weight and profit of each item vary depending on the bin it is assigned to. The objective is to assign items to bins in a way that maximizes the total profit while ensuring that the combined weight of items in any bin does not exceed its capacity. Note that an item may remain unassigned to any bin.

The problem may be formulated as a binary linear problem by considering a binary variable for each pair of item and bin in order to indicate whether or not this item is assigned to that bin. This formulation contains two families of inequalities: packing constraints ensuring that each item is assigned to at most one bin and capacity inequalities forcing the sum of the weights of the items assigned to any bin to not exceed its capacity.

A Lagrangian relaxation is obtained by dualizing the packing constraints. The resulting Lagrangian relaxed problem decomposes per bin and corresponds for each bin in a binary knapsack problem.

For both problems, our model provides accurate bounds in short running time. It is orders of magnitude faster than solving the Lagrangian dual problem using state-of-the-art bundle methods while providing an average gap that ranges from 2% for the small MC instances to less than 5% for the big ones. For GA, it is less than 2%. Our method provides really more accurate bounds than the ones given by solving the Lagrangian relaxed problem parametrized by Lagrangian multipliers predicted using baseline machine learning models like k-Nearest Neighbors regression or feed forward neural networks.

The Lagrangian multipliers predicted by our model can be used to initialize the bundle method. Indeed, this method proceeds iteratively, starting from an initial dual solution and converging to the optimal solution of the Lagrangian dual problem. The choice of the initial dual solution affects the number of iterations of the bundle method and, consequently, its running time. We have shown that using the output of our model as the initial solution reduces the total running time by between one-third and one-half, depending on the precision set for the stopping criterion of the bundle method.

Research Perspectives

In this last chapter, I briefly expose several research directions that correspond to ongoing works, or research questions I would like to address in the future. The research questions presented here are related with the works of current Ph.D. students under my co-supervision or open questions raised during the Ph.D. theses I co-supervised that were recently defended. Questions connected to parts I and II of this document are presented in Section 1 and 2, respectively. In Section 3, I present a research topic I have started working on this year.

1 Perspectives Related to Polyhedra

Polyhedra are everywhere in mixed-integer linear programming since the convex hull of the solutions to any such problem is a polyhedron. Studying these polyhedra gives insights on the structure of the solutions and may provide polynomial-time solvable cases and min/max relations. As such, polyhedra is one of my favorite way to tackle any mixed-integer linear problem and I intend to continue exploring these objects in my research. Hence, the following research questions do not fully represent my future research on polyhedra but correspond to some questions emerged from my previous and current works that I want to answer in the next future.

1.1 The Co-2-plex Polytope

This research perspective is a question that was raised at the end of the Ph.D. thesis of Alexandre Dupont-Bouillard.

Given a graph $G = (V, E)$, a *co-k-plex* is a set of nodes inducing a graph in which the maximum degree is less than k. Co-k-plexes generalize stable sets as these latter are co-1-plexes.

Recently, co-k-plexes have been studied since they allow to define cohesive groups in the complement graph of a social network in a less restrictive manner than stable sets (Seidman and Foster, 1978).

In (Dupont-Bouillard et al., 2024), we introduce the utter graph $u(G)$ of a graph $G = (V, E)$ so that there is a bijection between the co-2-plexes of G and the stable sets of $u(G)$. The *utter graph* $u(G)$ is defined as follows. It contains a node for each node $v \in V$ and for each edge $e \in E$. It has an edge uv

if the elements of G corresponding to u and v are either incident, adjacent or adjacent by contraction, where a node v and and a nonincident edge e (resp. two nonincident edges e and f) are *adjacent by contraction* if contracting e (resp. e and f) results in two adjacent nodes.

When the graph is perfect, its stable *set polytope*, that is the convex hull of the incidence vectors of its stable sets, is described by the clique and nonnegative inequalities (Chvátal, 1975). Due to the bijection between stable sets of $u(G)$ and co-2-plexes of G, this gives an extended formulation of the *co-2-plex polytope* of G, that is, the convex hull of the incidence vectors of its co-2-plexes, when $u(G)$ is perfect. The class of graphs whose utter graphs are perfect are the *contraction perfect graphs* which are the perfect graphs that remain perfect when contracting any set of edges (Dupont-Bouillard et al., 2024). In (Dupont Bouillard, 2024), we project the extended formulation onto the natural space $\mathbb{R}^V$ when the graph is a tree. Could we generalize this result to the class of contraction perfect graphs by projecting the extended formulation? A way to start answering this question would be to characterize the co-2-plex polytope of chordal graphs. These latter are a well-studied subclass of contraction perfect graphs and their utter graphs are also chordal. This implies that the extended formulation is compact since the number of cliques of a chordal graph is bounded by its number of nodes. Answering this first question is one of my future works.

1.2 The Box-TDIness of the Perfect Matching Polytope

This research perspective is the continuation of the work done in the context of the Ph.D. thesis of Francesco Pisanu.

A *matching* in a graph is a set of edges that are pairwise nonadjacent. A matching is *perfect* if it covers all the nodes of the graph. Perfect matchings are widely studied combinatorial objects (Lucchesi and Murty, 2024).

The *matching polytope* is the convex hull of the incidence vectors of edge sets corresponding to matchings. A linear description of the matching polytope has been given in (Edmonds, 1965), and Cunningham and Marsh (1978) prove that this system is TDI. Ding, Tan, and Zang (2018) characterize the class of graphs whose matching polytope is box-TDI.

The *perfect matching polytope (PMP)* is the convex hull of the incidence vectors of edge sets corresponding to perfect matchings. The PMP is a face of the matching polytope, and since all faces of a box-TDI polyhedron are box-TDI, the characterization of the box-TDIness of the matching polytope of Ding, Tan, and Zang (2018) gives sufficient conditions for the PMP to be box-TDI. However, there exists an infinite number of graphs whose PMP is box-TDI but its matching polytope is not. For instance, the matching polytope of the graph in Figure 1[5] is not box-TDI as it contains the subgraph in gray (Ding, Tan, and Zang, 2018). In contrast, the PMP of this graph is box-TDI. Moreover, this situation occurs for every odd subdivision of this graph, where a graph H is an *odd subdivision* of a graph G if H arises from G by replacing edges by paths with

[5]This figure is to be credited to Francesco Pisanu.

an odd number of edges. Indeed, odd subdivision preserves the box-TDIness of the PMP (Theorem 3.15 of (Pisanu, 2023)) and the non boxTDIness of the matching polytope (Theorem 3.38 of (Pisanu, 2023)).

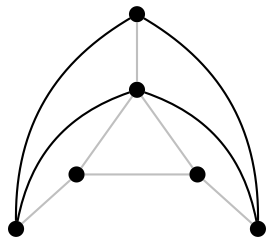

Figure 1: A graph whose PMP is box-TDI on the contrary to its matching polytope as it contains the subgraph in gray. Moreover, for any odd subdivision of this graph, the PMP remains box-TDI while the matching polytope remains non box-TDI.

A research question is to characterize the class of graphs for which the PMP is box-TDI. During the Ph.D. thesis of Francesco Pisanu, we obtained some preliminary results in this direction. Before stating them, we give some definitions.

A cut is *tight* if it intersects every perfect matching once. It is *trivial* if it is equal to $\delta(v)$ for some $v \in V$. Let $\mathcal{F}$ be a laminar family of nontrivial tight cuts of a graph G. Contracting a shore of a cut of $\mathcal{F}$ gives a smaller graph, and all the elements of $\mathcal{F}$ corresponding to cuts in the resulting graph are tight. Repeating the contractions until no element of $\mathcal{F}$ corresponds to a nontrivial tight cut gives a graph which is a *$\mathcal{F}$-contraction* of G. The family $\mathcal{F}$ has the *odd cycle property* if no $\mathcal{F}$-contraction is bipartite. A cut C has the *odd cycle property* if the family $\{C\}$ has.

In Figure 3, C and C' are two laminar nontrivial tight cuts. The different $\{C, C'\}$-contractions are given in Figure 2. Note that $\{C, C'\}$ has not the odd cycle property since the graph in Figure 2b is bipartite. However, one can check that both C and C' have the odd cycle property.

A *near-brick* is a graph having no tight cut with the odd cycle property. If $\mathcal{F}$ is a maximal laminar family of tight cuts with the odd cycle property, then every $\mathcal{F}$-contraction is a near-brick. A *near-brick of a graph G* is a near-brick which corresponds to a $\mathcal{F}$-contraction of some maximal laminar family of tight cuts $\mathcal{F}$ of G with the odd-cycle property.

We first prove that contracting the shores of laminar nontrivial tight cuts preserves the box-TDIness of the PMP.

Theorem 1 (Pisanu, 2023)**.** *Let G be a graph whose PMP is box-TDI, and let $\mathcal{F}$ be any laminar family of nontrivial tight cuts of G. Then, the PMP of each $\mathcal{F}$-contraction of G is box-TDI.*

We also characterize the class of near-bricks having a box-TDI PMP. An *odd intercyclic* graph is a graph having no two disjoint odd circuits.

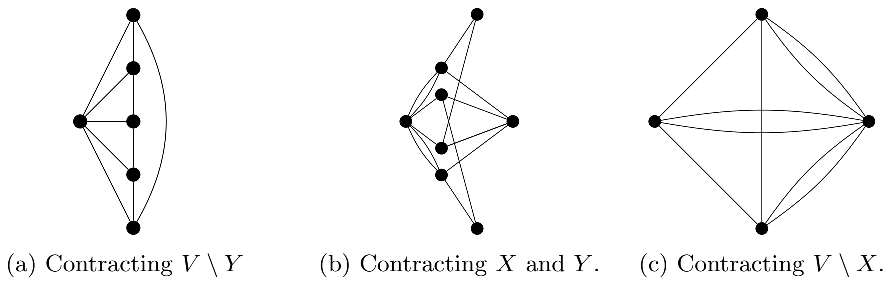
(a) Contracting $V \setminus Y$ (b) Contracting X and Y. (c) Contracting $V \setminus X$.

Figure 2: List of $\{C, C'\}$-contractions of the graph of Figure 3.

Theorem 2 (Pisanu, 2023)**.** *The PMP of a near-brick is box-TDI if and only if the near-brick is odd intercyclic.*

Theorems 1 and 2 imply that a necessary condition for the PMP of a graph G to be box-TDI is that every near-brick of G is odd intercyclic. However, this condition is not sufficient. Indeed, Figure 3 shows an example of a graph whose PMP is not box-TDI[6]. Its four near-bricks are obtained by contracting either a shore of C or of C', and they are all odd intercyclic (see (Pisanu, 2023) for more details on this example).

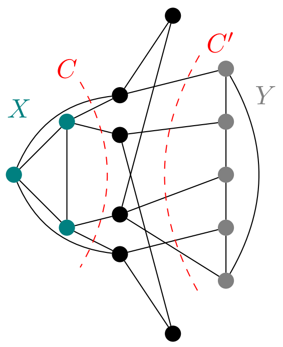

Figure 3: A graph whose PMP is not box-TDI but all its near-bricks are odd intercyclic.

This leads to the following questions: could we give necessary and/or sufficient conditions for the box-TDIness of the PMP of a graph provided that the PMP of all its near-bricks are box-TDI? Another standpoint could be considered. Edmonds, Pulleyblank, and Lovász (1982) show that for every matching covered graph, there exists a maximal laminar family of non trivial tight cuts whose cuts are of two forms: barrier cuts or 2-separation cuts[7]. We prove that the PMP of a graph G having a 2-separation cut $\delta(X)$ is box-TDI if and only

[6]This figure is also to be credited to Francesco Pisanu.

[7]Definitions of such types of cuts are omitted here but can be found in (Pisanu, 2023) for instance.

if those of G/X and $G/\overline{X}$ are. Hence, could we define some necessary and/or sufficient conditions of the same type for barrier cuts?

2 Perspectives Related to Machine Learning

The second part of this document has presented some of my works combining machine learning and mixed-integer linear programming. I intend to continue exploring the interplay between these two domains. I present in this section different research questions that correspond to my planned future work.

2.1 Using Machine Learning Alongside Bundle Methods

This research direction is an ongoing work with Francesca Demelas which has recently defended her Ph.D.

In Chapter 6, I present a work where dual bounds for mixed-integer linear problems are given using machine learning. More precisely, a model predicts Lagrangian multipliers and this prediction is used to parametrize the Lagrangian relaxed problem. Our approach is an amortization method that leverages machine learning to replace an iterative method such as subgradient or bundle methods. It can also be used as an initialization of such iterative methods.

Although the proposed method yields accurate results, it has certains drawbacks. A first one is the large dimension of the output space of our model since the number of predictions equals the number of dualized constraints. This tends to decrease the accuracy of our model when the size grows. Moreover, to obtain such accurate results, our architecture relies on a graph neural network whose size depends on the number of variables and constraints. Large datasets are required to train this end-to-end learning model, making training ressource intensive. Moreover, it is difficult to handle huge instances.

To overcome these drawbacks, we propose an alternative approach that leverages machine learning alongside the bundle method via unrolling algorithms. Such algorithms, originated from the seminal work of Gregor and LeCun (2010) for learning sparse coding, consist in representing a given number of iterations of an iterative algorithm using a recurrent neural network where each step corresponds to an iteration of the algorithm. Hence, a forward pass in the recurrent neural network corresponds to performing several iterations (equal to the number of steps in the network) of the algorithm. This leads to parametrized networks that are usually faster and need less steps than the corresponding algorithm for a same level of accuracy. Moreover, since they rely on a dedicated algorithm well-fitted for the task, they usually require less data and are more accurate than generic end-to-end neural networks (Monga, Li, and Eldar, 2021).

By using unrolling algorithms, we propose to predict, at each iteration of the bundle method, both the value of the regularization parameter used to weight the quadratic term in the objective function[8] and the optimal solution to the

[8]It corresponds to parameter η in (4.8).

restricted master problem. This allows us to reduce the running time in two different ways. On the one hand, we skip resolutions of quadratic problems. On the other hand, we reduce the number of iterations by predicting accurate solutions and regularizations.

During the Ph.D. thesis of Francesca Demelas, we propose a learning model when the Lagrangian relaxation is obtained by dualizing equations. In that case, the restricted master problem solved at each iteration is given by (4.8a)-(4.8b) since the Lagrangian multipliers are not required to be nonnegative. At each iteration of the bundle method, in order to get an optimal solution to the restricted master problem, one may solve its dual (Lemaréchal, 2001, Theorem 32). At iteration t, supposing that $\mathcal{S}' = \{x^1, \dots, x^t\}$, this dual has the following form:

$$\max_{z \in \Delta(t)} \sum_{i=1}^{t} \alpha_i z_i - \frac{\eta_t}{2} \left\| \sum_{i=1}^{t} g_i^\top z_i \right\|^2 , \tag{1}$$

where η_t is the regularization parameter at iteration t, $\{(g_i, \alpha_i) \mid i = 1, \dots, t\}$ is the gradient bundle[9] with $g_i = b - Ax^i$ and $\alpha_i = c^\top x^i - \eta_t g_i^\top \pi^s$ for all $i = 1, \dots, n$, and $\Delta(t)$ denotes the $(t-1)$-dimensional simplex, that is, $\Delta(t) = \{x \in \mathbb{R}^t_+ \mid \mathbf{1}^\top x = 1\}$.

At each step $t \in \{1, \dots, \ell\}$, a latent representation z_t is obtained by sampling following a probability distribution parametrized by the output y_t of the recurrent neural network at step t. This latent representation is given to a decoder composed of feed forward neural networks that outputs two vectors k_t and q_t corresponding to the key of (g_t, α_t) and query to (1) respectively, as well as the regularization parameter η_t. The predicted solution to (1) is given by a parameter-free attention mechanism: the query q_t is used with keys $k_1, \dots, k_t$ to get the point of $\mathbb{R}^t$ whose j^{th} component equals $k_j^\top q_t$ for $j = 1, \dots, t$. This point is projected onto $\Delta(t)$ using the sparsemax function (Martins and Astudillo, 2016) to obtain a solution to (1).

The input at each step t of the recurrent neural network is a handcrafted vector obtained using statistical measures on the solution to the restricted master problem and on (g_{t-1}, α_{t-1}).

While this approach gives good results on the multi commodity capacitated network design problem, some further investigations should be done.

Up to now, we only have considered equations as dualized constraints. Dualizing inequalities seems more difficult. Indeed, it modifies (1) and its solution set is no more a simplex, which may complexify the prediction of a solution. A first option is to keep solving (1) for the surrogate dual, construct by duality a solution to (4.8a)-(4.8b), and obtain a solution to the restricted master problem (4.8) by projecting into the nonnegative orthant[10]. However, could this approach give accurate predictions? Could we instead directly predict a feasible solution to the dual of (4.8)?

A major interest of the method is that, by considering the dual (1), the model has to predict at each iteration t a solution of dimension t. This re-

[9]We suppose in our model that a gradient is added to the bundle at each iteration.

[10]This can be done in a differentiable manner by using a softplus function for instance.

sults in a huge dimensional reduction of the prediction space compared to the approach presented in Chapter 6. However, in order to obtain a good representation of the problem, one has to find input features that correctly represent the current solution and problem. The proposed handcrafted features could be largely improved and this representation question should definitely be taken into consideration. Moreover, in the first iterations, the gradient bundle is too small so it is difficult to make accurate predictions. Could we incorporate instance features to guide the model in the first iterations when the bundle is not sufficient? This would represent a major improvement with respect to the current approach.

2.2 Solution Prediction with Graph Neural Networks and Markov Random Fields

In Chapter 6, we use machine learning to predict dual bounds of a mixed-integer linear problem by predicting Lagrangian multipliers and solving the associated Lagrangian relaxed problem. However, machine learning may also be used to give primal bounds by predicting near-optimal solutions (Zhang, Liu, et al., 2023). This constitutes a research question I have started working on in the context of Alexandre Schulz's Ph.D. thesis.

For a binary linear problem (P) with variables $x_1, \dots, x_n$, a way to predict solutions is to predict for each variable its probabilities to be equal to 0 and 1 respectively. Fixing each variable to its most probable value (or sampling independently each variable following its predicted distribution) gives a point that is unlikely to be feasible. Nevertheless, different algorithms exist that use these predicted probabilities to find high quality solutions. Nair et al. (2020) fix only a subset of variables to their most probable value. This results in a smaller problem that can be efficiently solved using traditional mixed-integer linear programming solvers. The choice of the variables to fix is made using the SelectiveNet classifier (Geifman and El-Yaniv, 2019). Ding, Zhang, et al. (2020) consider a relaxed variant of variable fixing. They consider a set F of variables corresponding to those having a high confidence, where the *confidence* of a variable is the probability of fixing it to its most probable value. Then, they add in the mixed-integer linear problem a linear constraint imposing that the number of variables in F that are not equal to their most probable value is no more than a fixed parameter α. The case $\alpha = 0$ corresponds to fixing each variable in F to its most probable value, whereas positive α implies that the model may have made some prediction errors and α may be corrected. Shen, Sun, et al. (2021) use the prediction to order variables with respect to their decreasing confidence. They solve the problem with a branch-and-bound method using a depth-first search following that order. More precisely, at each node of the branch-and-bound tree, the current relaxation is solved. If the continuous solution is not binary, then the first variable in the order with a fractional value is used for branching, and the next node to process is the child node corresponding to fixing the branching variable to its most probable value. The heuristic stops whenever a solution is found. Huang et al. (2022) combine both methods to increase performance: fix a set of variables with high confidence, and solve the smaller resulting problem with the described branch-and-bound algorithm.

To predict the variables' probabilities, these works follow the same learning scheme. A graph representation of (P) is used to derive a graph neural network. Initial features are extracted from the objective function, the constraints, and the primal and dual solutions to the continuous relaxation of (P). They serve as inputs for the graph neural network that computes variables' embeddings. These embeddings are then used as input of another neural network that predicts the probabilities of each variable to be equal to 0 and 1, respectively. Given a dataset consisting in a set of mixed-integer linear problems and their optimal solution, training the overall model to predict accurate probabilities is usually done in a supervised manner by minimizing the binary cross entropy with respect to the labelled solution.

A major drawback of these approaches is that the probability of each variable is predicted independently from the others, that is, the probability to have a solution $\bar{x}$ is given by:

$$P(x = \bar{x}) = \prod_{i=1}^{n} P(x_j = \bar{x}_j), \tag{2}$$

where $P(x_j = \bar{x}_j)$, $j = 1, \ldots, n$ are the predicted probabilities. The dependencies between variables have to be encoded via the embeddings of the graph neural network that are used for the prediction. We propose to leverage this by considering another factorization of the probability distribution using Markov random fields.

A *Markov Random Field (MRF)* models a distribution probability p over binary random variables[11] $x_1, \ldots, x_n$ that can be decomposed by factors, where a factor is a nonnegative function defined over a subset of variables. Given a set of factors $\psi_F(x_F)$, $F \in \mathcal{F}$, where $x_F \subseteq \{x_1, \ldots, x_n\}$, we consider the following distribution:

$$P(x = \bar{x}) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(x_F),$$

where $Z = \sum_{x \in \{0,1\}^n} \prod_{F \in \mathcal{F}} \psi_F(x_F)$ is the *partition function* ensuring that probabilities sum to one. A MRF is usually represented with a bipartite graph $(X \cup \mathcal{F}, E)$ where X and $\mathcal{F}$ are the set of random variables and factors respectively, and there is an edge xF for each $x \in X$ and each $F \in \mathcal{F}$ if and only if factor F is defined over a subset of variables containing x. This bipartite graph is referred to as the MRF *factor graph*[12].

[11]A MRF may be defined over nonbinary random variables but we restrict here to the binary case we are interested in to simplify the presentation.

[12]A MRF may also be represented as a graph where nodes are associated with random variables and cliques are in bijection with factors.

Considering $\mathcal{F} = \big\{\{x_1\}, \dots, \{x_n\}\big\}$ corresponds to a distribution like (2) but more complicated factors may be considered to take into account variable interactions.

Each factor $\psi_F(x_F)$ is usually defined as the exponential of the opposite of a dot product between the *factor parameter* θ_F and the *factor sufficient statistic* $\phi_F(x_F) = [\mathbb{1}_{x_F=y}]_{y\in\{0,1\}^F}$ where $\mathbb{1}_{x_F=y}$ is the indicator function that equals 1 if $x_F = y$ and 0 otherwise. The probability distribution is then equal to:

$$P(x = \bar{x}) = \frac{1}{Z}\prod_{f\in\mathcal{F}} exp(-\theta_F(h)^\top \phi_F(x_F)),$$

and the set $\theta = \{\theta_F \mid f \in \mathcal{F}\}$ is the MRF parameter set.

Our approach is to plug a MRF inside the learning scheme in order to obtain a better approximation of the probability distribution of the solutions to the input problem by taking variable correlations into account. The factor graph of this MRF is built upon the input problem and its parameters are predicted by a machine learning model from the embeddings outputted by the graph neural network. The probabilities for each variable are obtained by applying a marginal inference algorithm on the MRF like mean field or loopy belief propagation (Wainwright and Jordan, 2008). These probabilities take into account the interactions represented by the MRF. Figure 4 gives an overview of the whole machine learning framework to predict solutions.

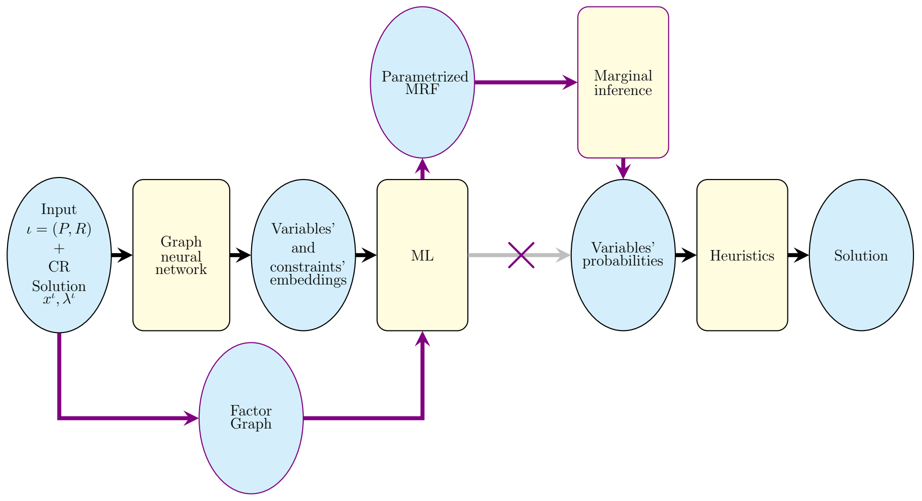

Figure 4: Our machine learning scheme to predict a solution to a binary linear problem. We depict in purple changes with respect to the baseline learning scheme.

Preliminary results on the maximum cardinality stable set problem show that adding a MRF inside the learning scheme improves the accuracy of the predictions and gives better solutions. However, this approach raises several questions.

How to build the MRF upon the input problem? A possibility is to consider as factor graph the bipartite representation of the input problem where constraints correspond to factors (Park and Shin, 2015). If this is possible for the stable set problem, since there are only two variables per constraint, this leads in the general case to MRF with factors defined on many variables. This implies too large factor parameters (with size up to 2^n if a constraint contains all variables) to deal with. One may approximate large factors by a set of small ones. But which constraints should be replaced and by which small factors?

Even if the MRF only contains small factors, the marginal inference is NP-hard in general. Inference heuristics like loopy belief propagation (Wainwright and Jordan, 2008) usually provide a good approximation but at the expense of extra computational burden. Should the factor graph be chosen to yield a polynomial inference problem, *e.g.,* be a tree? During training, may we approximate marginal inference with other methods (Wiseman and Kim, 2019; Kuck et al., 2020) to speed up training?

Finally, how sould we map the embeddings to the factor parameters? In our preliminary experiments, this question is simple. Indeed, we consider the same bipartite graph representation of a mixed-integer linear problem for the graph neural network (Gasse et al., 2019) and the factor graph. Hence, there is a bijection between graph neural network node embeddings and factor parameters. But how to do this mapping when both graphs differ? Several possibilities exist but thorough investigations have to be done.

Another interesting question links polyhedra and machine learning. Indeed, Chen, Liu, et al. (2023b) have shown that graph neural networks do not have enough separation power to predict the feasibility of a mixed-integer linear problem when only the objective function and the constraints are used to extract features (no feature associated with continuous relaxation). In other words, there exist a feasible input problem and an infeasible one, and any graph neural network will predict the same value (feasible or infeasible) for both problems. This does not happen for linear optimization problems (Chen, Liu, et al., 2023a). Since solving a mixed-integer linear problem is equivalent to solving a linear problem whose constraints correspond to the convex hull of its solutions, it is possible in theory to distinguish the feasibility of a mixed-integer linear problem using a graph neural network based on the graph representation of the linear description of its convex hull. Obviously, this is not possible to get such a graph neural network as it would contain in general an exponential number of nodes but even a partial description of the convex hull may increase its representation power. We plan to follow this approach by adding strengthening inequalities in our problem before predicting solutions. However, many questions arise: are strengthening inequalities useful from an experimental point of view to predict high quality solutions? Which kind of constraints should be added? How to handle these constraints in the MRF? More generally, how may machine learning take advantage from known polyhedral insights in order to predict solutions?

3 Mixed-Integer Linear Programming for Classification

Up to now, my works mixing machine learning and mixed-integer linear programming domains concern either the use of mixed-integer linear programming tools to formulate and solve the inference problem of some natural language processing task (Chapter 5), or the use of machine learning to enhance Lagrangian relaxation resolution (Chapter 6). A research question I really enjoy to investigate is the use of mixed-integer linear programming to design classification trees. This question is part of the Ph.D. thesis of Daniel Kopisitskiy who started in September 2024.

Consider a dataset $\mathcal{D} = \{(x^i, y^i), i = 1, \dots, n\}$ of n observations where $x^i \in \mathbb{R}^F$ is the feature vector of observation i, and $y^i \in \{0, \dots, L\}$ is its label. For ease of presentation, we consider a binary classification problem where each feature has a binary domain, that is, $x^i \in \{0,1\}^F$ and $y^i \in \{0,1\}$ for $i = 1, \dots, n$.

A *univariate classification tree* is an arborescence $D = (V, A)$ where each internal node is associated with a feature x_f, $f \in F$, and has two leaving arcs associated with value 0 and 1 respectively, and each leaf is associated with a label 0 or 1. Any feature vector $x \in \{0,1\}^F$ is associated with a path in the classification tree D from the root to a leaf. This path is given by choosing, starting from the root node, the arc leaving node v whose associated value equals x_f, where x_f, $f \in F$, is the feature associated with v. The label predicted by the classification tree for x is the label of the leaf in the corresponding path, see Figure 5 for an example.

Classification trees are widely used due to their interpretability as they can be easily understood by human beings, and to their ability to learn complex mappings. They constitute interesting models for explainable AI (Rudin, 2019).

Classification trees are usually constructed using heuristics such as CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993), see (Jena and Dehuri, 2020) for a review of the state-of-the-art heuristics. These heuristics locally optimize an objective that serves as a proxy for optimizing the global accuracy of the model. However, the design of classification trees may be posed as a combinatorial optimization problem. Given a dataset and a positive integer h, determine a classification tree of depth at most h which minimizes the number of misclassified observations. Such classification trees are called *optimal*. Experiments show that optimal classification trees provide better accuracy than the ones designed by heuristics, even on unseen observations (out-of-sample accuracy), that is on a test dataset different from the train dataset used for computing the optimal classification tree (Bertsimas and Dunn, 2017; Verwer and Zhang, 2017; Demirović et al., 2022; Linden et al., 2025). Moreover, such trees are usually much smaller than the greedy ones, increasing the interpretability of the model (Lin et al., 2020; Linden et al., 2025).

Features						Labels
x_1	x_2	x_3	x_4	x_5	x_6	y
0	1	0	1	1	0	0
0	0	0	1	0	1	0
0	0	1	1	1	0	0
1	0	1	1	0	1	0
0	1	1	0	1	0	1
0	0	1	0	0	1	1
1	0	0	1	1	0	1

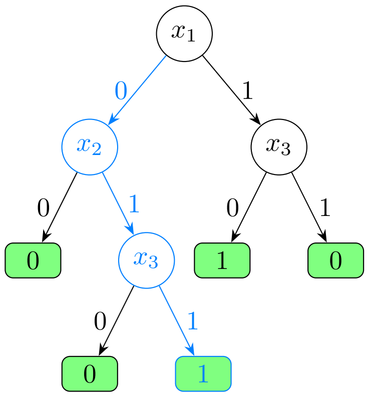

Figure 5: On the left, a dataset composed of seven observations is displayed. Each observation is defined by a binary feature vector of dimension 6 and a binary label. On the right, a univariate classification tree is displayed. Each internal node contains the feature used for splitting. The value of the feature is indicated on the leaving arcs. The leaves are depicted in green and contain their associated label inside. The fifth observation corresponds to the blue path in the classification tree so this latter predicts label 1 for this observation as it is the label of the leaf corresponding to the end node of the path. Since the label y^5 of the fifth observation is 1, the decision tree correctly classifies this observation.

Even if computing optimal classification trees is NP-hard (Hyafil and Rivest, 1976), several exact approaches have been designed (Demirović et al., 2022; Verhaeghe et al., 2020), especially using mixed-integer linear programming (Bertsimas and Dunn, 2017; Verwer and Zhang, 2019; Aghaei, Gómez, and Vayanos, 2024; Zhu et al., 2020), see (Carrizosa, Molero-Río, and Romero Morales, 2021) for more details on mathematical optimization for classification trees. Even if these models and algorithms do not yet scale to large datasets, several algorithms have been proposed which become more and more efficient.

Among them, the Benders decomposition developped by Aghaei, Gómez, and Vayanos (2024) seems to be well suited. Indeed, the authors formulate the computation of optimal classification trees as a multiflow problem: each correctly classified observation is a flow of value 1 from the root to a leaf, and the problem is to route the maximum amount of flow while the arc capacities are specified by the selected features for internal nodes and the selected labels for leaves. In their formulation, they consider binary variables for feature and label selection on nodes and continuous flow variables for each observation. In their Benders decomposition, the master problem is the design of the classification tree whereas the auxiliary problem reduces to computing a set of flows, and serves to count the number of correctly classified points for such a tree. The auxiliary problem decomposes per observation. For each observation, optimality cuts are inequalities associated with directed cuts in the tree and, by Menger's

Theorem (Menger, 1927), they ensure the existence of a flow if the observation is correctly classified. Hence, optimality cuts are separated observation by observation. The Benders algorithm improves the overall performance but gets stuck at some point due to the number of generated optimality cuts.

We propose to enhance the Benders decomposition of Aghaei, Gómez, and Vayanos (2024) by considering iterative surrogate problems obtained by partitioning observations into clusters and solving the problem considering each cluster as a single observation. At each iteration, some clusters have to be refined, that is, partitioned into several subclusters to obtain more accurate surrogate problems. A promising approach is to adapt the Benders adaptive-cuts method of Ramírez-Pico, Ljubić, and Moreno (2023) that has been designed for two-stage stochastic problems. In these problems, the number of scenarios is huge and classic Benders approaches generate either a cut per iteration by aggregating all scenarios (single-cut approach) or many cuts by separating Benders cuts scenario by scenario (multi-cut approach). Ramírez-Pico, Ljubić, and Moreno (2023) consider an adaptive-cuts approach where scenarios are dynamically partitioned into clusters, and the separation of Benders cuts is done cluster by cluster. Similar works considering partitioning scenarios for Benders decomposition exist for two-stage stochastic problems (Beltran-Royo, 2022; Biel and Johansson, 2020; Trukhanov, Ntaimo, and Schaefer, 2010; Vandenbussche et al., 2019). Note that a partition based approach has been considered for optimal classification trees by Ales, Huré, and Lambert (2024) but not coupled with a Benders decomposition.

In the Benders adaptive-cuts method (Aghaei, Gómez, and Vayanos, 2024), a core component is the way partitions are iteratively refined, and this has a dramatic impact on the performance of the algorithm. There exist generic algorithms (based on the values of dual variables in the auxiliary problem) but could we find a more efficient way to do it for optimal classification trees?

In their experiments, the authors consider, as initial partition, the partition with only one cluster containing all scenarios. For optimal classification trees, starting with such a partition may lead to several refinements increasing the computational time. Could we warmstart the algorithm by constructing a priori an initial partition? Ales, Huré, and Lambert (2024) propose different algorithms for partitioning the observations. Are their methods efficient in that case? Do we need to enhance this method with other algorithmic Benders tricks (Rahmaniani et al., 2017) like stabilization or cut selection/removal? More generally, how to improve this algorithm to scale it to large datasets?

Several variants of optimal classification trees exist: multivariate optimal classification trees (Bertsimas and Dunn, 2017) where a linear combination of features is used for splittings, or optimal classification model trees where the linear combination is done at each leaf (Roselli and Frank, 2025). Moreover, different requirements, for instance arising from social sciences, may be added like fairnees (Jo et al., 2023). Could we reshape our Benders adaptive-cuts algorithm to adapt it to those variants?

Bibliography

Aghaei, Sina, Andrés Gómez, and Phebe Vayanos (July 31, 2024). "Strong Optimal Classification Trees". In: *Operations Research*, opre.2021.0034. ISSN: 0030-364X, 1526-5463. DOI: `10.1287/opre.2021.0034`. URL: `https://pubsonline.informs.org/doi/10.1287/opre.2021.0034` (visited on 06/15/2025).

Ales, Zacharie, Valentine Huré, and Amélie Lambert (May 2024). "Clustering data for the Optimal Classication Tree Problem". working paper or preprint. URL: `https://hal.science/hal-04589656`.

Applegate, David L., Robert E. Bixby, Vašek Chvatál, and William J. Cook (2006). *The Traveling Salesman Problem: A Computational Study.* Princeton University Press. ISBN: 9780691129938. URL: `http://www.jstor.org/stable/j.ctt7s8xg` (visited on 08/20/2024).

Ba, Jimmy, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). "Layer Normalization". In: *ArXiv* abs/1607.06450. URL: `https://api.semanticscholar.org/CorpusID:8236317`.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473. URL: `https://api.semanticscholar.org/CorpusID:11212020`.

Balas, Egon (Dec. 1998). "Disjunctive programming: Properties of the convex hull of feasible points". In: *Discrete Applied Mathematics* 89.1, pp. 3–44. ISSN: 0166218X. DOI: `10.1016/S0166-218X(98)00136-X`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0166218X9800136X` (visited on 09/18/2024).

Barahona, Francisco (June 1993). "On cuts and matchings in planar graphs". In: *Mathematical Programming* 60.1, pp. 53–68. ISSN: 0025-5610, 1436-4646. DOI: `10.1007/BF01580600`. URL: `http://link.springer.com/10.1007/BF01580600` (visited on 08/22/2024).

Barahona, Francisco and Ali Ridha Mahjoub (June 1986). "On the cut polytope". In: *Mathematical Programming* 36.2, pp. 157–173. ISSN: 0025-5610, 1436-4646. DOI: `10.1007/BF02592023`. URL: `http://link.springer.com/10.1007/BF02592023` (visited on 08/22/2024).

Barbato, Michele, Roland Grappe, Mathieu Lacroix, and Emiliano Lancini (Jan. 2023). "Box-total dual integrality and edge-connectivity". en. In: *Mathe-*

matical Programming 197.1, pp. 307–336. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-021-01743-x. URL: https://link.springer.com/10.1007/s10107-021-01743-x (visited on 05/03/2024).

Barbato, Michele, Roland Grappe, Mathieu Lacroix, Emiliano Lancini, and Roberto Wolfler Calvo (Feb. 2022). "The Schrijver system of the flow cone in series–parallel graphs". en. In: *Discrete Applied Mathematics* 308, pp. 162–167. ISSN: 0166218X. DOI: 10.1016/j.dam.2020.03.054. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166218X20301530 (visited on 12/29/2021).

Barbato, Michele, Roland Grappe, Mathieu Lacroix, and Clément Pira (May 2018). "Lexicographical polytopes". en. In: *Discrete Applied Mathematics* 240, pp. 3–7. ISSN: 0166218X. DOI: 10.1016/j.dam.2017.04.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166218X17301889 (visited on 05/03/2024).

Beltran-Royo, C. (Jan. 2, 2022). "Fast Scenario Reduction by Conditional Scenarios in Two-Stage Stochastic MILP Problems". In: *Optimization Methods and Software* 37.1, pp. 23–44. ISSN: 1055-6788, 1029-4937. DOI: 10.1080/10556788.2019.1697696. URL: https://www.tandfonline.com/doi/full/10.1080/10556788.2019.1697696 (visited on 06/21/2025).

Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost (2021). "Machine learning for combinatorial optimization: a methodological tour d'horizon". In: *European Journal of Operational Research* 290.2, pp. 405–421.

Bertsekas, Dimitri P. (2016). *Nonlinear Programming*. 3rd ed. Belmont, Mass: Athena scientific. ISBN: 978-1-886529-05-2.

Bertsimas, Dimitris and Jack Dunn (July 2017). "Optimal Classification Trees". In: *Machine Learning* 106.7, pp. 1039–1082. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-017-5633-9. URL: http://link.springer.com/10.1007/s10994-017-5633-9 (visited on 02/06/2024).

Biel, Martin and Mikael Johansson (Oct. 2, 2020). *Dynamic Cut Aggregation in L-shaped Algorithms*. DOI: 10.48550/arXiv.1910.13752. arXiv: 1910.13752 [math]. URL: http://arxiv.org/abs/1910.13752 (visited on 06/21/2025). Pre-published.

Bondy, J. A. and U. S. R. Murty (2008). *Graph Theory*. Red. by S. Axler and K. A. Ribet. Vol. 244. Graduate Texts in Mathematics. London: Springer London. ISBN: 978-1-84628-969-9 978-1-84628-970-5. DOI: 10.1007/978-1-84628-970-5. URL: http://link.springer.com/10.1007/978-1-84628-970-5 (visited on 09/30/2024).

Borne, Sylvie, Pierre Fouilhoux, Roland Grappe, Mathieu Lacroix, and Pierre Pesneau (2015). "Circuit and bond polytopes on series–parallel graphs". In: *Discrete Optimization* 17, pp. 55–68. ISSN: 1572-5286. DOI: http://dx.doi.org/10.1016/j.disopt.2015.04.001. URL: http://www.sciencedirect.com/science/article/pii/S1572528615000201.

Bottou, Léon and Olivier Bousquet (2007). "The Tradeoffs of Large Scale Learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. Curran Associates, Inc.

URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf.

Bottou, Léon and Olivier Bousquet (Sept. 30, 2011). "The Tradeoffs of Large-Scale Learning". In: *Optimization for Machine Learning*. Ed. by Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. The MIT Press, pp. 351–368. ISBN: 978-0-262-29877-3. DOI: 10.7551/mitpress/8996.003.0015. URL: https://direct.mit.edu/books/book/2929/chapter/79595/The-Tradeoffs-of-Large-Scale-Learning (visited on 06/24/2025).

Bragin, Mikhail A. (Feb. 2024). "Survey on Lagrangian relaxation for MILP: importance, challenges, historical review, recent advancements, and opportunities". en. In: *Annals of Operations Research* 333.1. Publisher: Springer Science and Business Media LLC, pp. 29–45. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-023-05499-9. URL: https://link.springer.com/10.1007/s10479-023-05499-9 (visited on 01/14/2025).

Breiman, L., Jerome H. Friedman, Richard A. Olshen, and C. J. Stone (1984). "Classification and Regression Trees". In: Taylor & Francis. ISBN: 978-0-412-04841-8.

Carrizosa, Emilio, Cristina Molero-Río, and Dolores Romero Morales (Apr. 2021). "Mathematical Optimization in Classification and Regression Trees". In: *TOP* 29.1, pp. 5–33. ISSN: 1134-5764, 1863-8279. DOI: 10.1007/s11750-021-00594-1. URL: http://link.springer.com/10.1007/s11750-021-00594-1 (visited on 03/26/2024).

Chakrabarti, Amit, Lisa Fleischer, and Christophe Weibel (2012). "When the cut condition is enough, a complete characterization for multiflow problems in series-parallel networks." In: *Proceedings of the 44th annual ACM symposium on theory of computing, STOC 2012. New York, NY, USA, May 19–22, 2012.* ISBN: 978-1-4503-1245-5, pp. 19–26. ISSN: 07378017. DOI: 10.1145/2213977.2213980.

Chartrand, Gary and Frank Harary (1967). "Planar Permutation Graphs". en. In: *Annales de l'institut Henri Poincaré. Section B. Calcul des probabilités et statistiques* 3.4, pp. 433–438. URL: http://www.numdam.org/item/AIHPB_1967__3_4_433_0/.

Chen, Xujin, Guoli Ding, and Wenan Zang (2009). "The box-TDI system associated with 2-edge connected spanning subgraphs". In: *Discrete Applied Mathematics* 157.1. Publisher: Elsevier B.V., pp. 118–125. ISSN: 0166218X. DOI: 10.1016/j.dam.2008.05.001. URL: http://linkinghub.elsevier.com/retrieve/pii/S0166218X08002072.

Chen, Ziang, Jialin Liu, Xinshang Wang, and Wotao Yin (2023a). "On Representing Linear Programs by Graph Neural Networks". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net. URL: https://openreview.net/forum?id=cP2QVK-uygd.

— (2023b). "On Representing Mixed-Integer Linear Programs by Graph Neural Networks". In: *The Eleventh International Conference on Learning Representations.* URL: https://openreview.net/forum?id=4gc3MGZra1d.

Cheney, E. W. and A. A. Goldstein (Dec. 1959). “Newton’s method for convex programming and Tchebycheff approximation”. en. In: *Numerische Mathematik* 1.1, pp. 253–268. ISSN: 0029-599X, 0945-3245. DOI: 10.1007/BF01386389. URL: http://link.springer.com/10.1007/BF01386389 (visited on 01/23/2025).

Chervet, Patrick, Roland Grappe, Mathieu Lacroix, Francesco Pisanu, and Roberto Wolfler Calvo (Nov. 2023). “Hard problems on box-totally dual integral polyhedra”. en. In: *Discrete Optimization* 50, p. 100810. ISSN: 15725286. DOI: 10.1016/j.disopt.2023.100810. URL: https://linkinghub.elsevier.com/retrieve/pii/S157252862300052X (visited on 05/03/2024).

Chervet, Patrick, Roland Grappe, and Louis-Hadrien Robert (July 2021). “Box-total dual integrality, box-integrality, and equimodular matrices”. In: *Mathematical Programming* 188.1, pp. 319–349. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-020-01514-0. URL: https://link.springer.com/10.1007/s10107-020-01514-0 (visited on 09/14/2022).

Chimani, Markus, Martina Juhnke-Kubitzke, and Alexander Nover (Oct. 26, 2023). “On the bond polytope”. In: *Advances in Geometry* 23.4, pp. 461–480. ISSN: 1615-7168, 1615-715X. DOI: 10.1515/advgeom-2023-0014. URL: https://www.degruyter.com/document/doi/10.1515/advgeom-2023-0014/html (visited on 07/17/2024).

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014). “Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179/.

Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Chvátal, V (Apr. 1975). “On Certain Polytopes Associated with Graphs”. In: *Journal of Combinatorial Theory, Series B* 18.2, pp. 138–154. ISSN: 00958956. DOI: 10.1016/0095-8956(75)90041-6. URL: https://linkinghub.elsevier.com/retrieve/pii/0095895675900416 (visited on 06/04/2025).

Collins, Michael (2002). “Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms”. In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP ’02. USA: Association for Computational

Linguistics, pp. 1–8. DOI: 10.3115/1118693.1118694. URL: https://doi.org/10.3115/1118693.1118694.

Conforti, Michele, Gérard Cornuéjols, and Giacomo Zambelli (2010). "Polyhedral Approaches to Mixed Integer Linear Programming". In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Ed. by Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 343–385. ISBN: 978-3-540-68279-0. DOI: 10.1007/978-3-540-68279-0_11. URL: https://doi.org/10.1007/978-3-540-68279-0_11.

— (Apr. 2013). "Extended formulations in combinatorial optimization". In: *Annals of Operations Research* 204.1, pp. 97–143. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-012-1269-0. URL: http://link.springer.com/10.1007/s10479-012-1269-0 (visited on 08/22/2024).

Cook, William (1986). "On box totally dual integral polyhedra". In: *Mathematical Programming* 34.1, pp. 48–61. ISSN: 00255610. DOI: 10.1007/BF01582162.

Cornaz, Denis, Roland Grappe, and Mathieu Lacroix (2019). "Trader multiflow and box-TDI systems in series–parallel graphs". In: *Discrete Optimization* 31. ISSN: 15725286. DOI: 10.1016/j.disopt.2018.09.003.

Corro, Caio, Joseph Le Roux, and Mathieu Lacroix (2017). "Efficient Discontinuous Phrase-Structure Parsing via the Generalized Maximum Spanning Arborescence". en. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1644–1654. DOI: 10.18653/v1/D17-1172. URL: http://aclweb.org/anthology/D17-1172 (visited on 05/03/2024).

Corro, Caio, Joseph Le Roux, Mathieu Lacroix, Antoine Rozenknop, and Roberto Wolfler Calvo (2016). "Dependency Parsing with Bounded Block Degree and Well-nestedness via Lagrangian Relaxation and Branch-and-Bound". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ISBN: 9781510827585, pp. 355–366. DOI: 10.18653/v1/P16-1034. URL: http://www.aclweb.org/anthology/P16-1034.

Corso, Gabriele, Hannes Stark, Stefanie Jegelka, Tommi Jaakkola, and Regina Barzilay (Mar. 2024). "Graph neural networks". en. In: *Nature Reviews Methods Primers* 4.1, p. 17. ISSN: 2662-8449. DOI: 10.1038/s43586-024-00294-7. URL: https://www.nature.com/articles/s43586-024-00294-7 (visited on 03/21/2025).

Cunningham, W. H. and A. B. Marsh (1978). "A primal algorithm for optimum matching". In: *Polyhedral Combinatorics: Dedicated to the memory of D.R. Fulkerson*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 50–72.

Dantzig, G., R. Fulkerson, and S. Johnson (1954). "Solution of a Large-Scale Traveling-Salesman Problem". In: *Journal of the Operations Research Society of America* 2.4. Publisher: INFORMS, pp. 393–410. ISSN: 00963984. URL: http://www.jstor.org/stable/166695 (visited on 01/14/2025).

Demelas, Francesca, Joseph Le Roux, Mathieu Lacroix, and Axel Parmentier (2024). "Predicting Lagrangian Multipliers for Mixed Integer Linear Programs". In: *Forty-first International Conference on Machine Learning*. URL: `https://openreview.net/forum?id=aZnZOqUOHq`.

Demirović, Emir, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J. Stuckey (2022). "MurTree: Optimal Decision Trees via Dynamic Programming and Search". In: *Journal of Machine Learning Research* 23.26, pp. 1–47. URL: `http://jmlr.org/papers/v23/20-520.html`.

Dey, Santanu S. and Marco Molinaro (July 2018). "Theoretical challenges towards cutting-plane selection". en. In: *Mathematical Programming* 170.1, pp. 237–266. ISSN: 0025-5610, 1436-4646. DOI: `10.1007/s10107-018-1302-4`. URL: `http://link.springer.com/10.1007/s10107-018-1302-4` (visited on 09/26/2023).

Deza, Arnaud and Elias B. Khalil (2023). "Machine Learning for Cutting Planes in Integer Programming: A Survey". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. Macau, SAR China: International Joint Conferences on Artificial Intelligence Organization, pp. 6592–6600. DOI: `10.24963/ijcai.2023/739`.

Didi Biha, Mohamed and Ali Ridha Mahjoub (1996). "k-edge connected polyhedra on series-parallel graphs". In: *Operations research letters* 19.2, pp. 71–78.

Ding, Guoli, Li Feng, and Wenan Zang (Aug. 2008). "The complexity of recognizing linear systems with certain integrality properties". In: *Mathematical Programming* 114.2, pp. 321–334. ISSN: 0025-5610, 1436-4646. DOI: `10.1007/s10107-007-0103-y`. URL: `http://link.springer.com/10.1007/s10107-007-0103-y` (visited on 06/01/2021).

Ding, Guoli, Lei Tan, and Wenan Zang (2018). "When Is the Matching Polytope Box-Totally Dual Integral?" In: *Mathematics of Operations Research* 43.1, pp. 64–99.

Ding, Jian Ya, Chao Zhang, Lei Shen, Shengyin Li, Bing Wang, Yinghui Xu, and Le Song (2020). "Accelerating primal solution findings for mixed integer programs based on solution prediction". In: *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence* 2017.miplib2017. ISBN: 9781577358350, pp. 1452–1459. ISSN: 2159-5399.

Dong, Yuxiao, Reid A. Johnson, and Nitesh V. Chawla (2015). "Will This Paper Increase Your h-index? Scientific Impact Prediction". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. event-place: Shanghai, China. New York, NY, USA: Association for Computing Machinery, pp. 149–158. ISBN: 978-1-4503-3317-7. DOI: `10.1145/2684822.2685314`. URL: `https://doi.org/10.1145/2684822.2685314`.

Drozdov, Andrew, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou (2023). "Compositional Semantic Parsing with Large Language Models". In: *The Eleventh*

International Conference on Learning Representations. URL: https://openreview.net/forum?id=gJW8hSGBys8.

Duffin, R J (1965). “Topology of series-parallel networks”. In: *Journal of Mathematical Analysis and Applications* 10.2, pp. 303–318. ISSN: 0022-247X. DOI: 10.1016/0022-247X(65)90125-3. URL: http://www.sciencedirect.com/science/article/pii/0022247X65901253.

Dupont Bouillard, Alexandre (May 2024). “Co-k-plexes et coloration k-defective : polyèdres et algorithmes”. Theses. Université Paris-Nord - Paris XIII. URL: https://theses.hal.science/tel-04672951.

Dupont-Bouillard, Alexandre, Pierre Fouilhoux, Roland Grappe, and Mathieu Lacroix (2024). *Contractions in perfect graph*. arXiv: 2401.12793 [math.CO].

Edmonds, J. (Jan. 1965). “Maximum matching and a polyhedron with 0,1-vertices”. In: *Journal of Research of the National Bureau of Standards* 69B.

Edmonds, J., W. R. Pulleyblank, and L. Lovász (Sept. 1982). “Brick Decompositions and the Matching Rank of Graphs”. In: *Combinatorica* 2.3, pp. 247–274. ISSN: 0209-9683, 1439-6912. DOI: 10.1007/BF02579233. URL: http://link.springer.com/10.1007/BF02579233 (visited on 06/14/2025).

Edmonds, Jack and Rick Giles (1977). “A Min-Max Relation for Submodular Functions on Graphs”. In: *Annals of Discrete Mathematics* 1, pp. 185–204. ISSN: 0167-5060. DOI: 10.1016/S0167-5060(08)70734-9. URL: http://www.sciencedirect.com/science/article/pii/S0167506008707349.

Eisner, Jason (June 1996). “Efficient Normal-Form Parsing for Combinatory Categorial Grammar”. In: *34th Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, California, USA: Association for Computational Linguistics, pp. 79–86. DOI: 10.3115/981863.981874. URL: https://aclanthology.org/P96-1011/.

— (1997). “Bilexical Grammars and a Cubic-time Probabilistic Parser”. In: *Proceedings of the Fifth International Workshop on Parsing Technologies*. Ed. by Anton Nijholt et al. Boston/Cambridge, Massachusetts, USA: Association for Computational Linguistics, pp. 54–65. URL: https://aclanthology.org/1997.iwpt-1.10/.

Ford, L. R. and D. R. Fulkerson (1956). “Maximal Flow Through a Network”. In: *Canadian Journal of Mathematics* 8, pp. 399–404.

Fourier, J. B. Joseph (1826). “Solution d’une question particulière du calcul des inégalités”. In: *Nouveau bulletin des sciences par la société philomatique de Paris*, p. 99.

Frangioni, Antonio (Nov. 1996). “Solving semidefinite quadratic problems within nonsmooth optimization algorithms”. en. In: *Computers & Operations Research* 23.11. Publisher: Elsevier BV, pp. 1099–1118. ISSN: 0305-0548. DOI: 10.1016/0305-0548(96)00006-8. URL: https://linkinghub.elsevier.com/retrieve/pii/0305054896000068 (visited on 09/17/2024).

Gasse, Maxime, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi (2019). “Exact Combinatorial Optimization with Graph Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*.

Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc.

Geifman, Yonatan and Ran El-Yaniv (June 9–15, 2019). "SelectiveNet: A Deep Neural Network with an Integrated Reject Option". In: *Proceedings of the 36th International Conference on Machine Learning.* Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2151–2159. URL: `https://proceedings.mlr.press/v97/geifman19a.html`.

Geoffrion, A. M. (1974). "Lagrangean relaxation for integer programming". In: *Approaches to Integer Programming.* Ed. by M. L. Balinski. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 82–114. ISBN: 978-3-642-00740-8.

Giles, Frederick Richard and William R Pulleyblank (1979). "Total dual integrality and integer polyhedra". In: *Linear algebra and its applications* 25. Publisher: Elsevier, pp. 191–196.

Gómez-Rodríguez, Carlos, David Weir, and John Carroll (2009). "Parsing Mildly Non-Projective Dependency Structures". In: *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009).* Ed. by Alex Lascarides, Claire Gardent, and Joakim Nivre. Athens, Greece: Association for Computational Linguistics, pp. 291–299. URL: `https://aclanthology.org/E09-1034/`.

Grappe, Roland and Mathieu Lacroix (May 2018). "The st-bond polytope on series-parallel graphs". In: *RAIRO - Operations Research.* ISSN: 0399-0559. DOI: `10.1051/ro/2018035`. URL: `https://www.rairo-ro.org/10.1051/ro/2018035`.

Gregor, Karol and Yann LeCun (2010). "Learning Fast Approximations of Sparse Coding". In: *International Conference on Machine Learning.* URL: `https://api.semanticscholar.org/CorpusID:13333501`.

Gupte, Akshay (Mar. 2016). "Convex hulls of superincreasing knapsacks and lexicographic orderings". In: *Discrete Applied Mathematics* 201, pp. 150–163. ISSN: 0166218X. DOI: `10.1016/j.dam.2015.08.010`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0166218X15004059` (visited on 08/26/2024).

Havelka, Jiří (Apr. 2007). "Relationship between Non-Projective Edges, Their Level Types, and Well-Nestedness". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers.* Ed. by Candace Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai. Rochester, New York: Association for Computational Linguistics, pp. 61–64. URL: `https://aclanthology.org/N07-2016/`.

He, Guoxiu, Zhikai Xue, Zhuoren Jiang, Yangyang Kang, Star Zhao, and Wei Lu (2023). "H2CGL: Modeling dynamics of citation network for impact prediction". In: *Information Processing & Management* 60.6, p. 103512. ISSN: 0306-4573. DOI: `https://doi.org/10.1016/j.ipm.2023.103512`. URL: `https://www.sciencedirect.com/science/article/pii/S0306457323002492`.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet

Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (June 2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '16. IEEE, pp. 770–778.

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 1997). "Long Short-Term Memory". en. In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. URL: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (visited on 03/21/2025).

Hoffman, Alan J. and Joseph B. Kruskal (2010). "Integral Boundary Points of Convex Polyhedra". In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 49–76.

Huang, Lingying, Xiaomeng Chen, Wei Huo, Jiazheng Wang, Fan Zhang, Bo Bai, and Ling Shi (2022). "Improving Primal Heuristics for Mixed Integer Programming Problems Based on Problem Reduction: A Learning-Based Approach". In: *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 181–186. DOI: 10.1109/ICARCV57592.2022.10004252.

Hyafil, Laurent and Ronald L. Rivest (May 1976). "Constructing Optimal Binary Decision Trees Is NP-complete". In: *Information Processing Letters* 5.1, pp. 15–17. ISSN: 00200190. DOI: 10.1016/0020-0190(76)90095-8. URL: https://linkinghub.elsevier.com/retrieve/pii/0020019076900958 (visited on 06/15/2025).

Innes, Michael, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah (2018). "Fashionable Modelling with Flux". In: *CoRR* abs/1811.01457. arXiv: 1811.01457. URL: https://arxiv.org/abs/1811.01457.

Innes, Mike (2018). "Flux: Elegant Machine Learning with Julia". In: *Journal of Open Source Software*. DOI: 10.21105/joss.00602.

Jena, Monalisa and Satchidananda Dehuri (Dec. 15, 2020). "DecisionTree for Classification and Regression: A State-of-the Art Review". In: *Informatica* 44.4. ISSN: 1854-3871, 0350-5596. DOI: 10.31449/inf.v44i4.3023. URL: http://www.informatica.si/index.php/informatica/article/view/3023 (visited on 06/07/2025).

Jo, Nathanael, Sina Aghaei, Jack Benson, Andres Gomez, and Phebe Vayanos (2023). "Learning Optimal Fair Decision Trees: Trade-offs between Interpretability, Fairness, and Accuracy". In: *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*. Aies '23. Montréal, QC, Canada and New York, NY, USA: Association for Computing Machinery, pp. 181–192. ISBN: 979-8-4007-0231-0. DOI: 10.1145/3600211.3604664. URL: https://doi.org/10.1145/3600211.3604664.

Jurafsky, Daniel and James H. Martin (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics,* *and Speech Recognition with Language Models*. 3rd. URL: https://web.stanford.edu/~jurafsky/slp3/.

Kalchbrenner, Nal and Phil Blunsom (2013). "Recurrent Continuous Translation Models". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A Meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, pp. 1700–1709. DOI: 10.18653/V1/D13-1176. URL: https://doi.org/10.18653/v1/d13-1176.

Karp, Richard M. (1972). "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, pp. 85–103.

Kelley Jr., J. E. (Dec. 1960). "The Cutting-Plane Method for Solving Convex Programs". en. In: *Journal of the Society for Industrial and Applied Mathematics* 8.4, pp. 703–712. ISSN: 0368-4245, 2168-3484. DOI: 10.1137/0108053. URL: http://epubs.siam.org/doi/10.1137/0108053 (visited on 01/14/2025).

Kim, Yoon, Carl Denton, Luong Hoang, and Alexander M. Rush (2017). "Structured Attention Networks". In: *ArXiv* abs/1702.00887. URL: https://api.semanticscholar.org/CorpusID:6961760.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. URL: https://api.semanticscholar.org/CorpusID:6628106.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv: http://arxiv.org/abs/1312.6114v10.

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the 5th International Conference on Learning Representations*. ICLR '17. Place: Palais des Congrès Neptune, Toulon, France. URL: https://openreview.net/forum?id=SJU4ayYgl.

Kiwiel, Krzysztof C. (1986). "A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization". en. In: *IMA Journal of Numerical Analysis* 6.2, pp. 137–152. ISSN: 0272-4979, 1464-3642. DOI: 10.1093/imanum/6.2.137. URL: https://academic.oup.com/imajna/article-lookup/doi/10.1093/imanum/6.2.137 (visited on 01/28/2025).

— (Jan. 1989). "A Dual Method for Certain Positive Semidefinite Quadratic Programming Problems". en. In: *SIAM Journal on Scientific and Statistical Computing* 10.1, pp. 175–186. ISSN: 0196-5204, 2168-3417. DOI: 10.1137/0910013. URL: http://epubs.siam.org/doi/10.1137/0910013 (visited on 01/28/2025).

Kuck, Jonathan, Shuvam Chakraborty, Hao Tang, Rachel Luo, Jiaming Song, Ashish Sabharwal, and Stefano Ermon (2020). "Belief Propagation Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 667–678. URL: https:

//proceedings.neurips.cc/paper_files/paper/2020/file/07217414eb3fbe24d4e5b6cafb91ca18-Paper.pdf.

Lancini, Emiliano (2019). “TDIness and Multicuts”. 2019PA131083. PhD thesis. Université Sorbonne Paris Nord. URL: http://www.theses.fr/2019PA131083/document.

Le Cun, Yann, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang (2006). “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0.

Lemaréchal, Claude (2001). “Lagrangian Relaxation”. en. In: *Computational Combinatorial Optimization*. Ed. by Michael Jünger and Denis Naddef. Vol. 2241. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 112–156. ISBN: 978-3-540-42877-0 978-3-540-45586-8. DOI: 10.1007/3-540-45586-8_4. URL: http://link.springer.com/10.1007/3-540-45586-8_4 (visited on 09/20/2024).

Lin, Jimmy, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer (2020). “Generalized and Scalable Optimal Sparse Decision Trees”. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. JMLR.org.

Linden, Jacobus G. M. van der, Daniël Vos, Mathijs M. de Weerdt, Sicco Verwer, and Emir Demirović (Apr. 1, 2025). *Optimal or Greedy Decision Trees? Revisiting Their Objectives, Tuning, and Performance*. DOI: 10.48550/arXiv.2409.12788. arXiv: 2409.12788 [cs]. URL: http://arxiv.org/abs/2409.12788 (visited on 06/07/2025). Pre-published.

Lucchesi, Cláudio L. and U.S.R. Murty (2024). *Perfect Matchings: A Theory of Matching Covered Graphs*. Vol. 31. Algorithms and Computation in Mathematics. Cham: Springer Nature Switzerland. ISBN: 978-3-031-47503-0 978-3-031-47504-7. DOI: 10.1007/978-3-031-47504-7. URL: https://link.springer.com/10.1007/978-3-031-47504-7 (visited on 10/01/2024).

Lucena, Abilio (Nov. 2005). “Non Delayed Relax-and-Cut Algorithms”. en. In: *Annals of Operations Research* 140.1, pp. 375–410. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-005-3977-1. URL: http://link.springer.com/10.1007/s10479-005-3977-1 (visited on 01/14/2025).

— (2006). “Lagrangian Relax-and-Cut Algorithms”. en. In: *Handbook of Optimization in Telecommunications*. Ed. by Mauricio G. C. Resende and Panos M. Pardalos. Boston, MA: Springer US, pp. 129–145. ISBN: 978-0-387-30662-9 978-0-387-30165-5. DOI: 10.1007/978-0-387-30165-5_5. URL: http://link.springer.com/10.1007/978-0-387-30165-5_5 (visited on 10/28/2024).

Marchand, Hugues, Alexander Martin, Robert Weismantel, and Laurence Wolsey (Nov. 2002). “Cutting planes in integer and mixed integer programming”. en. In: *Discrete Applied Mathematics* 123.1-3, pp. 397–446. ISSN: 0166218X. DOI: 10.1016/S0166-218X(01)00348-1. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166218X01003481 (visited on 01/14/2025).

Martins, Andre and Ramon Astudillo (June 2016). “From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by

Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1614–1623. URL: https://proceedings.mlr.press/v48/martins16.html.

McDonald, Ryan, Koby Crammer, and Fernando Pereira (June 2005). "Online Large-Margin Training of Dependency Parsers". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ed. by Kevin Knight, Hwee Tou Ng, and Kemal Oflazer. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 91–98. DOI: 10.3115/1219840.1219852. URL: https://aclanthology.org/P05-1012/.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič (2005). "Non-projective dependency parsing using spanning tree algorithms". In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 523–530.

Menger, Karl (1927). "Zur allgemeinen Kurventheorie". ger. In: *Fundamenta Mathematicae* 10.1, pp. 96–115. URL: http://eudml.org/doc/211191.

Mitchell, John E. (2009). "Integer programming: branch and cut algorithms". In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, pp. 1643–1650. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_287. URL: https://doi.org/10.1007/978-0-387-74759-0_287.

Möhl, Mathias (2006). "Drawings as Models of Syntactic Structure: Theory and Algorithms". Diploma thesis. Programming Systems Lab, Universit"at des Saarlandes, Saarbr"ucken.

Monga, Vishal, Yuelong Li, and Yonina C. Eldar (Mar. 2021). "Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing". In: *IEEE Signal Processing Magazine* 38.2, pp. 18–44. ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2020.3016905. URL: https://ieeexplore.ieee.org/document/9363511/ (visited on 05/13/2025).

Motie, Soroor and Bijan Raahemi (2024). "Financial fraud detection using graph neural networks: A systematic review". In: *Expert Systems with Applications* 240, p. 122156. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.122156. URL: https://www.sciencedirect.com/science/article/pii/S0957417423026581.

Murphy, Kevin P. (2013). *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press. ISBN: 9780262018029 0262018020.

Nair, Vinod et al. (2020). "Solving Mixed Integer Programs Using Neural Networks". In: *CoRR* abs/2012.13349. arXiv: 2012.13349.

Nivre, Joakim (Apr. 2003). "An Efficient Algorithm for Projective Dependency Parsing". In: *Proceedings of the Eighth International Conference on Parsing Technologies*. Nancy, France, pp. 149–160. URL: https://aclanthology.org/W03-3017/.

— (Apr. 2006). "Constraints on Non-Projective Dependency Parsing". In: *11th Conference of the European Chapter of the Association for Computational Linguistics*. Ed. by Diana McCarthy and Shuly Wintner. Trento, Italy: Association for Computational Linguistics, pp. 73–80. URL: https://aclanthology.org/E06-1010/.

Park, Sejun and Jinwoo Shin (2015). “Max-Product Belief Propagation for Linear Programming: Applications to Combinatorial Optimization”. In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015, July 12-16, 2015, Amsterdam, the Netherlands.* Ed. by Marina Meila and Tom Heskes. AUAI Press, pp. 662–671. URL: http://auai.org/uai2015/proceedings/papers/106.pdf.

Paszke, Adam et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32.* Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pinheiro Cinelli, Lucas, Matheus Araújo Marins, Eduardo Antúnio Barros Da Silva, and Sérgio Lima Netto (2021). “Variational Autoencoder”. en. In: *Variational Methods for Machine Learning with Applications to Deep Networks.* Cham: Springer International Publishing, pp. 111–149. ISBN: 978-3-030-70678-4 978-3-030-70679-1. DOI: 10.1007/978-3-030-70679-1_5. URL: https://link.springer.com/10.1007/978-3-030-70679-1_5 (visited on 04/01/2025).

Pisanu, Francesco (Dec. 2023). “Sur la box-total dual integralité et la totale équimodularité”. Theses. Université Paris-Nord - Paris XIII. URL: https://theses.hal.science/tel-04457268.

Poljak, Boris T. (1987). *Introduction to optimization.* eng. Translations Series in mathematics and engineering. New York: Optimization Software. ISBN: 978-0-911575-14-9.

Prince, Simon J.D. (2023). *Understanding Deep Learning.* The MIT Press. URL: http://udlbook.com.

Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 1-55860-238-0. URL: http://portal.acm.org/citation.cfm?id=152181.

Rahmaniani, Ragheb, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei (2017). “The Benders Decomposition Algorithm: A Literature Review”. In: *European Journal of Operational Research* 259.3, pp. 801–817. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2016.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0377221716310244.

Ramírez-Pico, Cristian, Ivana Ljubić, and Eduardo Moreno (Sept. 2023). “Benders Adaptive-Cuts Method for Two-Stage Stochastic Programs”. In: *Transportation Science* 57.5, pp. 1252–1275. ISSN: 0041-1655, 1526-5447. DOI: 10.1287/trsc.2022.0073. URL: https://pubsonline.informs.org/doi/10.1287/trsc.2022.0073 (visited on 06/21/2025).

Roselli, Sabino Francesco and Eibe Frank (Mar. 17, 2025). *Experiments with Optimal Model Trees.* DOI: 10.48550/arXiv.2503.12902. arXiv: 2503.12902 [cs]. URL: http://arxiv.org/abs/2503.12902 (visited on 03/20/2025). Pre-published.

Rosenblatt, F. (1958). “The perceptron: A probabilistic model for information storage and organization in the brain.” en. In: *Psychological Review* 65.6, pp. 386–408. ISSN: 1939-1471, 0033-295X. DOI: `10.1037/h0042519`. URL: `https://doi.apa.org/doi/10.1037/h0042519` (visited on 03/20/2025).

Rudin, Cynthia (May 13, 2019). “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead”. In: *Nature Machine Intelligence* 1.5, pp. 206–215. ISSN: 2522-5839. DOI: `10.1038/s42256-019-0048-x`. URL: `https://www.nature.com/articles/s42256-019-0048-x` (visited on 06/15/2025).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (Oct. 1986). “Learning representations by back-propagating errors”. en. In: *Nature* 323.6088, pp. 533–536. ISSN: 0028-0836, 1476-4687. DOI: `10.1038/323533a0`. URL: `https://www.nature.com/articles/323533a0` (visited on 03/19/2025).

Rumelhart, David E. and James L. McClelland (1987). “Learning Internal Representations by Error Propagation”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pp. 318–362.

Scavuzzo, Lara, Karen Aardal, Andrea Lodi, and Neil Yorke-Smith (2024). “Machine learning augmented branch and bound for mixed integer linear programming”. In: *Mathematical Programming.* DOI: `10.1007/s10107-024-02130-y`.

Schrijver, Alexander (1981). “On total dual integrality”. In: *Linear Algebra and its Applications* 38, pp. 27–32. ISSN: 00243795. DOI: `10.1016/0024-3795(81)90005-7`. URL: `http://linkinghub.elsevier.com/retrieve/pii/0024379581900057`.

— (1986). *Theory of linear and integer programming.* Publication Title: Wiley-Interscience series in discrete mathematics and optimization. New York, NY, USA: John Wiley & Sons, Inc. xi, 471 p. ISBN: 0-471-90854-1.

— (2003). *Combinatorial Optimization Polyhedra and Efficiency.* Vol. 24. Issue: January 2003 ISSN: 1619-4500. Springer. 1881 pp. ISBN: ISBN 978-3-540-44389-6.

Seidman, Stephen B. and Brian L. Foster (Jan. 1, 1978). “A Graph-theoretic Generalization of the Clique Concept”. In: *The Journal of Mathematical Sociology* 6.1, pp. 139–154. ISSN: 0022-250X, 1545-5874. DOI: `10.1080/0022250X.1978.9989883`. URL: `https://www.tandfonline.com/doi/full/10.1080/0022250X.1978.9989883` (visited on 06/04/2025).

Seymour, P D (1981). “Matroids and Multicommodity Flows”. In: *European Journal of Combinatorics* 2.3, pp. 257–290. ISSN: 0195-6698. DOI: `10.1016/S0195-6698(81)80033-9`. URL: `http://www.sciencedirect.com/science/article/pii/S0195669881800339`.

Seymour, Paul D. (1979). “Sums of circuits”. In: *Graph theory and related topics* 1, pp. 341–355.

Shen, Libin and Aravind Joshi (Oct. 2005). “Incremental LTAG Parsing”. In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.* Ed. by Raymond Mooney, Chris Brew, Lee-Feng Chien, and Katrin Kirchhoff. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 811–818. URL: https://aclanthology.org/H05-1102/.

Shen, Yunzhuang, Yuan Sun, Andrew Eberhard, and Xiaodong Li (2021). "Learning Primal Heuristics for Mixed Integer Programs". In: *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533651.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada and Cambridge, MA, USA: MIT Press, pp. 3104–3112.

Tarjan, R. E. (1977). "Finding optimum branchings". In: *Networks* 7.1, pp. 25–35. DOI: 10.1002/net.3230070103.

Trukhanov, Svyatoslav, Lewis Ntaimo, and Andrew Schaefer (Oct. 2010). "Adaptive Multicut Aggregation for Two-Stage Stochastic Linear Programs with Recourse". In: *European Journal of Operational Research* 206.2, pp. 395–406. ISSN: 03772217. DOI: 10.1016/j.ejor.2010.02.025. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221710001566 (visited on 06/21/2025).

Van Belle, Rafaël, Charles Van Damme, Hendrik Tytgat, and Jochen De Weerdt (2022). "Inductive Graph Representation Learning for fraud detection". In: *Expert Systems with Applications* 193, p. 116463. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.116463. URL: https://www.sciencedirect.com/science/article/pii/S0957417421017449.

Vandenbussche, Baudouin, Stefanos Delikaraoglou, Ignacio Blanco, and Gabriela Hug (Dec. 2, 2019). *Data-Driven Adaptive Benders Decomposition for the Stochastic Unit Commitment Problem*. DOI: 10.48550/arXiv.1912.01039. arXiv: 1912.01039 [math]. URL: http://arxiv.org/abs/1912.01039 (visited on 06/21/2025). Pre-published.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018). "Graph Attention Networks". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJXMpikCZ.

Verhaeghe, Hélène, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus (Dec. 2020). "Learning Optimal Decision Trees Using Constraint Programming". In: *Constraints* 25.3–4, pp. 226–250. ISSN: 1383-7133, 1572-9354. DOI: 10.1007/s10601-020-09312-3. URL: http://link.springer.com/10.1007/s10601-020-09312-3 (visited on 06/15/2025).

Verwer, Sicco and Yingqian Zhang (2017). "Learning Decision Trees with Flexible Constraints and Objectives Using Integer Optimization". In: *Integration of AI and OR Techniques in Constraint Programming*. Ed. by Domenico Salvagnin and Michele Lombardi. Vol. 10335. Cham: Springer International Publishing, pp. 94–103. ISBN: 978-3-319-59775-1 978-3-319-59776-8. DOI: 10.1007/978-3-319-59776-8_8. URL: http://link.springer.com/10.1007/978-3-319-59776-8_8 (visited on 06/15/2025).

— (July 17, 2019). "Learning Optimal Classification Trees Using a Binary Linear Program Formulation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 1625–1632. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v33i01.33011624. URL: https://ojs.aaai.org/index.php/AAAI/article/view/3978 (visited on 03/01/2024).

Wagner, K. (1937). "Über eine Eigenschaft der ebenen Komplexe". In: *Mathematische Annalen* 114, pp. 570–590. URL: http://eudml.org/doc/159935.

Wainwright, Martin J. and Michael I. Jordan (2008). "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends® in Machine Learning* 1.1–2, pp. 1–305. ISSN: 1935-8237. DOI: 10.1561/2200000001. URL: http://dx.doi.org/10.1561/2200000001.

Wesselmann, Franz and Uh Suhl (2012). *Implementing cutting plane management and selection techniques (must read if you plan to do BnC)*. Tech. rep. University of Paderborn.

Wiseman, Sam and Yoon Kim (2019). "Amortized Bethe Free Energy Minimization for Learning Mrfs". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/dc554706afe4c72a60a25314cbaece80-Paper.pdf.

Yin, Zeyu, Jinsong Shao, Muhammad Jawad Hussain, Yajie Hao, Yu Chen, Xuefeng Zhang, and Li Wang (Dec. 2022). "DPG-LSTM: An Enhanced LSTM Framework for Sentiment Analysis in Social Media Text Based on Dependency Parsing and GCN". en. In: *Applied Sciences* 13.1, p. 354. ISSN: 2076-3417. DOI: 10.3390/app13010354. URL: https://www.mdpi.com/2076-3417/13/1/354 (visited on 03/03/2025).

Zhang, Aston, Zachary Lipton, Mu Li, and Alexander J. Smola (2024). *Dive into Deep Learning*. Cambridge New York Port Melbourne New Delhi Singapore: Cambridge University Press. 548 pp. ISBN: 978-1-009-38943-3.

Zhang, Jiayi, Chang Liu, Xijun Li, Hui-Ling Zhen, Mingxuan Yuan, Yawen Li, and Junchi Yan (2023). "A survey for solving mixed integer programming via machine learning". In: *Neurocomputing* 519, pp. 205–217.

Zheng, Hao, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li (2015). "Improving deep neural networks using softplus units". In: *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–4. DOI: 10.1109/IJCNN.2015.7280459.

Zhou, Jie, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun (2020). “Graph neural networks: A review of methods and applications”. en. In: *AI Open* 1, pp. 57–81. ISSN: 26666510. DOI: `10.1016/j.aiopen.2021.01.001`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S2666651021000012` (visited on 03/21/2025).

Zhou, Yu, Haixia Zheng, Xin Huang, Shufeng Hao, Dengao Li, and Jumin Zhao (Jan. 2022). “Graph Neural Networks: Taxonomy, Advances, and Trends”. In: *ACM Trans. Intell. Syst. Technol.* 13.1. Place: New York, NY, USA Publisher: Association for Computing Machinery. ISSN: 2157-6904. DOI: `10.1145/3495161`. URL: `https://doi.org/10.1145/3495161`.

Zhu, Haoran, Pavankumar Murali, Dzung Phan, Lam Nguyen, and Jayant Kalagnanam (2020). “A Scalable MIP-based Method for Learning Optimal Multivariate Decision Trees”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1771–1781. URL: `https://proceedings.neurips.cc/paper_files/paper/2020/file/1373b284bc381890049e92d324f56de0-Paper.pdf`.

Appendix: Full Length Papers

Discrete Optimization 17 (2015) 55–68

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Circuit and bond polytopes on series–parallel graphs[☆]

Sylvie Borne[a], Pierre Fouilhoux[b], Roland Grappe[a,*], Mathieu Lacroix[a], Pierre Pesneau[c]

[a] Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, (UMR 7030), F-93430, Villetaneuse, France

[b] Sorbonne Universités, Université Pierre et Marie Curie, Laboratoire LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France

[c] University of Bordeaux, IMB UMR 5251, INRIA Bordeaux - Sud-Ouest, 351 Cours de la libération, 33405 Talence, France

ARTICLE INFO

Article history:
Received 21 March 2014
Received in revised form 8 April 2015
Accepted 9 April 2015

Keywords:
Circuit polytope
Series–parallel graph
Extended formulation
Bond polytope

ABSTRACT

In this paper, we describe the circuit polytope on series–parallel graphs. We first show the existence of a compact extended formulation. Though not being explicit, its construction process helps us to inductively provide the description in the original space. As a consequence, using the link between bonds and circuits in planar graphs, we also describe the bond polytope on series–parallel graphs.

© 2015 Elsevier B.V. All rights reserved.

In an undirected graph, a *circuit* is a subset of edges inducing a connected subgraph in which every vertex has degree two. In the literature, a circuit is sometimes called *simple cycle*. Given a graph and costs on its edges, the *circuit problem* consists in finding a circuit of maximum cost. This problem is already NP-hard in planar graphs [1], yet some polynomial cases are known, for instance when the costs are non-positive.

Although characterizing a polytope corresponding to an NP-hard problem is unlikely, a partial description may be sufficient to develop an efficient polyhedral approach. Concerning the *circuit polytope*, which is the convex hull of the (edge-)incidence vectors of the circuits of the graph, facets have been exhibited by Bauer [2] and Coullard and Pulleyblank [3], and the cone has been characterized by Seymour [4]. Several variants of cardinality constrained versions have been studied, such as [5–8].

For a better understanding of the circuit polytope on planar graphs, a natural first step is to study it in smaller classes of graphs. For instance, in [3], the authors provide a complete description in Halin graphs.

Another interesting subclass of planar graphs are the series–parallel graphs. Due to their nice decomposition properties, many problems NP-hard in general are polynomial for these graphs, in which case it is quite

☆ Partially founded by GRO RO project.

* Corresponding author.
E-mail addresses: Sylvie.Borne@lipn.univ-paris13.fr (S. Borne), Pierre.Fouilhoux@lip6.fr (P. Fouilhoux), Roland.Grappe@lipn.univ-paris13.fr (R. Grappe), Mathieu.Lacroix@lipn.univ-paris13.fr (M. Lacroix), Pierre.Pesneau@math.u-bordeaux1.fr (P. Pesneau).

http://dx.doi.org/10.1016/j.disopt.2015.04.001
1572-5286/© 2015 Elsevier B.V. All rights reserved.

standard to (try to) characterize the corresponding polytopes. Results of this flavor were obtained for various combinatorial optimization problems, such as the stable set problem [9], graph partitioning problem [10], 2-connected and 2-edge-connected subgraph problems [11,12], k-edge-connected problems [13], Steiner-TSP problem [14].

Since a linear time combinatorial algorithm solves the circuit problem in series–parallel graphs, an obvious question arising is the description of the corresponding polytope. Surprisingly, it does not appear in the literature, and we fill in this gap with Theorem 11.

The main ingredient for the proof of our main theorem is the existence of a compact extended formulation for the circuit polytope on series–parallel graphs. An *extended formulation* of a given polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is a polyhedron $Q = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : Bx + Cy \leq d\}$ whose projection onto the x variables $proj_x(Q) = \{x \in \mathbb{R}^n :$ there exists $y \in \mathbb{R}^m$ such that $(x, y) \in Q\}$ is P. The *size* of a polyhedron is the number of inequalities needed to describe it. An extended formulation is called *compact* when its size is polynomial. We refer to [15] for further insights on this topic.

The past few years, extended formulations proved to be a powerful tool for polyhedral optimization, and thus received a growing interest in the community. Indeed, describing a polytope directly in its original space is often pretty challenging, and by looking for an extended formulation one has more tools at disposal. As an example, for most combinatorial optimization polytopes in series–parallel graphs, Martin et al. [16] proposed a general technique to derive extended formulations from dynamic programming algorithms, but the corresponding descriptions in the original space remain unknown.

Recently, it has been shown that the perfect matching polytope admits no compact extended formulation [17]. It means, even if an optimization problem is polynomial, there may not exist such a formulation. Here, though we are not able to explicitly construct a compact extended formulation for the circuit polytope on series–parallel graphs, we show that there exists one, see Section 2.1.1. The construction process of this extended formulation relies on a straightforward inductive description of the circuits of series–parallel graphs, combined with a theorem of Balas [18,19]. It allows us to prove by induction that the circuit polytope on series–parallel graphs is completely described by three families of inequalities. We provide examples where exponentially many of these inequalities define facets, see Corollary 19. Thus, the circuit polytope on series–parallel graphs is another example of polytope having exponentially many facet-defining inequalities that admits a compact extended formulation.

A graph is series–parallel if and only if, given any planar drawing of the graph, its dual is series–parallel. The dual of a circuit is a *bond*, that is a cut containing no other nonempty cut. These bonds play an important role e.g. in multiflow problems [20]. By planar duality and the description of the circuit polytope on series–parallel graphs, we get the description of the bond polytope on series–parallel graphs, see Theorem 13.

The paper is organized as follows. In Section 1, we fix graph related notation and definitions, and review some known and new auxiliary results about circuits in series–parallel graphs. Section 2 deals with the circuit polytope on series–parallel graphs. First, we get a polyhedral description of the latter for non trivial 2-connected series–parallel graphs, by providing the existence of a compact extended formulation, and then inductively projecting it. By applying standard techniques, the polyhedral description for general series–parallel graphs follows, which has exponential size in general. In Section 3, using the planar duality, we describe the bond polytope on series–parallel graphs, and then we study facet-defining inequalities, which have counterparts for the circuit polytope as well.

1. Circuits in series–parallel graphs

Throughout, $G = (V, E)$ will denote a connected undirected graph with $n = |V|$ vertices and $m = |E|$ edges. The graph *induced* by a subset W of V is the graph $G[W]$ obtained by removing the vertices of $V \setminus W$, and $\delta_G(W)$ is the set of edges having exactly one extremity in W. Given disjoint $U, W \subset V$, $\delta_G(U, W)$ is the set of edges having one extremity in each of U and W. When it is clear from the context, we will omit the subscript G. Given a set of edges $F \subseteq E$, $V(F)$ denotes the set of vertices incident to any edge of F. We denote by $A \Delta B = (A \cup B) \setminus (A \cap B)$ the symmetric difference of A and B.

A subset F of E is called a *cut* if $F = \delta_G(W)$ for some $W \subseteq V$. If $u \in W$ and $v \in V \setminus W$, the cut *separates* u and v. A cut defined by a singleton is a *star*. A *bond* is a cut containing no other nonempty cut. One can check that a nonempty cut $\delta_G(W)$ is a bond if and only if both $G[W]$ and $G[V \setminus W]$ are connected. In the literature, a bond is sometimes called a *central cut*. A *bridge* is an edge whose removal disconnects the graph, that is a bond of size one. Note that the symmetric difference of bonds is a cut.

A subset of edges is called a *cycle* if it induces a subgraph where every vertex has even degree. A connected cycle with every vertex of degree two is a *circuit*. If e is a circuit, it is called a *loop*. Let $\mathcal{C}(G)$ denote the set of circuits of G. Note that the symmetric difference of circuits is a cycle.

By definition, the emptyset is both a bond and a circuit.

When no removal of a single vertex disconnects a graph, the latter is said *2-connected*. Loops and bridges are called *trivial 2-connected* graphs. The *non trivial 2-connected components* of a graph are the maximal 2-connected subgraphs of the graph, *i.e.*, the components obtained after removing the loops and bridges.

A graph is *series–parallel* if all its non trivial 2-connected components can be built, starting from the circuit of length two C_2, by repeatedly applying the following operations: add a parallel edge to an existing edge; or subdivide an existing edge, that is replace the edge by a path of length two. This construction gives an inductive description of the circuits of such graphs.

Observation 1. *Let $G = (V, E)$ be a non trivial 2-connected series–parallel graph.*

(i) *If G is obtained from a graph H by subdividing an edge $e \in E(H)$ into e, f, then the circuits of G are obtained from those of H as follows:*
  - *C, for $C \in \mathcal{C}(H)$ not containing e,*
  - *$C \cup f$, for $C \in \mathcal{C}(H)$ containing e.*

(ii) *If G is obtained from a graph H by adding a parallel edge f to an edge $e \in E(H)$, then the circuits of G are obtained from those of H as follows:*
  - *C, for $C \in \mathcal{C}(H)$ not containing e,*
  - *C and $C \setminus e \cup f$, for $C \in \mathcal{C}(H)$ containing e,*
  - *$\{e, f\}$.*

A well-known characterization of cuts is that they are the sets of edges intersecting every circuit an even number of times. In series–parallel graphs, we have the following property [20].

Observation 2 ([20]). *In a series–parallel graph, a bond and a circuit intersect in zero or two edges.*

If the graph is also 2-connected, then this property becomes a characterization of circuits, see below. Note that the following does not hold if the series–parallel graph is not 2-connected.

Lemma 3. *In a non trivial 2-connected series–parallel graph, a set of edges is a circuit if and only if it intersects every bond in zero or two edges.*

Proof. We prove the non trivial direction. By contradiction, let G be a minimal counter-example, and let F be a set of edges intersecting every bond in zero or two edges that is not a circuit. First, suppose that G is build from H by adding a parallel edge f to an edge $e \in E(H)$. Necessarily, we have $f \in F$ as otherwise H would be a smaller counter-example. Similarly, $e \in F$. Suppose there exists $g \in F \setminus \{e, f\}$. Since G is planar and 2-connected, so is its dual. Any pair of edges in a 2-connected graph being contained in a circuit, the planar duality between circuits and bonds implies that there exists a bond B of G containing both g and e. Hence, B also contains f, which provides the contradiction $|F \cap B| \geq 3$. Now, assume that G is build from H by subdividing $e \in E(H)$ into $\{f, g\}$. Since G is 2-connected, $\{f, g\}$ is a bond, hence F contains either both f and g or none of them. In both cases, H is clearly a smaller counter-example, a contradiction. □

For an ordering $v_1, \dots, v_n$ of V such that $\delta(\{v_1, \dots, v_i\})$ is a bond for all $i = 1, \dots, n-1$, the partition $\mathcal{S} = \{S_1, \dots, S_{n-1}\}$ of E defined by $S_\ell = \delta(v_\ell, \{v_{\ell+1}, \dots, v_n\})$, for $\ell = 1, \dots, n-1$, is a *star decomposition*. We will denote the initial star $\delta(v_1, \{v_2, \dots, v_n\})$ by $I_{\mathcal{S}}$. Equivalently, a star decomposition is obtained by partitioning the edgeset by iteratively removing stars of the graph such that, at each step, the vertex to be removed is adjacent to some removed vertex, and the set of remaining vertices induces a connected graph. $I_{\mathcal{S}}$ is the unique element of the star decomposition which is a star of the original graph.

Using induction and the construction of non trivial 2-connected series–parallel graphs, one can see that in these graphs any vertex is the initial vertex of some star decomposition. In particular, such decompositions exist.

Lemma 4. *Given a star decomposition $\mathcal{S}$ of a series–parallel graph G, the following holds:*

(a) *a circuit intersects each member of $\mathcal{S}$ at most twice,*
(b) *a circuit does not intersect two members of $\mathcal{S}$ twice.*

Proof. Let C be a circuit of G and $v_1, \dots, v_n$ an ordering of V such that $\mathcal{S} = \{S_1, \dots, S_{n-1}\}$ with $S_\ell = \delta(v_\ell, \{v_{\ell+1}, \dots, v_n\})$, for $\ell = 1, \dots, n-1$.

Since every member of $\mathcal{S}$ is contained in a star of G and a circuit goes through each vertex at most once, Lemma 4(a) holds. Let us show Lemma 4(b) by contradiction, and let $i < j$ be such that $|S_i \cap C| = |S_j \cap C| = 2$ and $|S_k \cap C| \leq 1$ for all $k < j$, $k \neq i$. By construction of star decompositions, we have $C \cap S_\ell = \emptyset$, for all $\ell < i$, and $C \setminus (\bigcup_{\ell=1}^{j-1} S_\ell)$ is a path of which S_j contains two edges, hence $|\delta(\{v_1, \dots, v_j\}) \cap C| = 4$. Since $\delta(\{v_1, \dots, v_j\})$ is a bond, this contradicts Observation 2. □

Two sequences of edge subsets $\mathcal{M} = (M_0, \dots, M_k)$ and $\mathcal{N} = (N_1, \dots, N_k)$ form a *star-cut collection* if $\{M_0, \dots, M_k\} \subseteq \mathcal{S}$ and $M_0 = I_{\mathcal{S}}$, for some star decomposition $\mathcal{S}$ of G, and $M_i \Delta N_i$ is a cut of G, for $i = 1, \dots, k$. Note that the elements of $\mathcal{N}$ are not required to be disjoint.

2. Circuit polytope on series–parallel graphs

Given a graph $G = (V, E)$ and $F \subseteq E$, $\chi^F \in \mathbb{R}^E$ denotes the incidence vector of F, that is χ^F_e equals 1 if $e \in F$ and 0 otherwise. Since there is a bijection between edge sets and their incidence vectors, we will often use the same terminology for both. Let $C(G)$ be the convex hull of the incidence vectors of the circuits of G, that is $C(G) = \mathrm{conv}\{\chi^C : C \in \mathcal{C}(G)\}$. In this section, we give an external description of the circuit polytope on series–parallel graphs.

Note that the circuit polytope of the graph is the union of the circuit polytopes of its loops, bridges, and non trivial 2-connected components. Therefore, we start by studying the circuit polytope for this latter case, and then derive the description for general series–parallel graphs.

Throughout, we will use the following theorem of Balas [18,19]. His result holds for any finite union of polyhedra, yet we only state what we need in this paper, the union of two polytopes.

Theorem 5 (*Balas [18,19]*). *Given two polytopes $P_1 = \{x \in \mathbb{R}^n : A^1x \leq b^1\}$ and $P_2 = \{x \in \mathbb{R}^n : A^2x \leq b^2\}$, we have $conv\{P_1 \cup P_2\} = proj_x(Q)$, where $Q = \{x = x^1 + x^2, A^1x^1 \leq (1-\lambda)b^1, A^2x^2 \leq \lambda b^2, 0 \leq \lambda \leq 1\}$.*

Note that Theorem 5 applied to integral polytopes yields an extended formulation which is also integral. Furthermore, it also implies the following.

Corollary 6. *Given two polytopes P_1 and P_2, there exists an extended formulation of conv$\{P_1 \cup P_2\}$ whose size is two plus the sizes of P_1 and P_2.*

Later on, we shall use this corollary when P_2 is a vertex, in which case we get an extended formulation of conv$\{P_1 \cup P_2\}$ with two more inequalities than the one of P_1.

2.1. 2-connected series–parallel graphs

In this section, we describe the circuit polytope for non trivial 2-connected series–parallel graphs. The main ingredient of our proof is the existence of a compact extended formulation for this polytope, based on Observation 1. Though this extended formulation is not explicit, we use its construction process to prove inductively that the circuit polytope is described by the inequalities given in Theorem 10. Let us mention that there are examples where exponentially many of these inequalities are facet-defining, see Corollary 19.

In this section, $G = (V, E)$ is a non trivial 2-connected series–parallel graph.

2.1.1. Existence of a compact extended formulation

We show the existence of a compact extended formulation by induction on the construction of G. First, note that $C(C_2) = \text{conv}\{(0,0),(1,1)\} = \{x \in \mathbb{R}^2_+ : x_e = x_f, x_e + x_f \leq 2\}$, where e and f denote the edges of C_2. Next, let us describe how to get an extended formulation for $C(G)$ when G is obtained from a 2-connected series–parallel graph H by either subdividing an edge or adding a parallel edge.

When G is obtained from H by subdividing an edge $e \in E(H)$ into e, f, the following immediately derives from Observation 1(i).

Observation 7. *Suppose G is obtained from H by subdividing an edge $e \in E(H)$ into e, f. Then, adding a variable x_f to any extended formulation of $C(H)$ and imposing $x_e = x_f$ provides an extended formulation for $C(G)$.*

When G is obtained from H by adding a parallel edge f to $e \in E(H)$, an extended formulation for $C(G)$ can be obtained as follows.

Lemma 8. *Suppose G is obtained from H by adding a parallel edge f to an edge $e \in E(H)$ and let $Q(H)$ be an integral polyhedron which is an extended formulation of $C(H)$. Then,*

(a) *The polytope $S(G)$ obtained by replacing x_e by $x_e + x_f$ in $Q(H)$ and setting $0 \leq x_e$ and $0 \leq x_f$ is an extended formulation of the convex hull of the incidence vectors of all the circuits of G different from $\chi^{e,f}$.*
(b) *The convex hull of $S(G)$ union $\chi^{e,f}$ is an extended formulation of $C(G)$.*

Proof. (a) Let $R(G)$ denote the convex hull of incidence vectors of all the circuits of G except $\{e, f\}$. By Observation 1(ii), since $proj_x Q(H) = C(H)$, we have $proj_x S(G) \cap \mathbb{Z}^m = R(G) \cap \mathbb{Z}^m$. Since $Q(H)$ is integral, so is $S(G)$, which implies the integrality of $proj_x S(G)$.

(b) By (a), $proj_x S(G)$ is integral, hence so is $\text{conv}\{proj_x S(G) \cup \chi^{e,f}\}$. Since the projection of the convex hull of a set of points is the convex hull of its projected points, $proj_x\big(\text{conv}\{S(G) \cup \chi^{e,f}\}\big)$ is integral, and we are done. □

Note that the operations involved in Observation 7 and Lemma 8 preserve integrality. By construction of non trivial 2-connected series–parallel graphs, and since $C(C_2)$ is integral, we get an extended formulation for $C(G)$ by repeatedly applying Observation 7 and Lemma 8. Moreover, the extended formulation given by Lemma 8(a) yields two new inequalities, and that applying Corollary 6 in Lemma 8(b) provides an extended formulation with two more inequalities. Thus, if G is obtained from H by adding a parallel edge, then an extended formulation for $C(G)$ has 4 more inequalities than an extended formulation for $C(H)$. Furthermore, if G is obtained from H by subdividing an edge, then an extended formulation for $C(G)$ has the size of an extended formulation for $C(H)$. The following corollary stems from these observations and the fact that $C(C_2)$ is described by 3 inequalities.

Corollary 9. *There exists an extended formulation for $C(G)$ of size $O(|E(G)|)$.*

We mention here that a polytope closely related to the circuit polytope is, given a vertex r, the *r-circuit polytope*, that is the convex hull of the circuits containing r. Indeed, the circuit polytope of a graph can be seen as the union of all its r-circuit polytopes. In series–parallel graphs, the latter have been thoroughly studied by Baïou and Mahjoub in [14] who provide, in particular, their description into the original space. Therefore, an explicit extended formulation for the circuit polytope on series–parallel graphs can be obtained by applying Balas' Theorem [18,19] for the union of polyhedra together with their description. However, since the description of the r-circuit polytope has exponentially many inequalities, this approach yields an exponential-size extended formulation. Moreover, projecting such a formulation to get a description into the original space usually requires tremendous efforts. In contrast, our approach allows to project step by step, which is done in the next section.

2.1.2. Description in the original space

In this section, we show that the inequalities (1)–(3) given below describe the circuit polytope on non trivial 2-connected series–parallel graphs, see Theorem 10. Throughout, for a sequence $\mathcal{M} = (M_0, \ldots, M_k)$ of edge sets, $x(\mathcal{M})$ will stand for $\sum_{i=1}^{k} x(M_i)$.

$$x_e \geq 0 \qquad \text{for all } e \in E. \tag{1}$$

$$x_e \leq x(B \setminus e) \qquad \text{for all bonds } B \text{ of } G, \text{ for all } e \in B, \tag{2}$$

$$x(\mathcal{M}) - x(\mathcal{N}) \leq 2 \qquad \text{for all } \mathcal{M}, \mathcal{N} \text{ star-cut collections of } G, \tag{3}$$

Inequalities (1) are called *non-negativity constraints*, (2) are *bond constraints*, and (3) are *star-cut constraints*.

Theorem 10. $C(G) = \{x \in \mathbb{R}^m_+$ satisfying (2) and (3)$\}$.

Proof. Let us first show that (1)–(3) are valid for $C(G)$. Clearly, every incidence vector of a circuit satisfies the non-negativity constraints (1). The validity of bond constraints (2) comes from Observation 2. To show the validity of star-cut constraints (3), let $\mathcal{M}, \mathcal{N}$ be a star-cut collection and C a circuit of G. Since M_0 and $M_i \Delta N_i$ are cuts for $i \in \{1, \ldots, k\}$, each of them intersects C an even number of times. Therefore, if C intersects $M_i \in \mathcal{M}$ at most once, then $\chi^C(M_i) - \chi^C(N_i) \leq 0$ if $i \geq 1$ and $\chi^C(M_i) = 0$ if $i = 0$. The validity of $x(\mathcal{M}) - x(\mathcal{N}) \leq 2$ follows since, by Lemma 4, at most one member of $\mathcal{M}$ intersects C twice, the other ones intersecting C at most once.

Let us prove the theorem by induction. The first step of the induction comes from $C(C_2) = \{x \in \mathbb{R}^2_+ :$ satisfying (2) and $x_e + x_f \leq 2\}$ and the fact that $\{\{e, f\}\}, \emptyset$ forms a star-cut collection, where $C_2 = \{e, f\}$. Suppose now that $C(H)$ is given by inequalities (1)–(3) for a non trivial 2-connected series–parallel graph H, and let us show that $C(G)$ is also described by (1)–(3) when G is obtained from H by subdividing an edge or by adding a parallel edge in H.

First, remark that if G is obtained from H by subdividing e into e, f, then $C(G)$ is given by the inequalities of $C(H)$ and $x_e = x_f$. The inequalities of $C(H)$ of type (1)–(3) remain of the same type in G, and $x_e = x_f$ is implied by the two inequalities of type (2) associated with the bond $\{e, f\}$.

Now, let G be obtained from H by adding a parallel edge f to $e \in E(H)$. By the induction hypothesis, we have $C(H) = \{x^H \in \mathbb{R}^{m-1} : A^H x^H \leq b^H\}$ where A^H is given by the non-negativity (1), bond (2), and star-cut (3) constraints for H. Denote by $\bar{A}^H$ and $\bar{x}^H$ the matrix and vector obtained from A^H and x^H by, respectively, removing the column A^H_e corresponding to e and the component x^H_e. The application of Lemma 8(a) introduces a new variable y and provides the following description of $S(G)$:

$$\{(x^H, y) \in \mathbb{R}^{m-1} \times \mathbb{R} : \bar{A}^H \bar{x}^H + A^H_e(x^H_e + y) \leq b^H, 0 \leq x^H_e, 0 \leq y\}.$$

Lemma 8(b) implies that $C(G)$ is the convex hull of the union of $S(G)$ and $\chi^{\{e,f\}}$. Let us apply Theorem 5 to $P_1 = S(G)$ and $P_2 = \{\chi^{\{e,f\}}\}$. The latter being a vertex, we can get rid of x^1 and x^2 to get the following extended formulation of $C(G)$, where $\bar{x}$ denotes the vector x after the removal of x_e and x_f.

$$\{(\bar{x}, x_e, x_f, \lambda) \in \mathbb{R}^{m-2} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} : \bar{A}^H \bar{x} + A^H_e(x_e + x_f - 2\lambda) \leq (1-\lambda)b^H, \lambda \leq x_e, \lambda \leq x_f, 0 \leq \lambda \leq 1\}.$$

To project it by Fourier–Motzkin's method [21], we only need to consider the inequalities where λ appears, and since A^H is given by (1)–(3) for H, we may write them down explicitly, implicitly using the fact that if e belongs to a cut of G, then so does f, and conversely:

$$0 \leq \lambda \tag{4}$$

$$-1 \leq -\lambda \tag{5}$$

$$-x_h \leq -\lambda \quad \text{for } h = e, f \tag{6}$$

$$x_\ell - x(B \setminus \ell) \leq -2\lambda \quad \text{for all bonds } B \text{ of } G \text{ containing } e, f \text{ and } \ell \in B \setminus \{e, f\} \tag{7}$$

$$x_e + x_f - x(D \setminus \{e, f\}) \leq 2\lambda \quad \text{for all bonds } D \text{ of } G \text{ containing } e, f \tag{8}$$

$$x(\mathcal{M}) - x(\mathcal{N}) \leq 2 - 2(\alpha_e(\mathcal{M}, \mathcal{N}) + 1)\lambda \quad \text{for all star-cut collections } \mathcal{M}, \mathcal{N} \text{ of } G, \text{ with } \alpha_e(\mathcal{M}, \mathcal{N}) \geq 0, \tag{9}$$

where $\alpha_e(\mathcal{M}, \mathcal{N}) = |\{N \in \mathcal{N} : e \in N\}| - |\{M \in \mathcal{M} : e \in M\}|$.

We now prove that the inequalities obtained by projecting out λ are either contained or implied by the non-negativity constraints (1) and bond constraints (2) and star-cut constraints (3) for G, which implies our theorem. Recall that, to get rid of λ, one has to combine every inequality where λ's coefficient is negative with every inequality where it is positive [21]. Combinations with $0 \leq \lambda$ immediately give rise to inequalities of type (1), (2) or (3) for G. Thus, it remains to combine (8) with every other inequality.

First, remark that adding twice inequality (5) to any inequality (8) leads to an inequality obtained by adding non-negativity constraints and the star-cut constraint $x(M_0) \leq 2$ where M_0 is a star of G containing e, f. Moreover, adding (8) to twice (6) gives $x_h - x(D \setminus h) \leq 0$ for all bonds D containing e, f, and $h \in \{e, f\}$, which are inequalities of type (2).

Adding (8) to (7) gives $x_\ell \leq x(B \setminus \{e, f, \ell\}) + x(D \setminus \{e, f\})$. If D contains ℓ, the latter is a sum of non-negativity constraints (1). Otherwise, $B \Delta D$ is a cut contained in $B \cup D \setminus \{e, f\}$ and thus contains a bond J containing ℓ but not e, f, since a cut is a disjoint union of bonds. Hence, the inequality is the sum of $x_\ell \leq x(J \setminus \ell)$ and non-negativity constraints (1).

For a bond D containing e, f and a star-cut collection $\mathcal{M} = (M_0, \ldots, M_k)$, $\mathcal{N} = (N_1, \ldots, N_k)$ with $\alpha_e(\mathcal{M}, \mathcal{N}) \geq 0$, combining (8) and (9) gives $x(\mathcal{M}) - x(\mathcal{N}) + (\alpha_e(\mathcal{M}, \mathcal{N}) + 1)(x_e + x_f - x(D \setminus \{e, f\})) \leq 2$. If e and f belong to a member of $\mathcal{M}$, then $\alpha_e(\mathcal{M}, \mathcal{N}) + 1 = |\{N \in \mathcal{N} : e \in N\}|$. Moreover, considering separately the elements of $\mathcal{N}$ containing e and f from the other ones, the inequality can be rewritten as:

$$x(\mathcal{M}) - \sum_{N \in \mathcal{N} : e, f \in N} \Big(x(N \setminus \{e, f\}) + x(D \setminus \{e, f\}) \Big) - \sum_{N \in \mathcal{N} : e, f \notin N} x(N) \leq 2.$$

Since $x(N\Delta D) \leq x(N \setminus \{e,f\}) + x(D \setminus \{e,f\})$ for all $N \in \mathcal{N}$ containing e and f, the above inequality is implied by $x(\mathcal{M}) - x(\mathcal{N}') \leq 2$, where $\mathcal{N}' = (N'_1, \dots, N'_k)$ with N'_i equals $N_i \Delta D$ if $e \in N_i$ and N_i otherwise, for $i = 1, \dots, k$. Moreover, since D and $M_i \Delta N_i$ are cuts, so is $M_i \Delta N'_i$, for $i = 1, \dots, k$, as the symmetric difference of two cuts is a cut. Therefore, $\mathcal{M}, \mathcal{N}'$ is a star-cut collection.

Suppose now that no member of $\mathcal{M}$ contains e and f. Applying the previous argument leads to the inequality $x(\mathcal{M}) - x(\mathcal{N}') + x_e + x_f - x(D \setminus \{e,f\}) \leq 2$, where $\mathcal{M}, \mathcal{N}'$ is the star-cut collection defined above. Moreover, there exists M_{k+1} containing e, f such that $\{M_0, \dots, M_k, M_{k+1}\}$ is contained in a star decomposition. Let $\tilde{\mathcal{M}} = (M_0, \dots, M_{k+1})$ and $\tilde{\mathcal{N}} = (N'_1, \dots, N'_{k+1})$, where $N'_{k+1} = D \Delta M_{k+1}$. The symmetric difference being associative, $M_{k+1} \Delta N_{k+1}$ equals D, and hence $\tilde{\mathcal{M}}, \tilde{\mathcal{N}}$ is a star-cut collection of G. Moreover, the associated star-cut constraint (3) implies the inequality obtained by combination of (8) and (9). □

Let us mention a few simple constraints implied by the ones of Theorem 10. First, whenever $\{k, \ell\}$ is a bond, we have $x_k = x_\ell$, which is implied by the inequalities (2) for $\{k, \ell\}$. These will turn out to be the only hyperplanes containing $C(G)$. We postpone the proof of this fact to Section 3.3, see Corollary 18 . For every edge $uv \in E$, $\delta_G(u)$ is a bond since G is 2-connected. Then, we obtain the inequality $x_{uv} \leq 1$ by adding x_{uv} to each side of $x_{uv} \leq x(\delta_G(u) \setminus uv)$ and by applying $x(\delta_G(u)) \leq 2$, which is a special case of (3). We also mention that, given a bond B, the inequality $x(B) \leq 2$ is implied by a suitable star-cut constraint.

We will see at the end of Section 3 a family of examples where exponentially many of the inequalities of Theorem 10 define facets.

2.2. General series–parallel graphs

In this section, we provide a polyhedral description of the circuit polytope on general series–parallel graphs, see Theorem 11. The result is obtained by applying a standard union technique and the fact that the circuit polytope of a graph is the convex hull of the union of the circuit polytopes of its 2-connected components.

Theorem 11. *Let G be a series–parallel graph, $G_1, \dots, G_k$ its non trivial 2-connected components, $\mathfrak{L}$ its set of loops, and $\mathfrak{B}$ its set of bridges. Then*

$$C(G) = \left\{ \begin{array}{ll} x \in \mathbb{R}^m_+ \text{ satisfying (2)}, \ x(\mathfrak{B}) = 0 \text{ and} & \\ \displaystyle\sum_{i=1}^{k} (x(\mathcal{M}_i) - x(\mathcal{N}_i)) + 2x(\mathfrak{L}) \leq 2, & \begin{array}{l}\text{for all } i = 1, \dots, k, \\ \text{for all star-cut collections } \mathcal{M}_i, \mathcal{N}_i \text{ of } G_i\end{array} \end{array} \right\}.$$

Proof. We prove the result by induction on the number of 2-connected components.

Let us see the first step. Since no bridge b belongs to a circuit, its circuit polytope is described by $\{x_b = 0\}$. Moreover, the circuit polytope of a loop ℓ is described by $\{0 \leq 2x_\ell \leq 2\}$. Finally, the circuit polytope of a non trivial 2-connected series–parallel component is given by Theorem 10.

Suppose that the result holds for two series–parallel graphs I and $H = \bigcup_{i=1}^{k-1} H_i$, where I is 2-connected and $H_i, i = 1, \dots, k-1$ are the 2-connected components of H, and let $G = I \bigcup (\cup_{i=1}^{k-1} H_i)$ be the graph obtained by identifying a vertex of I and a vertex of H. Then, $C(G) = \text{conv}\{C(H) \cup C(I)\}$. Remark that $C(H) = \{x \in \mathbb{R}^{E(H)}_+ : A_H x_H \leq b_H\}$ and $C(I) = \{x \in \mathbb{R}^{E(I)}_+ : A_I x_I \leq b_I\}$ live in different spaces. Extend them to polytopes of $\mathbb{R}^{E(H)} \times \mathbb{R}^{E(I)}$ by setting the new coordinates to zero, and apply Theorem 5 to get $C(G) = proj_x\{x = (x_H, x_I), A_H x_H \leq \lambda b_H, A_I x_I \leq (1-\lambda) b_I, 0 \leq \lambda \leq 1\}$.

Let us get rid of λ in the above extended formulation. Combinations with $0 \leq \lambda$ or $\lambda \leq 1$ immediately give desired inequalities. It remains to combine $a_H x_H \leq \lambda b_H$ and $a_I x_I \leq (1-\lambda) b_I$ when $b_H \neq 0$ and $b_I \neq 0$.

Since in this case the induction hypothesis says that both inequality are of the new type, we get $b_H = b_I = 2$, thus the resulting inequality is $a_H x_H + a_I x_I \leq b_H$, and the theorem follows. $\square$

Since every 2-connected component of a series–parallel graph has a compact extended formulation, using the Theorem of Balas [18,19] for the union of several polytopes, one can extend Corollary 9 as follows.

Corollary 12. *If G is series–parallel, there exists a compact extended formulation of $C(G)$ in size $O(|E|)$.*

3. The bond polytope on series–parallel graphs

In this section, as a consequence of Theorem 11 and the planar duality, we describe the bond polytope on series–parallel graphs. We also provide examples where the latter contains exponentially many facets. Before stating these results, we introduce a few definitions.

3.1. Definitions

Given a series–parallel graph G, we denote its set of bonds by $\mathcal{B}(G)$, and the convex hull of their incidence vectors by $B(G)$.

If G is a non trivial 2-connected series–parallel graph, an *open nested ear decomposition* [22] $\mathcal{E}$ of G is a partition of $E(G)$ into a sequence $E_0, \ldots, E_k$ such that E_0 is a circuit of G and the *ears* E_i, $i \in \{1, \ldots, k\}$, are paths with the following properties:

- the two endpoints of each ear are distinct and appear in an E_j with $j < i$,
- no interior point of an ear E_i belongs to E_j for all $j < i$,
- if two ears E_i and $E_{i'}$ have both their endpoints in the same E_j, then any two paths contained in E_j, one between the endpoints of E_i and the other between the endpoints of $E_{i'}$, are either disjoint or contained one in another.

We will denote by $C_{\mathcal{E}}$ the unique circuit of an open nested ear decomposition $\mathcal{E}$. Two sequences of edge subsets $\mathcal{F} = (F_0, F_1, \ldots, F_k)$ and $\mathcal{P} = (P_1, \ldots, P_k)$ form an *ear-cycle collection* if $\{F_0, F_1, \ldots, F_k\}$ is contained in an open nested ear decomposition $\mathcal{E}$ of G, $F_0 = C_{\mathcal{E}}$, and $F_i \Delta P_i$ is a cycle for $i = 1, \ldots, k$. Note that the elements of $\mathcal{P}$ are not required to be disjoint.

A graph H is a *minor* of G if H arises from G by contractions and deletions of edges and deletions of vertices, where *contracting* an edge uv of E corresponds to deleting e and identifying u and v. A graph is series–parallel if and only if it does not contain a K_4-minor [23], where K_4 denotes the complete graph on four vertices.

3.2. The bond polytope on series–parallel graphs

K_4 being its own dual, a graph is series–parallel if and only if, given any planar drawing of the graph, its dual is series–parallel. It is immediate that the circuits of such a graph are precisely the bonds of its dual, thus the bond polytope of the graph is the circuit polytope of its dual. Then, applying Theorem 11 provides a description of the bond polytope on series–parallel graphs.

Given a circuit C and $e \in C$, $x_e \leq x(C \setminus e)$ is a *circuit constraint*, and given an ear-cycle collection, $x(\mathcal{F}) - x(\mathcal{P}) \leq 2$ is an *ear-cycle constraint.*

Theorem 13. *Let G be a series–parallel graph, $G_1, \dots, G_k$ its non trivial 2-connected components, $\mathfrak{L}$ its set of loops and $\mathfrak{B}$ its set of bridges.*

$$B(G) = \left\{ \begin{array}{ll} x \in \mathbb{R}^m_+ \text{ satisfying } x_e \le x(C \setminus e) \text{ for all circuits } C \text{ and } e \in C,\ x(\mathfrak{L}) = 0 \text{ and} & \\ \displaystyle\sum_{i=1}^{k} (x(\mathcal{F}_i) - x(\mathcal{P}_i)) + 2x(\mathfrak{B}) \le 2, & \begin{array}{l} \text{for all } i = 1, \dots, k, \\ \text{for all ear-cycle collections } \mathcal{F}_i, \mathcal{P}_i \text{ of } G_i \end{array} \end{array} \right\}.$$

Proof. Fix a planar drawing of G, and let $\tilde{G}$ be the dual graph of G. The edgesets of G and $\tilde{G}$ are in bijection, and $\tilde{e}$ will denote the edge of $E(\tilde{G})$ corresponding to $e \in E(G)$. As noted above, the bond polytope of G is precisely the circuit polytope of $\tilde{G}$. First, recall that bridges of G are in bijection with loops of $\tilde{G}$, and conversely. Then, by Theorem 11, to get the desired result, we just need to show that the bond polytope on non trivial 2-connected series–parallel graphs is given by non-negativity, circuit and ear-cycle constraints.

Let G be a non trivial 2-connected series–parallel graph. Then, $\tilde{G}$ is also a non trivial 2-connected series–parallel graph. Since, by Theorem 10, $C(\tilde{G})$ is described by (1)–(3), and by the bijection between circuits in G and bonds in $\tilde{G}$, we only have to show that the ear-cycle constraints are valid for $B(G)$ and that a star-cut collection of $\tilde{G}$ is an ear-cycle collection of G.

To see the validity of the constraints, let us show that, given an open nested ear decomposition $\mathcal{E} = \{E_0, \dots, E_k\}$ and a bond B of G,

$$(*) \text{ if } |B \cap E| = 2 \text{ for some } E \in \mathcal{E}, \text{ then } |B \cap F| \le 1 \text{ for all } F \in \mathcal{E} \setminus E.$$

First, note that, by Observation 2 and the fact that an ear is always contained in a circuit, we have $|B \cap E| \le 2$, for all $E \in \mathcal{E}$. Now, suppose that $E_i, E_j \in \mathcal{E}$ both intersect B twice, with $i < j$. Denote by u and v the extremities of E_j and let e be an edge of $E_i \cap B$. The graph induced by the edges of $E_0 \cup \ldots \cup E_i \cup \{uv\}$ is 2-connected so it contains a circuit containing e and uv. Replacing uv by the ear E_j, we get that G contains a circuit C containing e and E_j. Therefore, $|C \cap B| \ge 3$, yet B is a bond, a contradiction to Observation 2. Therefore $(*)$ holds.

Then, with arguments similar to those proving the validity of star-cut constraints for the circuit polytope (see Theorem 10), we get the validity of the ear-cycle constraints by $(*)$ and the fact that a circuit and a cut intersect each other an even number of times.

We now prove by induction on the number of edges of G that a star decomposition of $\tilde{G}$ corresponds to an open nested ear decomposition of G. We will use edge subdivision and parallel addition operations, thus note that these two operations are dual one of each other. As the dual of C_2 is C_2, one can easily check that a star decomposition of C_2 corresponds to an open nested ear decomposition in its dual.

If G is obtained from H by subdividing an edge $e \in E(H)$ into e, f, then $\tilde{G}$ is obtained from $\tilde{H}$ by adding a parallel edge $\tilde{f}$ to $\tilde{e}$. By induction, any star decomposition $\mathcal{S}$ of $\tilde{H}$ corresponds to an ear decomposition $\mathcal{E}_{\mathcal{S}}$ of H. Adding e to the suitable set of $\mathcal{S}$ (which is the first extremity of e appearing in the star decomposition) gives a star decomposition of $\tilde{G}$, which straightforwardly corresponds to the ear decomposition of G obtained from $\mathcal{E}_{\mathcal{S}}$ by replacing e by $\{e, f\}$ in the member of $\mathcal{E}_{\mathcal{S}}$ containing e.

If G is obtained from H by adding a parallel edge f to $e \in E(H)$, then $\tilde{G}$ is obtained from $\tilde{H}$ by subdividing $\tilde{e}$ into $\tilde{e}, \tilde{f}$. Let u be the vertex that is common to $\tilde{e}$ and $\tilde{f}$, and v, w the other ends of $\tilde{e}$ and $\tilde{f}$. Let $\mathcal{S} = \{\delta_{\tilde{G}_1^+}(v_1), \dots, \delta_{\tilde{G}_{n-1}^+}(v_{n-1})\}$ be a star decomposition of $\tilde{G}$. We may suppose, without loss of generality, that u and v or u and w are consecutive in the star decomposition. Indeed, otherwise, $u = v_i$ for some $i \in \{2, \dots, n-1\}$, and since G_i^+ and G_i^- are connected, exactly one of v, w is in G_i^- that is, equals some v_j for $j < i$. In this case, the star decomposition $\mathcal{S}'$ obtained from $\mathcal{S}$ by removing u, and then inserting

u right after v_j, without changing the rest, gives the same partition of E as $\mathcal{S}$. Thus $\mathcal{S}$ and $\mathcal{S}'$ are in bijection with the same partition of the edgesets of G.

Since u and one of v, w appear consecutively in the star decomposition, contracting them gives a star decomposition of $\tilde{H}$. By induction, the latter corresponds to an ear decomposition $\mathcal{E}$ of H. Now, possibly having exchanged the role of e and f because of the contraction, $\mathcal{E} \cup \{f\}$ is an ear decomposition of G, and we are done.

To finish the proof, if suffices to apply Theorem 11 and the fact that the dual of a cut is a cycle. $\square$

It turns out that there are more ear-cycle collections of G than star-cut collections of $\tilde{G}$, and it is unclear which restrictions are to be made in order to get a bijection. As a consequence, if $B(G)$ can be deduced from $C(\tilde{G})$ in a rather simple manner, the converse seems more challenging.

3.3. Facet-defining inequalities

Determining directly which inequalities are facet-defining for the circuit polytope is not that easy. Surprisingly, the bond polytope is much simpler to study polyhedrally. The main reason is that we can safely remove parallel edges, see Observation 14. Thus Theorem 13 is not only a standard planar duality result, but also a tool to prove polyhedral results for the original polytope.

First, we provide the dimension of the bond polytope, see Lemma 15. Then, we characterize which of the non-negativity and circuit constraints are facet-defining, see Lemma 16. Unfortunately, it seems challenging to exhibit the structures for which ear-cycle inequalities define facets. We provide an example where exponentially many of them are facet-defining, see Claim 17.

Seen the structure of the inequalities given by Theorem 13, it is enough to study the facet-defining ones for non trivial 2-connected series–parallel graphs. In this section, let $G = (V, E)$ be such graph.

Observation 14. *The set of bonds of G is unchanged if we remove parallel edges.*

Proof. Whenever two edges of the graph are parallel, every bond contains either both or none. $\square$

By the above observation and the construction of non trivial 2-connected series–parallel graphs, we may assume there are no parallel edges. This is emphasized by the following lemma.

Lemma 15. *The dimension of $B(G)$ is the number of edges of the graph obtained from G by removing every parallel edge.*

Proof. By Observation 14, we may assume that G has no parallel edges. Then, since the emptyset is a bond, that is $0 \in B(G)$, the result is equivalent to the existence of $|E(G)|$ linearly independent bonds of G. Clearly, $\dim B(G) \leq |E(G)|$. We prove the result by induction on $|E(G)|$, noting that $\dim B(C_3) = 3$.

Since G has no parallel edges, G is obtained from a non trivial 2-connected series–parallel graph H by subdividing an edge e into f, g.

If H contains no parallel edges, then the induction hypothesis gives a family of $\dim B(H) = |E(H)| = |E(G)| - 1$ linearly independent bonds of H. Replacing e by f, for each member of $\mathcal{F}$ containing e, and then adding $\{f, g\}$, gives a linearly independent family of $|E(G)|$ bonds of G, and we are done.

Since G did not contain parallel edges, if H does, then these parallel edges are $\{e, h\}$ for some $h \in E(G) \setminus \{f, g\}$. In this case, we have $B(H) \subseteq \{x_e = x_h\}$. By the induction hypothesis, there exists a family $\mathcal{L}$ of $\dim B(H) = |E(H)| - 1 = |E(G)| - 2$ linearly independent bonds of H. We may assume that $e \in B$ for some $B \in \mathcal{L}$. Define $D = B \setminus e \cup f$, we get the family $\mathcal{L} \cup D \cup \{f, g\}$ of bonds of G. Let us prove that they are linearly independent, by showing that the corresponding matrix A has full column rank. Since $D = B \setminus e \cup f$, by basic column operations we get that A has the same rank as the matrix whose columns are composed of χ^e, χ^f and the elements of $\mathcal{L}$. Thus A has full column rank if and only if the matrix obtained from $\mathcal{L}$ by deleting the coordinate corresponding to e has. It is indeed the case because $\mathcal{L}$ is a family of linearly independent circuits of H, and they all satisfy $x_e = x_h$. □

The following lemma characterizes which of the non-negativity and circuit inequalities are facet-defining.

Lemma 16. *The inequality*

1. $x_\ell \geq 0$ *defines a facet of* $B(G)$ *if and only if* ℓ *is not contained in a triangle.*
2. $x_\ell \leq x(C \setminus \ell)$ *defines a facet of* $B(G)$ *if and only if* C *has no chord and* $|C| \geq 3$.

Proof. By Observation 14, we may assume that G has no parallel edges. We prove both results by a maximality argument.

(1) First, suppose that ℓ is contained in a triangle, say $\{\ell, e, f\}$. The two circuit inequalities $x_e - x_f - x_\ell \leq 0$ and $-x_e + x_f - x_\ell \leq 0$ give $x_\ell \geq 0$ so the latter is not facet-defining.

Suppose now that ℓ is not contained in a triangle. Consider the face F defined by $x_\ell \geq 0$ and suppose that it is not a facet, that is, there exists a face F' defined by an inequality $ax \leq b$ of $B(G)$ such that $F \subseteq F'$. Since $\emptyset \in F$, we have $b = 0$. For every edge uv non incident to ℓ, the bonds $\delta(u)$, $\delta(v)$ and $\delta(\{u, v\})$ belong to F, implying that $a(\delta(u)) = a(\delta(v)) = a(\delta(\{u, v\})) = 0$, leading to $a_{uv} = 0$. Finally, for any edge $f = uv$ incident to ℓ at node u, $\delta(v) \in F$. By hypothesis, f is the only edge of $\delta(v)$ incident to ℓ, implying that $a_f = 0$. Thus, $a = \rho\chi_\ell$, for some $\rho < 0$ and F defines a facet.

(2) If $|C| = 2$, then C is two parallel edges e and f, and $B(G) \subseteq \{x_e = x_f\}$. If C has a chord c, let C' and C'' be the two circuits defined by $C' \cup C'' \setminus c = C$, and assume $c \in C'$. Then, the inequality $x_\ell \leq x(C \setminus \ell)$ is obtained applying the circuit inequalities for ℓ and C' and then for c and C'', $x_\ell \leq x(C' \setminus \ell) = x(C' \setminus \{\ell, c\}) + x_c \leq x(C' \setminus \{\ell, c\}) + x(C'' \setminus c) = x(C \setminus \ell)$.

Suppose that C has no chord, $|C| \geq 3$, and $F' = \{x_\ell \leq x(C \setminus \ell)\} \subseteq \{ax \leq b\} = F$, where F is facet-defining. Since $0 \in F'$, we have $b = 0$. Let $uv \in E$ with $u, v \notin V(C)$. Since $\{uv\} = (\delta_G(u) \cup \delta_G(v)) \setminus \delta_G(\{u, v\})$, and $\delta_G(u), \delta_G(v), \delta_G(\{u, v\})$ are bonds, and are contained in F', we have

$$(\#)\ a_{uv} = 0, \text{ for all } uv \in E \text{ such that } u, v \notin V(C).$$

Denote the vertices of C by $\{v_1, \ldots, v_k\}$ where $\ell = v_k v_1$ and $v_i v_{i+1} \in C$ for $i = 1, \ldots, k-1$. Let $u \in V \setminus V(C)$. Note that there are at most two edges between u and $\{v_1, \ldots, v_k\}$. If there is exactly one, say uv_i, then, by $(\#)$ and $\delta_G(u) \in F'$, we have $a_{uv_i} = 0$. If there are two, say uv_i and uv_j, since G is series–parallel, every uv_i-path not containing v_j does not intersect $V(C)$. Therefore, since G is 2-connected, there exists a bond $B = \delta_G(W)$ containing uv_i and ℓ such that $B' = \delta_G(W \cup \{u\})$ is also a bond. Since the edges of $B \Delta B'$ are uv_i, uv_j, and edges not in $\delta_G(C)$, and then by $B, B' \in F'$ and $(\#)$, we get $a_{uv_i} = a_{uv_j}$. By $\delta_G(u) \in F'$, we have $a_{uv_i} + a_{uv_j} = 0$. Therefore, $a_{uv_i} = a_{uv_j} = 0$. Since C had no chord, we proved $a_{uv} = 0$ whenever $uv \notin C$.

To finish the proof, since G is 2-connected, there exists a bond B_i containing ℓ and $v_i v_{i+1}$ for all $i = 1, \ldots, k-1$. By Observation 2, $B_i \cap C = \{\ell, v_i v_{i+1}\}$, thus $B_i \in F'$. Therefore, we have $a_\ell = -a_f$ for all $f \in C \setminus \ell$. Since $ax \leq 0$ is valid for the bond $\delta_G(v_2)$, we have $0 \geq a_{v_1 v_2} + a_{v_2 v_3} = 2a_{v_1 v_2}$, thus we may assume $a_\ell = 1$, and then we get $F' = F$. □

We now provide a family of series–parallel graphs where an exponential number of ear-cycle constraints are facet-defining.

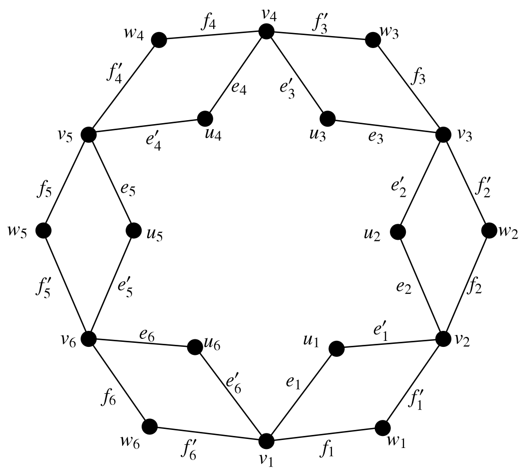

Fig. 1. An example of graph obtained from C_6 by parallel addition and subdivision of all the edges.

Example

The graph G_k we consider is built from the circuit on k edges C_k where a parallel edges is added to every edge and then all edges are subdivided. Fig. 1 shows the construction of such a graph from C_6. Denote by e_i and e'_i (resp. f_i and f'_i) the edges obtained by subdividing one parallel edge (resp. the other one). Let u_i (resp. w_i) be the vertex incident to e_i and e'_i (resp. f_i and f'_i) for all $i = 1, \dots, k$.

Claim 17. *$x(C) \le 2$ is facet-defining for $B(G_k)$ if C is a circuit of $2k$ edges.*

Proof. Without loss of generality, suppose that $C = \{e_i, e'_i : i = 1, \dots, k\}$. Let F' be the face induced by the inequality and suppose that $F' \subseteq F$ where F is a facet induced by the constraint $ax \le b$. Since $\{e_i, e'_i\} \in F'$, we have $a_{e_i} + a_{e'_i} = b$ for $i = 1, \dots, n$. Moreover, $\{e_i, f_i, f_j, e_j\}$ and $\{e'_i, f_i, f_j, e_j\}$ belong to F', for all $j \ne i$, from which we get $a_{e_i} = a_{e'_i}$. Combining these two remarks give $a_{e_i} = a_{e'_i} = b/2$, for $i = 1, \dots, n$. Now, since both $\{e_i, f_i, f_j, e_j\}$ and $\{e_i, f'_i, f_j, e_j\}$ belong to F', we have $a_{f_i} = a_{f'_i} = 0$, for $i = 1, \dots, n$. The emptyset being a bond, we have $b \ge 0$. In fact, $b > 0$ because otherwise $(a, b) = 0$. Therefore, without loss of generality, we may assume that $b = 2$. Then, we get $F = F'$, and we are done. □

If we set $E_i = \{e_i, e'_i\}$ and $F_i = \{f_i, f'_i\}$ for all $i = 1, \dots, k$, one can also prove that the inequalities $x(E_j \cup F_j) + \sum_{i \ne j}(x(M_i) - x(E_i \cup F_i \setminus M_i)) \le 2$ for all $j \in \{1, \dots, k\}$, where $M_i \in \{E_i, F_i\}$ for all $i = 1, \dots, k$ are facet-defining. In fact, together with the inequalities of Claim 17 and Lemma 16, this gives all the facet-defining inequalities for the example. However, other examples show that ear-cycle constraints are not always that nicely structured.

Let us mention some dual consequences of the results of Section 3.3 for the circuit polytope on series–parallel graphs.

First, we get its dimension by planar duality and Lemma 15.

Corollary 18. *Let F be a minimal set of edges intersecting every size two bond of $G = (V, E)$. Then, the dimension of $C(G)$ is $|E \setminus F|$.*

Moreover, Claim 17 implies the following.

Corollary 19. *There are examples for which exponentially many of the star-cut constraints (3) define facets of the circuit polytope on series–parallel graphs.*

Thus, the circuit polytope on series–parallel graphs is another example of polytope having exponentially many facet-defining inequalities that admits a compact extended formulation.

Acknowledgments

The authors thank the anonymous referees for their valuable comments. They also thank Denis Cornaz for pointing out the link between a bond J [24] and a circuit C in planar graphs.

References

[1] M.R. Garey, D.S. Johnson, R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, SIAM J. Comput. 5 (4) (1976) 704–714.
[2] P. Bauer, The circuit polytope: facets, Math. Oper. Res. 22 (1) (1997) 110–145.
[3] C.R. Coullard, W.R. Pulleyblank, On cycle cones and polyhedra, Linear Algebra Appl. 114–115 (1989) 613–640.
[4] P.D. Seymour, Sums of circuits, in: J.A. Bondy, U.S.R. Murty (Eds.), Graph Theory and Related Topics, Academic, New York, 1979, pp. 341–355.
[5] P. Bauer, J.T. Linderoth, M.W.P. Savelsbergh, A branch and cut approach to the cardinality constrained circuit problem, Math. Program. Ser. A 91 (2002) 307–348.
[6] V. Kaibel, R. Stephan, On cardinality constrained cycle and path polytopes, Math. Program. 123 (2) (2010) 371–394.
[7] V.H. Nguyen, J.-F. Maurras, On the linear description of the 3-cycle polytope, European J. Oper. Res. 13 (2) (2002) 310–325.
[8] V.H. Nguyen, J.-F. Maurras, On the linear description of the k-cycle polytope, PC_n^k, Int. Trans. Oper. Res. 8 (2001) 673–692.
[9] A.R. Mahjoub, On the stable set polytope of a series-parallel graph, Math. Program. 40 (1988) 53–57.
[10] S. Chopra, The graph partitioning polytope on series-parallel and 4-wheel free graphs, SIAM J. Discrete Math. 7 (1) (1994) 16–31.
[11] M. Baïou, A.R. Mahjoub, Steiner 2-edge connected subgraph polytopes on series-parallel graphs, SIAM J. Discrete Math. 10 (3) (1997) 505–514.
[12] G. Cornuéjols, J. Fonlupt, D. Naddef, The travelling salesman problem on a graph and some related integer polyhedra, Math. Program. 33 (1985) 1–27.
[13] M. Didi Biha, A.R. Mahjoub, k-edge connected polyhedra on series-parallel graphs, Oper. Res. Lett. 19 (2) (1996) 71–78.
[14] M. Baïou, A.R. Mahjoub, The Steiner traveling salesman polytope and related polyhedra, SIAM J. Optim. 13 (2) (2002) 498–507.
[15] M. Conforti, G. Cornuéjols, G. Zambelli, Extended formulations in combinatorial optimization, Ann. Oper. Res. 204 (1) (2013) 97–143.
[16] R. Kipp Martin, R.L. Rardin, B.A. Campbell, Polyhedral characterization of discrete dynamic programming, Oper. Res. 38 (1990) 127–138.
[17] T. Rothvoß, The matching polytope has exponential extension complexity, Comput. Res. Reposit. (2013).
[18] E. Balas, Disjunctive programming: properties of the convex hull of feasible points, Discrete Appl. Math. 89 (1998) 1–44.
[19] E. Balas, Disjunctive programming and a hierarchy of relaxations for discrete optimization problems, SIAM J. Algebraic Discrete Methods 6 (1985) 466–486.
[20] A. Chakrabarti, L. Fleischer, C. Weibel, When the Cut Condition is Enough: A Complete Characterization for Multiflow Problems in Series-Parallel Networks, Proceedings of the 44th symposium on Theory of Computing STOC'12 (2012) 19–26.
[21] J.B.J. Fourier, Solution d'une question particulière du calcul des inégalités, Nouveau bulletin des sciences par la société philomatique de Paris (1826) 317–319.
[22] D. Eppstein, Parallel recognition of series-parallel graphs, Inf. Comput. 98 (1) (1992) 41–55.
[23] R.J. Duffin, Topology of series-parallel networks, J. Math. Anal. Appl. 10 (1965) 303–318.
[24] I. Fleming, Casino Royale, Jonathan Cape, (1953).

RAIRO-Oper. Res. 52 (2018) 923–934
https://doi.org/10.1051/ro/2018035

RAIRO Operations Research
www.rairo-ro.org

THE *ST*-BOND POLYTOPE ON SERIES-PARALLEL GRAPHS

Roland Grappe and Mathieu Lacroix[*]

Abstract. The *st*-bond polytope of a graph is the convex hull of the incidence vectors of its *st*-bonds, where an *st*-bond is a minimal *st*-cut. In this paper, we provide a linear description of the *st*-bond polytope on series-parallel graphs. We also show that the *st*-bond polytope is the intersection of the *st*-cut dominant and the bond polytope.

Mathematics Subject Classification. 90C27, 90C35 and 90C57

Received October 4, 2016. Accepted May 2, 2018.

1. Introduction

In combinatorial optimization, *st*-bonds are well-known objects since they are precisely the *st*-cuts involved in the famous max-flow min-cut theorem [11]. However, they are not well described from a polyhedral point of view. In this paper, we make a contribution in this regard by providing a linear description of the *st*-bond polytope on series-parallel graphs. These graphs are precisely those with no K_4-minor [8].

In an undirected graph $G = (V, E)$, a *cut* is the set of edges between a subset of vertices and its complement. A cut containing only itself and the emptyset as cuts is called a *bond*. Nonempty bonds are the cuts whose removal yields exactly two connected components.

Although cuts and bonds are similar, they behave quite differently. From a complexity point of view, optimizing over bonds is harder than optimizing over cuts. The maximum cut problem – which asks for a cut of maximum weight in an edge-weighted graph – is NP-hard in general [13] but becomes polynomial for graphs with no K_5-minor [15], and hence for planar graphs [16]. In comparison, finding a bond of maximum weight in a planar graph is already NP-hard but becomes polynomial for the subclass of graphs with no K_4-minor. Indeed, bonds are planar duals of circuits, and finding a circuit of maximum weight is NP-hard for planar graphs [14] and polynomial for graphs with no K_4-minor [5].

The polyhedral aspects of these problems reflect these complexity results. The *cut polytope* and the *bond polytope* of a graph are the convex hulls of the incidence vectors of its cuts and its bonds, respectively. The cut polytope has been described for graphs with no K_5-minor in [3] whereas a description of the bond polytope is only known for graphs with no K_4-minor [5].

Given two distinct vertices s and t, an *st-cut* and an *st-bond* are respectively a cut and a bond whose removal disconnects s and t. It follows that *st*-bonds are inclusionwise minimal *st*-cuts. The *st-cut polytope* and the *st-bond polytope* are the convex hull of the incidence vectors of *st*-cuts and *st*-bonds, respectively.

Keywords and phrases: Bond, minimal *st*-cut, *st*-bond polytope, series-parallel graph.

Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, UMR 7030, Villetaneuse, France.

[*] Corresponding author: Mathieu.Lacroix@lipn.univ-paris13.fr

The behaviour of st-cuts and st-bonds is different from a complexity point of view. For instance, consider the class of planar graphs where s and t belong to a same face, hereafter called *st-planar graphs*. In these graphs, finding an st-bond of maximum weight is NP-hard whereas an st-cut of maximum weight can be found in polynomial time. Indeed, a nonempty bond is an st-bond for some pair of vertices s and t on a same face. Hence, finding an st-bond of maximum weight is at least as hard as finding a bond of maximum weight in a planar graph. Computing an st-cut of maximum weight is polynomial-time solvable because adding an edge st with a sufficiently large weight reduces the problem to a maximum cut problem in a graph with no K_5-minor.

From a polyhedral point of view, remark that when the graph contains the edge st, the st-cut (resp. st-bond) polytope is a face of the cut (resp. bond) polytope. Hence, a polyhedral description for the st-cut (resp. st-bond) polytope is known for graphs with no K_5-minor (resp. K_4-minor) but containing the edge st. In this paper, we extend this polyhedral characterization by describing the st-bond polytope for graphs with no K_4-minor for any pair of vertices s and t.

By definition, the st-bond polytope is contained in the st-cut polytope. Nevertheless, these two polyhedra have the same dominant. This so-called *st-cut dominant* has been thoroughly studied in [18].

In general, the intersection of two integer polyhedra is not integer. Even adding a simple constraint to a well-structured polyhedron may destroy its integrality – see *e.g.* [6]. Few families of polyhedra are known to remain integer after intersection. For instance, box-TDI polyhedra keep their integrality after being intersected with any integer box [17]. The intersection of two polymatroids [9] or of two lexicographical polytopes [4] is integer. Calvillo characterizes the graphs whose stable set polytope remains integer after being intersected with specific hyperplanes [6]. For st-planar graphs, the intersection of the cut polytope [3] and the st-cut dominant [18] is integer: it is nothing but the st-cut polytope. For series-parallel graphs, we prove that the st-bond polytope is the intersection of the bond polytope and the st-cut dominant.

The contributions of this paper concern st-bonds in series-parallel graphs. We prove that finding an st-bond of maximum weight in a series-parallel graph can be performed in polynomial time. Moreover, we provide a polyhedral description of the st-bond polytope for such graphs. The approach is as follows. We describe the st-bond polytope on the subclass of 2-connected outerplanar graphs. We also provide a sufficient condition for a series-parallel graph to be outerplanar. We exploit this result to extend this polyhedral description of the st-bond polytope to series-parallel graphs. A consequence of our result is that intersecting the bond polytope and the st-cut dominant of a series-parallel graph does not destroy integrality: it precisely gives the st-bond polytope.

The outline of the paper is as follows. In Section 2, we first give the main definitions and notation used in this paper. We then show that a simple series-parallel graph with two vertices of degree at most two is outerplanar. We use it to show that finding an st-bond of maximum weight in series-parallel graphs can be done in polynomial time. In Section 3, we describe the st-bond polytope for 2-connected outerplanar graphs. In Section 4, we extend this description to series-parallel graphs. As a consequence, we prove that, for these graphs, the st-bond polytope is the intersection of the bond polytope and the st-cut dominant.

2. Bonds and st-bonds in series-parallel graphs

Throughout the paper, $G = (V, E)$ will denote a loopless connected undirected graph and s and t two distinct vertices of G. Given a vertex v of G, the graph $G \setminus v$ is obtained from G by removing the vertex v and its incident edges. Let uv be an edge of G. The graph G/uv is the graph obtained from G by *contracting* uv, that is, by identifying u and v into a new vertex which becomes adjacent to all the former neighbors of u and v. We define $G \setminus uv = (V, E \setminus \{uv\})$. Let F be a subset of E. A set of vertices is *covered* by F if each of its vertices is incident to at least one edge of F. Given a subgraph H of G, the set $F|_H$ denotes the set of edges of F which belong to H. Given two sets A and B, the symmetric difference $A\Delta B$ is $A\Delta B = (A \cup B) \setminus (A \cap B)$.

A subset of edges is called a *cycle* if it induces a graph in which every vertex has even degree. A *circuit* is a cycle inducing a connected graph in which every vertex has degree two. An *st-circuit* is a circuit covering both s and t. A *uv-path* P is a set of edges inducing a connected subgraph in which u and v have degree one and the other vertices have degree two. The vertices u and v are called *end vertices* of P whereas the other vertices covered by P are called *internal vertices.* A *path* is a uv-path for some distinct vertices u and v. Let P be an uv-path and W a set of vertices covered by P. Running P from u to v induces a (total) order on W which corresponds to the order in which the vertices of W are traversed by P starting from u. This order is called the *order of* W *induced by* P *starting from* u.

A subset F of E is a *cut* if it is the set of edges having exactly one extremity in X, for some $X \subseteq V$; it is denoted by $F = \delta(X)$. If $s \in X$ and $t \in V \setminus X$, the cut $\delta(X)$ is an st*-cut* and *separates* s and t. A nonempty *bond* is a cut containing no other nonempty cut. Equivalently, a nonempty bond is a cut whose removal gives exactly two connected components. An st*-bond* is an st-cut which is a bond. A *bridge* is an edge whose removal disconnects the graph. Equivalently, a bridge is a bond composed of a single edge.

A vertex whose removal yields several connected components is a *cut vertex.* If s and t belong to different connected components after the removal of a cut vertex v, then v is an st*-cut vertex.* When no vertex removal disconnects a connected graph with at least two vertices, the latter is said 2*-connected.* The 2*-connected components* of a graph are the inclusionwise maximal 2-connected subgraphs of the graph. A 2-connected graph composed of a single edge is called *trivial.* Note that the trivial 2-connected components of a graph $G = (V, E)$ are the graphs induced by the bridges of E.

2.1. Series-parallel and outerplanar graphs

A nontrivial 2-connected graph is *series-parallel* if it can be built, starting from the circuit of length two, by repeatedly applying the following operations:

- *parallel addition:* add a parallel edge to an existing edge,
- *subdivision:* replace an edge uv by the path $\{uw, wv\}$ where w is a new vertex.

A graph is *series-parallel* if all its nontrivial 2-connected components are.

Series-parallel graphs are also known to be those admitting the following kind of decomposition. An *open nested ear decomposition* [10] of a graph $G = (V, E)$ is a partition $\mathcal{E}$ of E into a sequence $E_0, \dots, E_k$ such that E_0, also denoted by $C_{\mathcal{E}}$, is a circuit of G and the *ears* E_i, for $i = 1, \dots, k$, are paths with the following properties:

- the end vertices of E_i are both covered by E_j for some $j < i$,
- no internal vertex of E_i is covered by E_j for all $j < i$,
- if two ears E_i and $E_{i'}$ have both their end vertices in the same member E_j of $\mathcal{E}$, then there exit a path P_i in E_j between the end vertices of E_i and a path $P_{i'}$ in E_j between the end vertices of $E_{i'}$ such that P_i and $P_{i'}$ are either disjoint or contained one in another.

A nontrivial 2-connected graph is series-parallel if and only if it admits an open nested ear decomposition [10].

An important subclass of series-parallel graphs are *outerplanar* graphs. They are the graphs admitting a planar drawing such that all the vertices lie on the unbounded face of the drawing, called the *external* face. An outerplanar graph is always supposed to be drawn in this way and the same terminology is used for the graph and the drawing. The *chords* are the edges which do not belong to the external face. In a simple outerplanar graph, the external face and the chords are uniquely defined. The faces other than the external one are called *inner faces.*

Lemma 2.2 provides a sufficient condition for a series-parallel graph to be outerplanar. We first give the following lemma since it is used in the proof of Lemma 2.2. In both proofs, we implicitly use that deleting a parallel edge or contracting an edge incident to a vertex of degree two in a graph preserves both 2-connectivity and series-parallelness.

Lemma 2.1. *A nontrivial simple* 2*-connected series-parallel graph has at least two vertices of degree two.*

Proof. Let $G = (V, E)$ be a counter example with a minimum number of vertices, that is, G is a nontrivial simple 2-connected series-parallel graph with at most one vertex of degree two. By construction of nontrivial 2-connected series-parallel graphs, G, being simple, has exactly one vertex of degree two, say v. Let xv and vy be its incident edges. Contracting xv preserves the degrees in $V \setminus \{v\}$ and maintains 2-connectivity and series-parallelness. Hence, by the minimality of G, the graph $G' = G/xv$ is not simple, and thus $xy \in E$. Thus, $G \setminus v$ is a simple 2-connected series-parallel graph with less vertices than G. By the minimality assumption, $G \setminus v$ has at least two vertices of degree two. All the vertices except x and y have the same degree in G and $G \setminus v$, so they all are of degree greater than two. Therefore, the degrees of x and y equal two in $G \setminus v$ and hence three in G. But then $(G \setminus v)/xy$ is a counter example – a contradiction to the minimality of G. □

Lemma 2.2. *A simple series-parallel graph having exactly two vertices of degree at most two is outerplanar.*

Proof. Let κ_G denote the number of vertices of degree at most two of a graph G.

We first prove the assertion for nontrivial 2-connected graphs. Let $G = (V, E)$ be a counter example with a minimum number of vertices, that is, G is a nontrivial simple 2-connected series-parallel graph with $\kappa_G = 2$ but is not outerplanar. Then, G differs from the circuit of length two. Since G is 2-connected, it has exactly two vertices of degree two. Let v be a vertex of degree two of G and xv and vy its incident edges. The graph $G' = G/xv$ is nontrivial, 2-connected and series-parallel. As $\kappa_{G'} = 1$, Lemma 2.1 implies that G' is not simple, and hence $xy \in E$. Hence, $G'' = G \setminus v$ is a simple 2-connected series-parallel graph. By Lemma 2.1, G'' has at least two vertices of degree two. Thus, by construction, at least one among x and y has degree two in G''. If G'' was outerplanar, then it would admit a planar drawing in which all its vertices, and hence also the edge xy, lie on the external face. But then G would be outerplanar – a contradiction. Hence, G'' is not outerplanar and, by the minimality assumption, G'' has at least three vertices of degree two. Since G has exactly two vertices of degree two, G'' has three vertices of degree two including x and y. Then, G''/xy is a counter example having fewer vertices than G – a contradiction. Thus, the assertion is proved for nontrivial 2-connected graphs. Note that the assertion also holds for trivial 2-connected graphs.

We now prove the result for simple series-parallel graphs. Given $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, identifying a vertex $v_1 \in V_1$ and a vertex $v_2 \in V_2$ yields a graph G satisfying $\kappa_G \geq \kappa_{G_1} + \kappa_{G_2} - 2$. Let G be a simple series-parallel graph with $\kappa_G = 2$ and $G_1, \ldots, G_k$ be its 2-connected components. By Lemma 2.1 and since $\kappa_G \geq \sum_{i=1}^{k} \kappa_{G_i} - 2(k-1)$, we deduce $\kappa_{G_i} = 2$ for $i = 1, \ldots, k$. Therefore, as proved above, each 2-connected component of G is outerplanar, hence so is G. □

2.2. Bonds and *st*-bonds

A well-known characterization of cuts is that they are the sets of edges intersecting every circuit an even number of times. Bonds being cuts, each bond intersects every circuit an even number of times. For series-parallel graphs, this property can be refined as follows.

Observation 2.3 (Chakrabarti *et al.* [7])**.** *In a series-parallel graph, a bond and a circuit intersect in zero or two edges.*

If the graph is also 2-connected, this property becomes a characterization of bonds.

Lemma 2.4 (Borne *et al.* [5])**.** *In a 2-connected series-parallel graph, a set of edges is a bond if and only if it intersects every circuit in zero or two edges.*

Let W be the set of st-cut vertices of G together with s and t. Then, every st-path of G covers W. Moreover, the order of W induced by running any st-path from s to t is the same. This order will be denoted by $v_1, v_2, \ldots, v_{|W|}$ throughout. Note that $s = v_1$ and $t = v_{|W|}$. Let $k = |W| - 1$ and ℓ be the number of 2-connected components of G. Let $H_1 = (V_1, E_1), \ldots, H_\ell = (V_\ell, E_\ell)$ be the 2-connected components of G numbered in such a way that, for $i = 1, \ldots, k$, $H_i = (V_i, E_i)$ contains v_i and v_{i+1}. Each bond separating two vertices of W being an st-bond, we obtain the following observation.

Observation 2.5. *The set of st-bonds of G is the union of the sets of $v_i v_{i+1}$-bonds of H_i over $i = 1, \ldots, |W| - 1$.*

By Observation 2.5, in order to solve the maximum st-bond problem, it is enough to solve the problem in each 2-connected component. In series-parallel graphs, this remark, combined with Lemma 2.2, gives the following complexity result.

Corollary 2.6. *Finding an st-bond of maximum weight in a series-parallel graph can be done in polynomial time.*

Proof. Let $G = (V, E)$ be a series-parallel graph with edge weights $w \in \mathbb{R}^E$. By Observation 2.5, we may suppose that G is 2-connected. We may suppose that G is nontrivial as otherwise it contains a unique bond. The following graph operations may be applied on G without changing the weight of a maximum st-bond. If e and f are two parallel edges, they can be replaced by a single edge with weight $w_e + w_f$. If e and f are in series, that is, there exists a vertex v of degree two incident to e and f, then the edge among e and f with the minimum weight may be contracted whenever v differs from s and t. By definition of series-parallelness, applying these two types of operations as long as possible yields a simple 2-connected series-parallel graph G' in which no vertex is of degree two except possibly s and t. Hence, by Lemmas 2.1 and 2.2, G' is outerplanar. From the definition of outerplanar graphs, the number of st-bonds in G' is quadratic in the number of its vertices. Thus, finding an st-bond of maximum weight in G' can be done in polynomial time by enumeration. Moreover, by construction, one can retrieve from such an st-bond one of G having the same weight. $\square$

3. The *st*-bond polytope on 2-connected outerplanar graphs

In this section, we give a linear description of the st-bond polytope in 2-connected outerplanar graphs. Let $G = (V, E)$ be such a graph. If G is trivial, then it only has s and t as vertices and its only edge is st. The st-bond polytope of G is then $\{x_{st} = 1\}$. We now suppose, for the rest of this section, that G is nontrivial. We denote by P_{st} and Q_{st} the two st-paths of the external face F_{ext}, that is, $F_{\text{ext}} = P_{st} \cup Q_{st}$. We denote by $B_{st}(G)$ the convex hull of the incident vectors of the st-bonds of G.

We denote by $P(G)$ the polytope defined by the set of $x \in \mathbb{R}^E$ satisfying the following inequalities:

$$
\begin{aligned}
&x_e \geq 0, && \text{for all } e \in E, && (3.1)\\
&x_e \leq x(F \setminus e), && \text{for all chords } e \text{ and faces } F \text{ containing } e, && (3.2)\\
&x(P) \geq 1, && \text{for all } st\text{-paths } P, && (3.3)\\
&x(P) = 1, && \text{for all } st\text{-paths } P \text{ contained in a circuit.} && (3.4)
\end{aligned}
$$

We now prove in the next three lemmas that $P(G)$ is a linear description of $B_{st}(G)$. More precisely, we show the validity of inequalities (3.1)–(3.4) for $B_{st}(G)$ in Lemma 3.1. We then show that every integer vector satisfying these inequalities is the incident vector of an st-bond in Lemma 3.2. Finally, we end the proof by showing that $P(G)$ is an integer polytope in Lemma 3.4.

Lemma 3.1. *$B_{st}(G) \subseteq P(G)$.*

Proof. We show that the incidence vectors of st-bonds satisfy (3.1)–(3.4). A bond being a cut and a face being a circuit, a bond cannot intersect a face in exactly one edge by Observation 2.3. This ensures the validity of (3.2). The validity of (3.3) stems from the fact that removing the edges of an st-bond separates s and t. If an st-path is contained in a circuit, then this circuit is the union of two disjoint st-paths. By (3.3), any st-bond intersects this circuit at least twice. Thus, by Observation 2.3, inequalities (3.4) are valid. $\square$

Lemma 3.2. *$P(G) \cap \mathbb{Z}^E \subseteq B_{st}(G)$.*

Proof. Let $\bar{x}$ be a point of $P(G) \cap \mathbb{Z}^E$. We need to show that it is the incident vector of an st-bond.

If the intersection of a circuit C and an inner face F is a chord of G, then, by (3.2), we have that $\bar{x}(C) = \bar{x}(C \setminus e) + \bar{x}_e \leq \bar{x}(C \setminus e) + \bar{x}(F \setminus e) = \bar{x}(C')$ with $C' = C \Delta F$. By repeating this argument, and since $\bar{x}(F_{\text{ext}}) = 2$ due to (3.4) associated with P_{st} and Q_{st}, we obtain that

$$\bar{x}(C) \leq 2, \quad \text{for each circuit } C. \tag{3.5}$$

Inequalities (3.2) can be generalized to any circuit and any edge of this circuit as shown in the following claim.

Claim 3.3. *$\bar{x}_e \le \bar{x}(C \setminus e)$ for all circuits C and for all edges $e \in C$.*

Proof. Let F be an inner face of G and e an edge of $F \cap F_{\text{ext}}$. Without loss of generality, we suppose that $e \in P_{st}$. Let $P = P_{st} \cup F \setminus e$. Then, P contains an st-path and by (3.1), (3.3) and (3.4), we have $0 \le \bar{x}(P) - \bar{x}(P_{st}) = \bar{x}(F \setminus P_{st}) - \bar{x}_e \le \bar{x}(F \setminus e) - \bar{x}_e$. This, together with (3.2), gives

$$\bar{x}_e \le \bar{x}(F \setminus e), \quad \text{for all inner faces } F \text{ and for all edges } e \in F. \tag{3.6}$$

In a planar graph, a circuit is the symmetric difference of inner faces. Suppose that a circuit C is the symmetric difference of two inner faces F_1 and F_2. Let f be the edge of $F_1 \cap F_2$ and let e be an edge of $F_1 \setminus F_2$. Adding inequalities (3.6) associated with e and F_1, and with f and F_2 gives $\bar{x}_e \le \bar{x}(C \setminus e)$. The face F_1 and the edge e have been chosen arbitrarily so the inequality holds for every circuit consisting in the symmetric difference of two inner faces and for every edge it contains. By induction on the number of inner faces contained in a circuit, we obtain the desired result. ☐

For any edge $e \in E$, let F be an inner face containing e. Combining inequality (3.5) associated with F and inequality (3.6) associated with F and e gives $\bar{x}_e \le 1$. Thus, by (3.1), $\bar{x}$ is the incidence vector of a subset of E, say $\bar{E}$.

Since $\bar{x}$ satisfies (3.3), removing the set of edges $\bar{E}$ disconnects s and t. To end the proof, it remains to show that $\bar{x}$ is the incident vector of a bond. Since outerplanar graphs are series-parallel, by Lemma 2.4, one just needs to show that $\bar{x}(C)$ equals 0 or 2 for each circuit C as $\bar{x}$ is a 0–1 vector. This is immediate by inequalities (3.5) and Claim 3.3. ☐

Lemma 3.4. *$P(G) = B_{st}(G)$.*

Proof. By Lemmas 3.1 and 3.2, it remains to show that $P(G)$ is an integer polytope. Suppose this is not true and let G be a counter example with a minimal number of edges. By hypothesis, there exists a fractional extreme point of $P(G)$, say $\bar{x}$. The minimality assumption implies that G has no parallel edges because two parallel edges have the same value in $\bar{x}$ by (3.2). In the following, an st-path P satisfying $\bar{x}(P) = 1$ will be called a *tight path*.

Claim 3.5. *$\bar{x}_e > 0$ for all the edges e of the external face.*

Proof. If $\bar{x}_e = 0$ for some $e \in F_{\text{ext}}$, then contract e. The resulting graph G/e is still outerplanar. By minimality, its st-bond polytope is described by (3.1)–(3.4), yet these are precisely the inequalities obtained by setting x_e to zero in $P(G)$, thus $\bar{x}$ yields a fractional extreme point in $P(G/e)$ – a contradiction to the minimality assumption. ☐

Claim 3.6. *No vertex of $V \setminus \{s, t\}$ has degree two.*

Proof. Suppose that $u \in V \setminus \{s, t\}$ is such that $\delta(u) = \{e, f\}$. Note that $e, f \in F_{\text{ext}}$, hence we may assume $e, f \in P_{st}$. By Claim 3.5 and equation (3.4) associated with P_{st}, we have $0 < \bar{x}_e < 1$ and $0 < \bar{x}_f < 1$. Let $\tilde{x}$ be the vector obtained from $\bar{x}$ by adding ϵ to $\bar{x}_e$ and $-\epsilon$ to $\bar{x}_f$ for some positive scalar ϵ. If ϵ is small enough, then $\tilde{x}$ belongs to P_G. As $\tilde{x}$ satisfies with equality all the inequalities (3.1)–(3.4) which are tight for $\bar{x}$, it contradicts the extremality of $\bar{x}$. ☐

Since G is a simple nontrivial 2-connected outerplanar graph, Lemma 2.1 and Claim 3.6 imply that s and t have degree two. Moreover, $st \notin E$ as otherwise G/st would have only one vertex of degree 2 – a contradiction to Lemma 2.1. Let $p = |E \setminus F_{\text{ext}}|$. Then, F_{ext} has p chords and G has $p+1$ inner faces. Number the faces in such a way that F_1 is the face containing s and $|F_i \cap F_{i+1}| = 1$ for $i = 1, \dots, p$. Then, F_{p+1} contains t. Moreover, F_1 and F_{p+1} are triangles and every inner face has at least one edge on the external face. Denote the chords of F_{ext} by $c_1, \dots, c_p$ such that $\{c_i\} = F_i \cap F_{i+1}$, for $i = 1, \dots, p$ – see Figure 1 for an illustration.

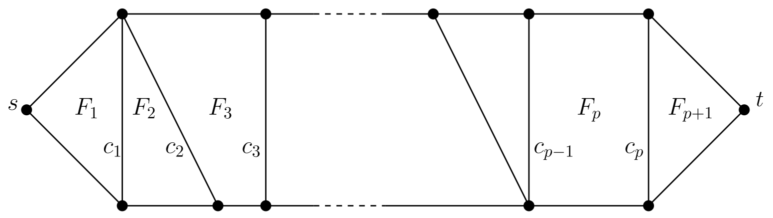

FIGURE 1. Numbered inner faces and chords.

Now, inequalities (3.2) can be rewritten as:

$$x_{c_j} \le x(F_j \setminus c_j), \qquad \text{for } j = 1 \dots, p, \tag{3.7}$$

$$x_{c_{j-1}} \le x(F_j \setminus c_{j-1}), \qquad \text{for } j = 2, \dots, p+1. \tag{3.8}$$

We have the following property.

Claim 3.7. *For each $j = 2, \dots, p$, at most one inequality among (3.7) and (3.8) associated with j is tight for $\bar{x}$.*

Proof. By contradiction, suppose there exists $j \in \{2, \dots, p\}$ such that $\bar{x}_{c_j} = \bar{x}(F_j \setminus c_j)$ and $\bar{x}_{c_{j-1}} = \bar{x}(F_j \setminus c_{j-1})$. Then $\bar{x}_e = 0$ for all $e \in F_j \setminus \{c_{j-1}, c_j\}$. Since F_j contains at least an edge of the external face, this contradicts Claim 3.5. □

Let k be the maximum integer such that inequality (3.7) is satisfied with equality for every $j = 1, \dots, k$. Set $k = 0$ if inequality (3.7) associated with $j = 1$ is not tight. Similarly, let ℓ be the minimum number such that inequality (3.8) is satisfied with equality for every $j = \ell, \dots, p+1$. By Claim 3.7, we have $k < \ell$.

Claim 3.8. *For $j \in \{1, \dots, k\} \cup \{\ell - 1, \dots, p\}$, we have $\bar{x}_{c_j} > 0$ and c_j belongs to no tight path.*

Proof. Since every inner face intersects the external face, the first part stems from Claim 3.5 and the tightness of inequalities (3.7) for $j = 1, \dots, k$ and (3.8) for $j = \ell, \dots, p+1$. We now prove the second part of the assertion. Suppose it is not true for some $j \in \{1, \dots, k\}$. Let $1 \le j' \le k$ be the minimum index such that $c_{j'}$ is in a tight path P. If $j' = 1$, then $1 = \bar{x}(P) = \bar{x}(P \setminus c_1) + \bar{x}(F_1 \setminus c_1)$ because $\bar{x}_{c_1} = \bar{x}(F_1 \setminus c_1)$. Since $(P \cup F_1) \setminus c_1$ is the disjoint union of an st-path and an edge of the external face, this contradicts (3.3) by Claim 3.5. Thus, $j' \ge 2$. But now, $1 = \bar{x}(P) = \bar{x}(P \setminus c_{j'}) + \bar{x}(F_{j'} \setminus c_{j'}) \ge \bar{x}(P') \ge 1$, where $P' \subseteq P \cup F_{j'} \setminus c_{j'}$ is an st-path containing $c_{j'-1}$. This implies that $c_{j'-1}$ is in a tight path – a contradiction to the minimality of j'. The case $j \in \{\ell - 1, \dots, p\}$ can be proved similarly. □

Let f_s be an edge of $F_{k+1} \cap F_{\text{ext}}$. Without loss of generality, we assume $f_s \in P_{st}$. Let X_s be the set of vertices of the component of P_{st} containing s after the removal of f_s. Note that $\delta(X_s) = \{c_1, \dots, c_k, f_s, g_s\}$, where g_s is the edge incident to s which does not belong to P_{st}. By construction of X_s and by Claim 3.8, every tight path of G intersects $\delta(X_s)$ exactly once.

We define a vertex set X_t with respect to t in a similar way. Then, $\delta(X_t) = \{c_{\ell-1}, \dots, c_p, f_t, g_t\}$, where $f_t \in F_{\ell-1} \cap F_{\text{ext}}$ and $g_t \in \delta(t)$. Moreover, every tight path of G intersects $\delta(X_t)$ exactly once. We refer the reader to Figure 2 for an example of X_s and X_t.

Given an edge set $F \subseteq E$, we denote by χ^F its incidence vector. Let $z = \chi^{\delta(X_s)} - \chi^{\delta(X_t)}$. By construction, $z(P) = 0$ for all tight paths. Furthermore, all the inequalities (3.7) except the one associated with $j = k+1$ and all the inequalities (3.8) are satisfied with equality by $\chi^{\delta(X_s)}$. Moreover, all the inequalities (3.8) except the one associated with $j = \ell - 1$ and all the inequalities (3.7) are tight for $\chi^{\delta(X_t)}$. Hence, z satisfies with equality all the inequalities (3.7) and (3.8) except two: the inequality (3.7) associated with $k+1$ and the inequality (3.8) associated with $\ell - 1$. Since none of these two inequalities is tight for $\bar{x}$, and since $\bar{x}_e > 0$ for all

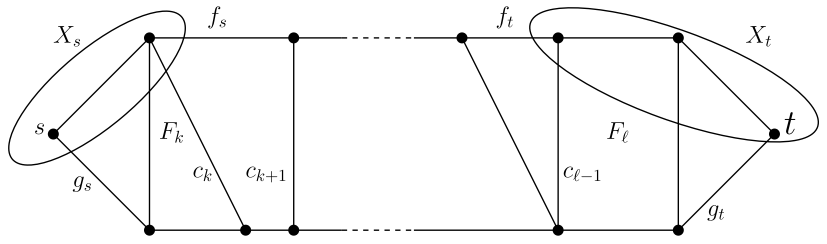

FIGURE 2. Example of X_s and X_t.

$e \in \delta(X_s) \cup \delta(X_t)$ by Claims 3.5 and 3.8, the two points $\bar{x} \pm \epsilon z$ belong to $P(G)$ for some $\epsilon > 0$ small enough. As $z \neq \mathbf{0}$, this contradicts the extremality of $\bar{x}$. □

4. The st-bond polytope on series-parallel graphs

We start this section by describing the st-bond polytope on 2-connected series-parallel graphs, see Section 4.1. We then extend this description to general series-parallel graphs, see Section 4.2. We conclude by showing that for these graphs, the st-bond polytope is actually the intersection between the bond polytope and the st-cut dominant, see Section 4.3. Recall that $B_{st}(G)$ denotes the convex hull of the incidence vectors of the st-bonds of G. We define $B(G)$ as the convex hull of the incidence vectors of the bonds of G.

4.1. 2-Connected series-parallel graphs

An ear of an open nested ear decomposition $\mathcal{E}$ whose circuit $C_{\mathcal{E}}$ is an st-circuit is called an *st-ear.*

Lemma 4.1. *Given a 2-connected series-parallel graph $G = (V, E)$ and distinct vertices s and t, the st-bond polytope $B_{st}(G)$ is the set of $x \in \mathbb{R}^E$ satisfying the following inequalities.*

$$x_e \geq 0, \quad \text{for all } e \in E, \tag{4.1}$$
$$x(Q) \leq x(C \setminus Q), \quad \text{for all } st\text{-ears } Q \text{ and circuits } C \supseteq Q, \tag{4.2}$$
$$x(P) \geq 1, \quad \text{for all } st\text{-paths } P, \tag{4.3}$$
$$x(P) = 1, \quad \text{for all } st\text{-paths } P \text{ contained in a circuit}, \tag{4.4}$$
$$x_{st} = 1, \quad \text{whenever } st \in E. \tag{4.5}$$

Proof. The result holds if G is trivial because $B_{st}(G) = \{x_{st} = 1\}$. Suppose now that G is nontrivial. Equation (4.5) can then be removed because if st is an edge of G, then it is an st-path of G contained in a circuit and the equation is of type (4.4). The result also holds when G is the circuit of length two, say $\{e, f\}$, since $B_{st}(G) = \{x_e = x_f = 1\}$.

Let us prove the result by induction on the number of edges of G. Assume that the bond polytope for every 2-connected series-parallel graph with less edges than G is given by (4.1)–(4.4).

First, suppose that G has two parallel edges g and h. By induction, $B_{st}(G \setminus h)$ is given by (4.1)–(4.4). Remark that $x_g = x_h$ for all $x \in B_{st}(G)$ as every bond contains either none of g and h or both of them. This equation is given in (4.1)–(4.4) for G by the two inequalities (4.2) associated with the circuit $C = \{g, h\}$. Moreover, for any other inequality $ax \leq b$, at most one among a_g and a_h is nonzero. Finally, the system (4.1)–(4.4) is such that for every inequality $ax \leq b$, there exists another $a'x \leq b$ where a' is obtained from a by exchanging a_g and a_h. These remarks imply that $B_{st}(G)$ is described by (4.1)–(4.4).

Assume now that G has no parallel edge. By Lemma 2.1, G has at least two vertices of degree two. Suppose that there are exactly two such vertices. Then G is outerplanar by Lemma 2.2. In this case, in the open nested ear decomposition $\mathcal{E}$ with $C_{\mathcal{E}} = F_{\text{ext}}$, the st-ears are precisely the chords of the graph. Thus, inequalities (4.2) contain inequalities (3.2). We now prove the validity of (4.2) for $B_{st}(G)$. Let Q be any st-ear and let $\mathcal{E}$ be an open nested ear decomposition containing Q such that $C_{\mathcal{E}}$ is an st-circuit. For each $e \in C_{\mathcal{E}}$, since G is a 2-connected series-parallel graph, there exists a circuit containing both e and Q. By Observation 2.3, a bond intersecting twice Q does not intersect $C_{\mathcal{E}}$, which implies that it is not an st-bond. Observation 2.3 gives the validity of (4.2). Therefore, the result follows from Lemma 3.4.

The last case to consider is when G has at least three vertices of degree two. Then, one of them, say v, is different from s and t. Let f and g be the two edges incident to v and H be the 2-connected series-parallel graph obtained from G by replacing the path $\{f, g\}$ by a single edge e. By the induction hypothesis, $B_{st}(H)$ is described by (4.1)–(4.4). Let Q be the polytope obtained by replacing x_e by $x_f + x_g$ in $B_{st}(H)$, and adding $x_f \geq 0$ and $x_g \geq 0$. This operation preserves integrality. Indeed, let $\bar{z}$ be a vertex of Q. At least one of $\bar{z}_f$ and $\bar{z}_g$ equals zero, as otherwise the points $\bar{z} \pm \epsilon(\chi^f - \chi^g)$ both belong to Q for some $\epsilon > 0$. Without loss of generality, assume that $\bar{z}_g = 0$. Then, the point $\bar{x}$ defined by $\bar{x}_h = \bar{z}_h$ for $h \neq e$ and $\bar{x}_e = \bar{z}_f$ belongs to $B_{st}(H)$. By definition of Q and since $x_e \geq 0$ is an inequality of $B_{st}(H)$, $\bar{x}$ is a vertex of $B_{st}(H)$. As $B_{st}(H)$ is an integer polytope, $\bar{x}$ is integer and so is $\bar{z}$. This implies that Q is an integer polytope.

Furthermore, the st-bonds of G are obtained from those of H as follows: keep the st-bonds of H not containing e, and for every st-bond B of H containing e, take $B \setminus e \cup f$ and $B \setminus e \cup g$. This implies that $Q = B_{st}(G)$. Given the form of inequalities (4.1)–(4.4), this proves the result. □

4.2. General case

For the rest of the paper, $G = (V, E)$ will denote a series-parallel graph and W the set of st-cut vertices of G together with s and t. Recall that the vertices of W may be ordered according to the order induced by running any st-path from s to t. This order is denoted by $s = v_1, v_2, \ldots, v_{|W|} = t$. Let $k = |W| - 1$ and ℓ be the number of 2-connected components of G. Let $H_1 = (V_1, E_1), \ldots, H_\ell = (V_\ell, E_\ell)$ be the 2-connected components of G numbered in such a way that, for $i = 1, \ldots, k$, $H_i = (V_i, E_i)$ contains v_i and v_{i+1}. Observation 2.5 may then be translated in terms of polytopes as follows.

Observation 4.2. $B_{st}(G) = conv\left(\bigcup_{i=1}^{k} B_{v_i v_{i+1}}(H_i)\right)$.

The following observation stems from the definition of the 2-connected components $H_1, \ldots, H_k$.

Observation 4.3. *Every st-path of G is the union of $v_i v_{i+1}$-paths of H_i over $i = 1, \ldots, k$, and conversely.*

We now provide a polyhedral description of the st-bond polytope on general series-parallel graphs, see Theorem 4.4. The result is a consequence of Lemma 4.1, Observations 4.2 and 4.3, and the theorem of Balas [1, 2] on the union of polyhedra.

Theorem 4.4. *Let $G = (V, E)$ be a series-parallel graph and s and t be distinct vertices of V. The st-bond polytope $B_{st}(G)$ is the set of $x \in \mathbb{R}^E$ satisfying the following inequalities.*

$$x_e = 0, \quad \text{for all } e \text{ which belongs to no } st\text{-path}, \tag{4.6}$$

$$x_e \geq 0, \quad \text{for all } e \in E, \tag{4.7}$$

$$x(Q) \leq x(C \setminus Q), \quad \text{for all } v_i v_{i+1}\text{-ears } Q \text{ and circuits } C \supseteq Q \text{ of } H_i,\ i = 1, \ldots, k, \tag{4.8}$$

$$x(P) \geq 1, \quad \text{for all } st\text{-paths } P \text{ of } G, \tag{4.9}$$

$$x(P) \leq 1, \quad \text{for all } st\text{-paths } P \text{ of } G \text{ such that } P|_{H_i} \text{ is contained in a circuit for all } i = 1, \ldots, k \text{ with } H_i \text{ nontrivial.} \tag{4.10}$$

Proof. First, note that every edge of an st-bond belongs to an st-path, which implies the validity of inequalities (4.6). Moreover, the edges involved in (4.6) are precisely those which belong to $H_{k+1}, \ldots, H_\ell$. From now on, we may suppose that the only 2-connected components of G are $H_1, \ldots, H_k$.

Remark that in Lemma 4.1, equalities (4.4) and (4.5) can be replaced by inequalities of type $\leq$ because of inequalities (4.3). Moreover, (4.5) is taken into account only for trivial 2-connected components as noticed in the proof of Lemma 4.1. Since the polytopes of the right-hand side of Observation 4.2 live in different spaces, by the theorem of Balas [1, 2], $B_{st}(G)$ is the projection into the x-space of the set of $x = (x^1, \dots, x^k) \in \mathbb{R}^E$ (where $x^i \in \mathbb{R}^{E_i}$ for $i = 1, \dots, k$) and $\lambda \in \mathbb{R}^k$ satisfying:

$$\begin{aligned}
&x^i_e \geq 0, && \text{for all } i = 1, \dots, k, \text{ for all } e \in E_i, && (4.11)\\
&x^i(Q) - x^i(C \setminus Q), \leq 0 && \text{for all } i = 1, \dots, k, \text{ for all } v_iv_{i+1}\text{-ears } Q \text{ and circuits } C \supseteq Q \text{ of } H_i, && (4.12)\\
&-x^i(P) \leq -\lambda^i, && \text{for all } i = 1, \dots, k, \text{ for all } v_iv_{i+1}\text{-paths } P \text{ of } H_i, && (4.13)\\
&x^i(P) \leq \lambda^i, && \text{for all } i = 1, \dots, k, \text{ for all } v_iv_{i+1}\text{-paths } P \text{ of } H_i \text{ contained in a circuit}, && (4.14)\\
&x^i_{v_iv_{i+1}} \leq \lambda^i, && \text{for all trivial } H_i, && (4.15)\\
&0 \leq \lambda^i, && \text{for all } i = 1, \dots, k, && (4.16)\\
&1 = \sum_{i=1}^{k} \lambda^i. && && (4.17)
\end{aligned}$$

The description of $B_{st}(G)$ in the natural space is obtained by projecting out λ from (4.11)–(4.17). Since (4.7)–(4.10) are valid for $B_{st}(G)$, we prove that the projected inequalities are either contained in or implied by (4.6)–(4.10), which implies our theorem.

We project out λ applying Fourier-Motzkin's elimination method [12]. Recall that, to get rid of λ^i, one has to combine every inequality where λ^i's coefficient is negative with every inequality where it is positive. Due to the structure of (4.11)–(4.17), it is enough to consider the following combinations.

Inequalities (4.11) and (4.12) have to be kept in the projection and are nothing but (4.7) and (4.8) respectively. The inequalities obtained by combination with (4.17) are of two forms. First, due to Observation 4.3, combining (4.17) and (4.13) for $i = 1, \dots, k$ yields precisely inequalites (4.9). Second, by Observation 4.3, the combination of (4.17) with, for every $i = 1, \dots, k$, an inequality of (4.14) if H_i is nontrivial and (4.15) otherwise, gives precisely (4.10). Note that, in the previous combination, replacing some inequalities of (4.14) or (4.15) by (4.16) provides redundant inequalities.

The last possible combinations are those involving inequalities containing only λ^i, for some $i \in \{1, \dots, k\}$. It gives $x^i \geq 0$, $x^i(P) \geq 0$ and $x^i(P) - x^i(P') \leq 0$ for all v_iv_{i+1}-paths P and P' of H_i such that P is contained in a circuit. These are redundant with respect to (4.7) and (4.9)–(4.10). □

4.3. Intersection property

The *dominant* of a polyhedron P of $\mathbb{R}^n$ is $P^+ = \{x \in \mathbb{R}^n : x \geq y \text{ for some } y \in P\}$. In this section, we show that intersecting the bond polytope and the st-cut dominant of a series parallel graph preserves integrality. Indeed, using Theorem 4.4 and the descriptions of the bond polytope [5] and the st-cut dominant [18], we prove that this intersection is nothing but the st-bond polytope.

We first state the result of [5] describing the bond polytope of a series-parallel graph, for which we need a few definitions. Two collections of edge subsets $\mathcal{F} = \{F_0, F_1, \dots, F_q\}$ and $\mathcal{P} = \{P_1, \dots, P_q\}$ form an *ear-cycle collection* if $\mathcal{F}$ is contained in an open nested ear decomposition $\mathcal{E}$ of G, $F_0 = C_{\mathcal{E}}$, and $F_i \Delta P_i$ is a cycle for $i = 1, \dots, q$.

Theorem 4.5 (Borne *et al.* [5]). *Let G be a series-parallel graph, $G_1, \dots, G_p$ its nontrivial 2-connected components and $\mathfrak{B}$ its set of bridges. The bond polytope of G is the set of $x \in \mathbb{R}^E$ satisfying the following* *inequalities.*

$$
\begin{aligned}
& x_e \geq 0, && \textit{for all } e \in E, && (4.18)\\
& x_e \leq x(C \setminus e), && \textit{for all circuits } C, \textit{ for all } e \in C, && (4.19)\\
& \sum_{i=1}^{p}(x(\mathcal{F}_i) - x(\mathcal{P}_i)) + 2x(\mathfrak{B}) \leq 2, && \begin{array}{l}\textit{for all } i = 1, \ldots, p,\\ \textit{for all ear-cycle collections } \mathcal{F}_i, \mathcal{P}_i \textit{ of } G_i.\end{array} && (4.20)
\end{aligned}
$$

The st-cut dominant $B_{st}(G)^+$ of G is described as follows, see *e.g.* [18]:

$$B_{st}(G)^+ = \{x \in \mathbb{R}_+^E : x(P) \geq 1 \text{ for all } st\text{-paths } P\}.$$

An st-bond being both an st-cut and a bond, we get the inclusion $B_{st}(G) \subseteq B_{st}(G)^+ \cap B(G)$. It turns out that the converse holds for series-parallel graphs, see below.

Corollary 4.6. *If G is series-parallel, then $B_{st}(G) = B_{st}(G)^+ \cap B(G)$.*

Proof. To prove $B_{st}(G)^+ \cap B(G) \subseteq B_{st}(G)$, by Theorem 4.4, we show that (4.6)–(4.10) are valid for $B_{st}(G)^+ \cap B(G)$. Note that $B_{st}(G)^+$ is the set of $x \in \mathbb{R}^E$ satisfying (4.7) and (4.9), as mentioned above. Recall that the 2-connected components $H_1, \ldots, H_\ell$ of G are numbered in such a way that, for $i = 1, \ldots, k$, H_i contains both v_i and v_{i+1}. Note that $G_1, \ldots, G_p$ of Theorem 4.5 correspond to the nontrivial 2-connected components among $H_1, \ldots, H_\ell$.

Let P be any st-path associated with (4.10). By definition of P and Observation 4.3, there exists an st-path P' such that the restriction C_i of $P \cup P'$ to H_i is a v_iv_{i+1}-circuit, for $i = 1, \ldots, k$ with H_i nontrivial. For $i = k+1, \ldots, \ell$ such that H_i is nontrivial, pick a circuit C_i of H_i. Since every circuit is contained in some open nested ear decomposition, $\{C_i\}, \emptyset$ is an ear-cycle collection of H_i for $i = 1, \ldots, \ell$ such that H_i is nontrivial. Let $\mathfrak{B}_H$ denote the edges of the trivial H_i's. Now, applying (4.9) for P and P', the definition of C_i and $\mathfrak{B}_H$, (4.7), and finally (4.20), yields

$$1 + 1 \leq x(P) + x(P') = \sum_{\substack{i=1\\ \text{nontrivial } H_i}}^{k} x(C_i) + 2x(\mathfrak{B}_H) \leq \sum_{i=1}^{p} x(C_i) + 2x(\mathfrak{B}) \leq 2. \tag{4.21}$$

Hence, there is equality everywhere and the validity of (4.10) follows. Moreover, since an edge e which belongs to no st-path is either in $\mathfrak{B} \setminus \mathfrak{B}_H$ or in some circuit of a nontrivial H_j with $j > k$, we also get (4.6).

Finally, we prove the validity of (4.8). Let $j \in \{1, \ldots, k\}$, Q be a v_jv_{j+1}-ear of H_j, and C be a circuit of H_j containing Q. By definition, there exists an open nested ear decomposition $\mathcal{E}$ of H_j containing Q such that $C_{\mathcal{E}}$ is a v_jv_{j+1}-circuit. By (4.21), we have

$$\sum_{i=1}^{p} x(C_i) + 2x(\mathfrak{B}) = 2 \tag{4.22}$$

for $C_j = C_{\mathcal{E}}$ and C_i, $i \neq j$, is a circuit defined as in (4.21).

Note that $\{C_{\mathcal{E}}, Q\}, \{C \setminus Q\}$ is an ear-cycle collection of H_j. Inequality (4.20) associated with $\mathcal{F}_j$, $\mathcal{P}_j = \{C_{\mathcal{E}}, Q\}, \{C \setminus Q\}$ and $\mathcal{F}_i, \mathcal{P}_i = \{C_i\}, \emptyset$ for $i \neq j$ is

$$\sum_{i=1}^{p} x(C_i) + x(Q) - x(C \setminus Q) + 2x(\mathfrak{B}) \leq 2. \tag{4.23}$$

Subtracting equation (4.22) to inequality (4.23) gives (4.8). □

Acknowledgements. We are indebted to the anonymous referees for their valuable comments which greatly helped to improve the presentation of the paper.

This research benefited from the support of the "FMJH Program Gaspard Monge in optimization and operation research" and from the support to this program from EDF. It also benefited from the University Paris 13 BQR research project *Connected and equicolored partitions of smartgrids*.

The problem of finding an *st*-bond arises in the context of partitioning a smart grid into areas having autonomy requirements. One of these requirements is to ensure the connectivity of each area for local energy transportation. Hence, partitioning a smart grid into two such areas reduces to finding an *st*-bond with additional properties.

References

[1] E. Balas, Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J. Algebr. Discrete Methods* **6** (1985) 466–486.

[2] E. Balas, Disjunctive programming: properties of the convex hull of feasible points. *Discrete Appl. Math.* **89** (1998) 1–44.

[3] F. Barahona and A.R. Mahjoub, On the cut polytope. *Math. Program.* **36** (1986) 157–73.

[4] M. Barbato, R. Grappe, M. Lacroix and C. Pira, Lexicographical polytopes. *Discrete Appl. Math.* **240** (2018) 3–7.

[5] S. Borne, P. Fouilhoux, R. Grappe, M. Lacroix and P. Pesneau, Circuit and bond polytopes on series-parallel graphs. *Discrete Optim.* **17** (2015) 55–68.

[6] G. Calvillo, The concavity and intersection properties for integral polyhedra, in Combinatorics 79. Part I. Vol 8 of *Ann. Discrete Math.* North-Holland, Amsterdam (1980) 221–228.

[7] A. Chakrabarti, L. Fleischer and C. Weibel, When the cut condition is enough: a complete characterization for multiflow problems in series-parallel networks, in *Proc. of the 44th Symposium on Theory of Computing STOC'12* (2012) 19–26.

[8] R.J. Duffin, Topology of series-parallel networks. *J. Math. Anal. Appl.* **10** (1965) 303–318.

[9] J. Edmonds, Submodular functions, matroids and certain polyhedra, in Combinatorial Structures and their Applications. (1970) 69–87.

[10] D. Eppstein, Parallel recognition of series-parallel graphs. *Inf. Comput.* **98** (1992) 41–55.

[11] L.R. Ford and D.R. Fulkerson, Maximal flow through a network. *Can. J. Math.* **8** (1956) 399–404.

[12] J.B.J. Fourier, Solution d'une question particuliere du calcul des inégalités. Nouveau Bulletin des Sciences par la Société philomatique de Paris (1826) 99–100.

[13] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979).

[14] M.R. Garey, D.S. Johnson and R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete. *SIAM J. Comput.* **5** (1976) 704–714.

[15] F.O. Hadlock, Finding a maximum cut of planar graph in polynomial time. *SIAM J. Comput.* **4** (1975) 221–225.

[16] K. Kuratowski, Sur le problème des courbes gauches en topologie. *Fundam. Math.* **15** (1930) 271–283.

[17] A. Schrijver, Combinatorial Optimization. Springer-Verlag, Berlin, Heidelberg (2003).

[18] M. Skutella and A. Weber, On the dominant of the s-t-cut polytope: vertices, facets, and adjacency. *Math. Program.* **124** (2010) 441–454.

Discrete Applied Mathematics 240 (2018) 3–7

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Lexicographical polytopes

Michele Barbato, Roland Grappe [*], Mathieu Lacroix, Clément Pira

Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, (UMR 7030), F-93430, Villetaneuse, France

A R T I C L E I N F O

Article history:
Received 30 March 2015
Received in revised form 4 April 2017
Accepted 19 April 2017
Available online 30 May 2017

Keywords:
Lexicographical polytopes
Polyhedral description
Superdecreasing knapsacks

A B S T R A C T

Within a fixed integer box of $\mathbb{R}^n$, lexicographical polytopes are the convex hulls of the integer points that are lexicographically between two given integer points. We provide their descriptions by means of linear inequalities.

Throughout, ℓ, u, r, s will denote integer points satisfying $\ell \leq r \leq u$ and $\ell \leq s \leq u$, that is r and s are within $[\ell, u]$. A point $x \in \mathbb{Z}^n$ is *lexicographically smaller than* $y \in \mathbb{Z}^n$, denoted by $x \preccurlyeq y$, if $x = y$ or the first nonzero coordinate of $y - x$ is positive. We write $x \prec y$ if $x \preccurlyeq y$ and $x \neq y$. The *lexicographical polytope* $P_{\ell,u}^{r \preccurlyeq s}$ is the convex hull of the integer points within $[\ell, u]$ that are lexicographically between r and s:

$$P_{\ell,u}^{r \preccurlyeq s} = conv\{x \in \mathbb{Z}^n : \ell \leq x \leq u, r \preccurlyeq x \preccurlyeq s\}.$$

The *top-lexicographical polytope* $P_{\ell,u}^{\preccurlyeq s} = conv\{x \in \mathbb{Z}^n : \ell \leq x \leq u, x \preccurlyeq s\}$ is the special case when $r = \ell$. Similarly, the *bottom-lexicographical polytope* is $P_{\ell,u}^{r \preccurlyeq} = conv\{x \in \mathbb{Z}^n : \ell \leq x \leq u, r \preccurlyeq x\}$.

Given $a, u \in \mathbb{R}_+^n$ and $b \in \mathbb{R}_+$, the *knapsack polytope* defined by $K_u^{a,b} = conv\{x \in \mathbb{Z}^n : \mathbf{0} \leq x \leq u, ax \leq b\}$ is *superdecreasing* if:

$$\sum_{i>k} a_i u_i \leq a_k \quad \text{for } k = 1, \dots, n. \tag{1}$$

Close relations between top-lexicographical and superdecreasing knapsack polytopes appear in the literature. For the 0/1 case, that is when $\ell = \mathbf{0}$ and $u = \mathbf{1}$, Gillmann and Kaibel [2] first noticed that top-lexicographical polytopes are special cases of superdecreasing knapsack ones, and the converse has been later established by Muldoon et al. [5]. Recently, Gupte [3] generalized the latter result by showing that all superdecreasing knapsacks are top-lexicographical polytopes.

To prove this last statement, Gupte [3] observes that a superdecreasing knapsack $K_u^{a,b}$ is the top-lexicographical polytope $P_{\mathbf{0},u}^{\preccurlyeq s}$, where s the lexicographically greatest integer point of $K_u^{a,b}$. The non trivial inclusion actually holds because every integer point x of $P_{\mathbf{0},u}^{\preccurlyeq s}$ satisfies $ax \leq as$. Indeed, by definition, if $x \prec s$, there exists $k \in \{1, \dots, n\}$ such that $x_k + 1 \leq s_k$ and $x_i = s_i$

[*] Corresponding author.
E-mail addresses: Michele.Barbato@lipn.univ-paris13.fr (M. Barbato), Roland.Grappe@lipn.univ-paris13.fr (R. Grappe), Mathieu.Lacroix@lipn.univ-paris13.fr (M. Lacroix), Clement.Pira@lipn.univ-paris13.fr (C. Pira).

http://dx.doi.org/10.1016/j.dam.2017.04.022
0166-218X/© 2017 Elsevier B.V. All rights reserved.

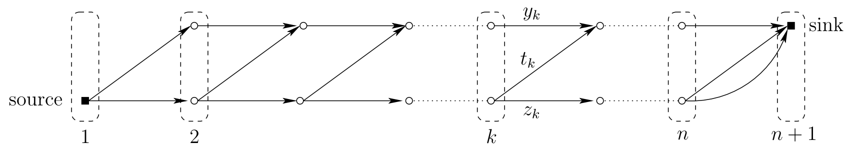

Fig. 1. Path representation of the points of $X^{\preccurlyeq s}_{\ell,u}$.

for $i < k$. Hence, we have $b - ax \geq as - ax \geq \sum_{i>k} a_i(s_i - x_i) + a_k \geq \sum_{i>k} a_i(s_i - x_i + u_i) \geq 0$, because of (1), $s_i \geq 0$ and $u_i \geq x_i$.

It turns out that top-lexicographical polytopes are superdecreasing knapsack polytopes. Indeed, let $P^{\preccurlyeq s}_{\ell,u}$ be a top-lexicographical polytope for some s within $[\ell, u]$. Possibly after translating, we may assume $\ell = \mathbf{0}$. Define a by $a_k = \sum_{i>k} a_i u_i + 1$, for $k = 1, \ldots, n$, and let $b = as$. Since the associated knapsack polytope $K^{a,b}_u$ is superdecreasing, if $x \preccurlyeq s$ then $ax \leq as = b$, for all x within $[\mathbf{0}, u]$. Moreover, the converse holds because, inequalities (1) being all strict, $s \prec x$ implies $b = as < ax$. Therefore, $P^{\preccurlyeq s}_{\mathbf{0},u} = K^{a,b}_u$. These observations are summarized in the following.

Observation 1. *Superdecreasing knapsacks are top-lexicographical polytopes, and conversely (up to translations).*

Motivated by a wide range of applications, such as knapsack cryptosystems [6] or binary expansion of bounded integer variables (*e.g.*, [8, p. 477]), several papers are devoted to the polyhedral description of these families of polytopes. For the 0/1 case, the description appeared in [4] from the knapsack point of view. It was later rediscovered from the lexicographical point of view in [2,5]. Moreover, Muldoon et al. [5] and Angulo et al. [1] independently showed that intersecting a 0/1 top- with a 0/1 bottom-lexicographical polytope yields the description of the corresponding lexicographical polytope. Recently, these results were generalized for the bounded case by Gupte [3].

In this paper, we provide the description of the lexicographical polytopes using extended formulations. Our approach provides alternative proofs of the aforementioned results of Gupte [3].

The outline of the paper is as follows. In Section 1, we provide a flow based extended formulation of the convex hull of the componentwise maximal points of a top-lexicographical polytope. Projecting this formulation is surprisingly straightforward, and thus we get the description in the original space. In Section 2, using the fact that a top-lexicographical polytope is, up to translation, the submissive of the above convex hull, we derive the description of top-lexicographical polytopes. We then show that a lexicographical polytope is the intersection of its top- and bottom-lexicographical polytopes.

1. Convex hull of componentwise maximal points

From now on, $X^{\preccurlyeq s}_{\ell,u}$ will denote the set of the points $p^i = (s_1, \ldots, s_{i-1}, s_i - 1, u_{i+1}, \ldots, u_n)$, for $i = 1, \ldots, n+1$ such that $s_i > \ell_i$, where $p^{n+1} = s$ by definition. Note that $X^{\preccurlyeq s}_{\ell,u}$ consists of the componentwise maximal integer points of $P^{\preccurlyeq s}_{\ell,u}$, to which we added, for later convenience, the point $p^n = (s_1, \ldots, s_{n-1}, s_n - 1)$ if $s_n > \ell_n$.

1.1. *A flow model for $X^{\preccurlyeq s}_{\ell,u}$*

We first model the points of $X^{\preccurlyeq s}_{\ell,u}$ as paths from 1 to $n+1$ in the digraph given in Fig. 1.

Our digraph is composed of $n+1$ layers, each containing two nodes except the first and the last ones. There are three arcs connecting the layer k to the layer $k+1$, an upper arc y_k, a diagonal arc t_k and a lower arc z_k. The only exception concerns the first level, which does not have the upper arc.

The arcs connecting two successive layers correspond to a coordinate of $x \in X^{\preccurlyeq s}_{\ell,u}$. More precisely, given a directed path P from 1 to $n+1$, we define the point x by setting, for $k = 1, \ldots, n$,

$$x_k = \begin{cases} u_k & \text{if } y_k \in P, \\ s_k - 1 & \text{if } t_k \in P, \\ s_k & \text{if } z_k \in P. \end{cases}$$

As shown in Observation 2, the set of (x, y, z, t) satisfying the following set of inequalities is an extended formulation of $conv(X^{\preccurlyeq s}_{\ell,u})$:

$$x_i = u_i y_i + (s_i - 1)t_i + s_i z_i \qquad \text{for } i = 1, \ldots, n, \tag{2}$$

$$y_1 = 0 \tag{3}$$

$$y_i = y_{i-1} + t_{i-1} \qquad \text{for } i = 2, \ldots, n, \tag{4}$$

$$z_i = z_{i+1} + t_{i+1} \qquad \text{for } i = 1, \ldots, n-1, \tag{5}$$

$$t_i = 0 \qquad \text{whenever } s_i = \ell_i, \tag{6}$$

$$y_n + t_n + z_n = 1 \tag{7}$$

$$y_i, t_i, z_i \geq 0 \qquad \text{for } i = 1, \ldots, n. \tag{8}$$

Observation 2. *$conv(X^{\preccurlyeq s}_{\ell,u}) = proj_x\{(x, y, z, t)$ satisfying (2)–(8)$\}$.*

Proof. First, note that there is a one-to-one correspondence between the points of $X^{\preccurlyeq s}_{\ell,u}$ and the paths from layer 1 to layer $n+1$ of the digraph. This implies that $X^{\preccurlyeq s}_{\ell,u}$ is the projection onto the x variables of the integer points of $Q = \{(x, y, z, t)$ satisfying (2)–(8)$\}$. The digraph being acyclic, the set of (y, z, t) satisfying (3)–(8) is a path polytope and thus is an integral polytope [7, Theorem 13.10]. The integrality of u and s implies that Q is integer, hence so is its projection onto the x variables, which concludes the proof. □

1.2. *Description of $conv(X^{\preccurlyeq s}_{\ell,u})$*

In the following result, we use Observation 2 to provide a linear description of $conv(X^{\preccurlyeq s}_{\ell,u})$.

Lemma 3. *$conv(X^{\preccurlyeq s}_{\ell,u})$ is described by the inequalities:*

$$\sum_{i=1, s_i>\ell_i}^{n} A_i(x) \geq -1 \tag{9}$$

$$A_k(x) \leq 0 \qquad \textit{for } k = 1, \dots, n, \tag{10}$$

$$A_k(x) \geq 0 \qquad \textit{when } s_k = \ell_k, \tag{11}$$

where, for $k = 1, \dots, n$,

$$A_k(x) := (x_k - s_k) + (u_k - s_k) \sum_{i=1, s_i>\ell_i}^{k-1} \left(\prod_{j=i+1, s_j>\ell_j}^{k-1} (u_j - s_j + 1) \right) (x_i - s_i).$$

Proof. By Observation 2, it suffices to project onto the x variables of the set of x, y, t, z satisfying (2)–(8).

For $k = 1, \dots, n$, we get $y_k = \sum_{i=1}^{k-1} t_i$ by (3) and (4). This, combined with (5) and (7), yields $z_k = 1 - \sum_{i=1}^{k} t_i$. Using those two equations in (2), and $t_k = 0$ whenever $s_k = \ell_k$, we obtain

$$t_k = s_k - x_k + (u_k - s_k) \sum_{i=1, s_i>\ell_i}^{k-1} t_i, \quad \text{for } k = 1, \dots, n. \tag{12}$$

We now show by induction on k that, for all $k = 1, \dots, n$,

$$\sum_{i=1, s_i>\ell_i}^{k} t_i = \sum_{i=1, s_i>\ell_i}^{k} (s_i - x_i) \prod_{j=i+1, s_j>\ell_j}^{k} (u_j - s_j + 1). \tag{13}$$

By definition of t_k, (13) holds for $k = 1$. Let us suppose that (13) holds for $k < n$ and show that it holds for $k + 1$. The result is immediate if $s_{k+1} = \ell_{k+1}$, hence assume that $s_{k+1} > \ell_{k+1}$. We have

$$\sum_{i=1, s_i>\ell_i}^{k+1} t_i = (s_{k+1} - x_{k+1}) + (u_{k+1} - s_{k+1}) \sum_{i=1, s_i>\ell_i}^{k} t_i + \sum_{i=1, s_i>\ell_i}^{k} t_i \tag{14}$$

$$= (s_{k+1} - x_{k+1}) + (u_{k+1} - s_{k+1} + 1) \sum_{i=1, s_i>\ell_i}^{k} (s_i - x_i) \prod_{j=i+1, s_j>\ell_j}^{k} (u_j - s_j + 1) \tag{15}$$

$$= \sum_{i=1, s_i>\ell_i}^{k+1} (s_i - x_i) \prod_{j=i+1, s_j>\ell_j}^{k+1} (u_j - s_j + 1).$$

Above, equality (14) follows from (12) applied to t_{k+1} and equality (15) follows using (13).

Injecting (13) in (12) yields

$$t_k = s_k - x_k + (u_k - s_k) \sum_{i=1, s_i>\ell_i}^{k-1} (s_i - x_i) \prod_{j=i+1, s_j>\ell_j}^{k-1} (u_j - s_j + 1) \quad \text{for } k = 1, \dots, n. \tag{16}$$

Up to now, we only used linear transformations, thus projecting out the variables y, z gives us (16), $\sum_{i=1, s_i>\ell_i}^{n} t_i \leq 1$, $t_k = 0$ whenever $s_k = \ell_k$ and $t_k \geq 0$ otherwise. Then, projecting onto the x variable gives the desired result. □

Note that the following derives from the above proof by combining (12) and the fact that, by (16), we have $t_k = -A_k$:

$$A_k(x) = (x_k - s_k) + (u_k - s_k) \sum_{i=1, s_i > \ell_i}^{k-1} A_i(x), \quad \text{for } k = 1, \dots, n. \tag{17}$$

2. Lexicographical polytopes

In this section, we first provide the description of top-lexicographical polytopes. We then show that a lexicographical polytope is the intersection of its top- and bottom-lexicographical polytopes.

2.1. Description of top-lexicographical polytopes

The following observation unveils the polyhedral relation between a top-lexicographical polytope and the convex hull of its componentwise maximal points.

Observation 4. $P_{\ell,u}^{\preccurlyeq s} = (conv(X_{\ell,u}^{\preccurlyeq s}) + \mathbb{R}_-^n) \cap \{x \geq \ell\}$.

Proof. Since $conv(X_{\ell,u}^{\preccurlyeq s})$ is integer and contained in $\{x \geq \ell\}$, the polyhedron on the right is integer. Seen the definitions, the observation follows. □

Remark that, when $\ell = \mathbf{0}$, $P_{\ell,u}^{\preccurlyeq s}$ is precisely the submissive of $conv(X_{\ell,u}^{\preccurlyeq s})$. Now, we derive from Lemma 3 and Observation 4 the linear description of top-lexicographical polytopes.

Theorem 5. $P_{\ell,u}^{\preccurlyeq s} = \{x \in \mathbb{R}^n : \ell \leq x \leq u, A_k(x) \leq 0,$ *for* $k = 1, \dots, n\}$.

Proof. Theorem 5 immediately follows from Observation 4 and the following description of $conv(X_{\ell,u}^{\preccurlyeq s}) + \mathbb{R}_-^n$,

$$conv(X_{\ell,u}^{\preccurlyeq s}) + \mathbb{R}_-^n = \{x \in \mathbb{R}^n : x \leq u \text{ and } A_k(x) \leq 0, \ \text{for } k = 1, \dots, n\}. \tag{18}$$

To prove (18), denote by Q its right hand side. By Lemma 3, the above inequalities are valid for $conv(X_{\ell,u}^{\preccurlyeq s})$. Since their coefficients for x are nonnegative, they also hold for $conv(X_{\ell,u}^{\preccurlyeq s}) + \mathbb{R}_-^n$. Note that the latter and Q have the same recession cone, thus it remains to show that the vertices of Q are vertices of $conv(X_{\ell,u}^{\preccurlyeq s})$. Let us prove it by induction on the dimension, the base case being immediate. We may assume that $u_n > s_n$, as otherwise $A_n(x) = x_n - s_n$ and the induction concludes. Let $\bar{x}$ be a vertex of Q.

Claim 6. $\sum_{i=1, s_i > \ell_i}^{n} A_i(\bar{x}) \geq -1$.

Proof. The indices i of $A_i(x)$ involved in sums throughout this proof satisfy $s_i > \ell_i$, yet to ease the reading, we will omit the subscripts "$s_i > \ell_i$". By contradiction, assume that $\sum_{i=1}^{n} A_i(\bar{x}) < -1$. Since $\bar{x}$ is a vertex, and x_n appears only in $x_n \leq u_n$ and $A_n(x) \leq 0$, at least one of them holds with equality. If the latter does, then by (17) and $u_n > s_n$, we get the contradiction $0 = A_n(\bar{x}) \leq (u_n - s_n)(1 + A_1(\bar{x}) + \cdots A_{n-1}(\bar{x})) < (u_n - s_n)(1 - 1) = 0$. Therefore $A_n(\bar{x}) < 0$ and $\bar{x}_n = u_n$. For $x \in \mathbb{R}^n$, we denote $x' := (x_1, \dots, x_{n-1})$. Necessarily, $\bar{x}'$ satisfies to equality $n-1$ linearly independent of the remaining inequalities, and hence $\bar{x}'$ is a vertex of $\{x \in \mathbb{R}^{n-1} : x_k \leq u_k, A_k(x) \leq 0, \ \text{for } k = 1, \dots, n-1\}$. By the induction hypothesis, $\bar{x}'$ is a vertex of $conv(X_{\ell',u'}^{\preccurlyeq s'}) + \mathbb{R}_-^{n-1}$, hence $\sum_{i=1}^{n-1} A_i(\bar{x}') \geq -1$. But now $A_n(\bar{x}) < 0$, $\bar{x}_n = u_n$ and (17) imply $A_1(\bar{x}') + \cdots + A_{n-1}(\bar{x}') < -1$, a contradiction. ■

Let us show that $A_k(\bar{x}) = 0$ whenever $s_k = \ell_k$. Indeed, in this case, $\bar{x}_k$ only appears in $A_k(\bar{x}) \leq 0$ and $\bar{x}_k \leq u_k$, and one is satisfied with equality since $\bar{x}$ is a vertex. If $\bar{x}_k = u_k$, then by (17), Claim 6 and $A_i(\bar{x}) \leq 0$, for $i = 1 \dots, n$, we get $0 \geq A_k(\bar{x}) = (u_k - s_k)(1 + \sum_{i=1, s_i > \ell_i}^{k-1} A_i(\bar{x})) \geq 0$. Consequently, $\bar{x}$ belongs to $conv(X_{\ell,u}^{\preccurlyeq s})$ and this proves (18). □

Symmetrically, bottom-lexicographical polytopes are described as follows.

Corollary 7. $P_{\ell,u}^{r \preccurlyeq} = \{x \in \mathbb{R}^n : \ell \leq x \leq u, B_k(x) \leq 0,$ *for* $k = 1, \dots, n\}$, *where, for* $k = 1, \dots, n$,

$$B_k(x) = (r_k - x_k) + (r_k - \ell_k) \sum_{i=1, r_i < u_i}^{k-1} \left(\prod_{j=i+1, r_j < u_j}^{k-1} (r_j - \ell_j + 1) \right) (r_i - x_i).$$

2.2. Lexicographical polytopes

By definition, we have $P^{r\preccurlyeq s}_{\ell,u} \subseteq P^{r\preccurlyeq}_{\ell,u} \cap P^{\preccurlyeq s}_{\ell,u}$. It turns out that the converse holds, see Theorem 8. In particular, $P^{r\preccurlyeq}_{\ell,u} \cap P^{\preccurlyeq s}_{\ell,u}$ is an integer polytope.

Theorem 8. *A lexicographical polytope is the intersection of its top- and bottom-lexicographical polytopes.*

Proof. It remains to prove that $P^{r\preccurlyeq s}_{\ell,u} \supseteq Q$, where $Q = P^{r\preccurlyeq}_{\ell,u} \cap P^{\preccurlyeq s}_{\ell,u}$. Let us prove it by induction on the dimension, the one-dimensional case being immediate.

If $r_1 = s_1$, then the problem reduces to the $(n-1)$-dimensional case, and using induction concludes.

If $r_1 + 1 \le \pi \le s_1 - 1$ for some integer π, then let ℓ' be obtained from ℓ by replacing ℓ_1 by π. By $s_1 > \ell'_1$ and the definition of $A_k(x)$, applying Theorem 5 gives $P^{\preccurlyeq s}_{\ell,u} \cap \{x_1 \ge \pi\} = P^{\preccurlyeq s}_{\ell',u}$. Moreover, since $\pi > r_1$, the latter is contained in $P^{r\preccurlyeq}_{\ell,u}$. Therefore $Q \cap \{x_1 \ge \pi\} = P^{\preccurlyeq s}_{\ell',u}$ is integer. Similarly, $Q \cap \{x_1 \le \pi\}$ is integer, hence so is Q, and we are done.

The remaining case is when $r_1 = s_1 - 1$. Let $\bar{x} \in P^{r\preccurlyeq}_{\ell,u} \cap P^{\preccurlyeq s}_{\ell,u}$. If $\bar{x}_1 = s_1$, when $\bar{x}$ is written as a convex combination of integer points of $P^{\preccurlyeq s}_{\ell,u}$, all of them have their first coordinate equal to s_1, and hence belong to $P^{r\preccurlyeq s}_{\ell,u}$. By convexity, so does $\bar{x}$ and we are done. A similar argument may be applied if $\bar{x}_1 = r_1$. Therefore, we may assume that $r_1 < \bar{x}_1 < s_1$.

Let $\lambda = \bar{x}_1 - r_1$, and define y by $y_1 = s_1$ and $y_k = u_k + \frac{\bar{x}_k - u_k}{\lambda}$ for $k = 2, \dots, n$. Similarly, define z by $z_1 = r_1$ and $z_i = \ell_i + \frac{\bar{x}_i - \ell_i}{1-\lambda}$, for $i = 2, \dots, n$. The following claim finishes the proof, where, given two points v and w of $\mathbb{R}^n$, $\max(v, w)$ (resp. $\min(v, w)$) will denote the point of $\mathbb{R}^n$ whose ith coordinate is $\max\{v_i, w_i\}$ (resp. $\min\{v_i, w_i\}$) for $i = 1, \dots, n$.

Claim 9. *$\bar{x}$ is a convex combination of $\bar{y} = \max(y, \ell)$ and $\bar{z} = \min(z, u)$ which both belong to $P^{r\preccurlyeq s}_{\ell,u}$.*

Proof. First, let us show that $y \in conv(X^{\preccurlyeq s}_{\ell,u}) + \mathbb{R}^n_-$. As $\bar{x} \le u$, we have $y \le u$. Moreover, $A_1(y) = y_1 - s_1 = 0$. Now, we prove by induction that $A_k(y) = \frac{1}{\lambda} A_k(\bar{x})$ for $k = 2, \dots, n$. Using (17), $A_1(y) = 0$, the definition of y_k, and the induction hypothesis, we have $A_k(y) = \frac{1}{\lambda}[\bar{x}_k - s_k + (\lambda - 1)(u_k - s_k) + (u_k - s_k)\sum_{i=2, s_i > \ell_i}^{k-1} A_i(\bar{x})]$. Since $\lambda - 1 = \bar{x}_1 - s_1 = A_1(\bar{x})$ and $s_1 = r_1 + 1 > \ell_1$, we get by (17) that $A_k(y) = \frac{1}{\lambda} A_k(\bar{x})$, for $k = 2, \dots, n$. Since $A_k(\bar{x}) \le 0$, we have $A_k(y) \le 0$. Hence, $y \in conv(X^{\preccurlyeq s}_{\ell,u}) + \mathbb{R}^n_-$. Therefore, there exists y^+ of $conv(X^{\preccurlyeq s}_{\ell,u})$ with $y^+ \ge y$. Clearly, $y^+ \ge \ell$ hence $y^+ \ge \max(y, \ell)$. Thus, $\max(y, \ell)$ belongs to $conv(X^{\preccurlyeq s}_{\ell,u}) + \mathbb{R}^n_-$ and, by Observation 4, to $P^{\preccurlyeq s}_{\ell,u}$. Moreover, as its first coordinate equals s_1, $\max(y, \ell)$ belongs to $P^{r\preccurlyeq s}_{\ell,u}$. Similarly, $\min(z, u)$ also belongs to $P^{r\preccurlyeq s}_{\ell,u}$.

Finally, we have $(1-\lambda)\bar{z}_1 + \lambda\bar{y}_1 = (1-\lambda)(s_1 - 1) + \lambda s_1 = s_1 - 1 + \lambda = \bar{x}_1$. For $i \in \{2, \dots, n\}$, we have $(1-\lambda)\bar{z}_i + \lambda\bar{y}_i = \min(\bar{x}_i - \lambda\ell_i, (1-\lambda)u_i) + \max((\lambda-1)u_i + \bar{x}_i, \lambda\ell_i) = \bar{x}_i - \max(\lambda\ell_i, (\lambda-1)u_i + \bar{x}_i) + \max((\lambda-1)u_i + \bar{x}_i, \lambda\ell_i) = \bar{x}_i$. Therefore, $\bar{x} = (1-\lambda)\bar{z} + \lambda\bar{y}$ and we are done. ■ □

Note that the above result implies that the family of lexicographical polytopes defined on a fixed box $[\ell, u]$ is closed by intersection. Beside, combined with Theorem 5 and Corollary 7, it provides the description of lexicographical polytopes.

Corollary 10. *The lexicographical polytope $P^{r\preccurlyeq s}_{\ell,u}$ is described as follows:*

$$P^{r\preccurlyeq s}_{\ell,u} = \left\{ \begin{array}{lll} x \in \mathbb{R}^n : & A_k(x) \le 0 & \text{for } k = 1, \dots, n \\ & B_k(x) \le 0 & \text{for } k = 1, \dots, n \\ & \ell \le x \le u & \end{array} \right\}.$$

References

[1] G. Angulo, S. Ahmed, S.S. Dey, V. Kaibel, Forbidden vertices, Math. Oper. Res. 40 (2) (2015) 350–360.
[2] R. Gillmann, V. Kaibel, Revlex-initial 0/1-polytopes, J. Combin. Theory Ser. A 113 (5) (2006) 799–821.
[3] A. Gupte, Convex hulls of superincreasing knapsacks and lexicographic orderings, Discrete Appl. Math. 201 (2016) 150–163.
[4] M. Laurent, A. Sassano, A characterization of knapsacks with the max-flow–min-cut property, Oper. Res. Lett. 11 (2) (1992) 105–110.
[5] F.M. Muldoon, W.P. Adams, H.D. Sherali, Ideal representations of lexicographic orderings and base-2 expansion of integer variables, Oper. Res. Lett. 41 (2013) 32–39.
[6] A.M. Odlyzko, The rise and fall of knapsack cryptosystems, in: Cryptology and Computational Number Theory, A.M.S, 1990, pp. 75–88.
[7] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer-Verlag, Berlin, Heidelberg, New York, 2003.
[8] F. Vanderbeck, L.A. Wolsey, Reformulation and decomposition of integer programs, in: Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, Laurence A. Wolsey (Eds.), 50 Years of Integer Programming 1958-2008, Springer, Berlin, Heidelberg, 2010, pp. 431–502.

Discrete Optimization 50 (2023) 100810

Contents lists available at ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Hard problems on box-totally dual integral polyhedra

Patrick Chervet [a], Roland Grappe [b,c], Mathieu Lacroix [b], Francesco Pisanu [b,*], Roberto Wolfler Calvo [b]

[a] Lycée Olympe de Gouges, rue de Montreuil à Claye, 93130, Noisy le Sec, France
[b] Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430, Villetaneuse, France
[c] LAMSADE (CNRS UMR7243), University Paris-Dauphine, PSL Research University, France

ARTICLE INFO

Keywords:
Box-TDI polyhedron
Totally equimodular matrix
Incidence matrix

ABSTRACT

In this paper, we study the complexity of some fundamental questions regarding box-totally dual integral (box-TDI) polyhedra. First, although box-TDI polyhedra have strong integrality properties, we prove that Integer Programming over box-TDI polyhedra is NP-complete, that is, finding an integer point optimizing a linear function over a box-TDI polyhedron is hard. Second, we complement the result of Ding et al. (2008) who proved that deciding whether a given system is box-TDI is co-NP-complete: we prove that recognizing whether a polyhedron is box-TDI is co-NP-complete.

To derive these complexity results, we exhibit new classes of totally equimodular matrices – a generalization of totally unimodular matrices – by characterizing the total equimodularity of incidence matrices of graphs.

1. Introduction

Totally dual integral systems were introduced in the late 70's and serve as a general framework for establishing various min–max relations in combinatorial optimization [1]. A rational system of linear inequalities $Ax \le b$ is *totally dual integral (TDI)* if the minimization problem in the linear programming duality relation:

$$\max\{c^\top x : Ax \le b\} = \min\{b^\top y : A^\top y = c, y \ge \mathbf{0}\}$$

admits an integer optimal solution for each integer vector c such that the maximum is finite. As is well-known, such systems $Ax \le b$ can be used to define every integer polyhedron, with b integral [2].

A stronger property is the box-total dual integrality, where a system $Ax \le b$ is *box-totally dual integral (box-TDI)* if $Ax \le b$, $\ell \le x \le u$ is TDI for all rational vectors ℓ and u (with possible infinite components). General properties of such systems can be found in Cook [3] and Section 22.4 of Schrijver [1].

Box-TDI systems are intimately related to totally unimodular matrices. A matrix is *totally unimodular (TU)* if every subset of linearly independent rows forms a unimodular matrix, a matrix being *unimodular* if it has full row rank and all its nonzero maximal minors have value ± 1. A matrix A is TU if and only if the system $Ax \le b$ is box-TDI for each rational vector b [1, Page 318].

Until recently, the vast majority of known box-TDI systems were systems associated with TU matrices. For instance, König's Theorem [4] can be seen as a consequence of the fact that the vertex-edge incidence matrix of a graph is TU if and only if the graph is bipartite [5].

[*] Corresponding author.
E-mail addresses: grappe@lipn.fr (R. Grappe), lacroix@lipn.fr (M. Lacroix), pisanu@lipn.fr (F. Pisanu), wolfler@lipn.fr (R. Wolfler Calvo).

https://doi.org/10.1016/j.disopt.2023.100810
Received 16 March 2023; Received in revised form 5 September 2023; Accepted 2 November 2023
Available online 8 November 2023
1572-5286/© 2023 Elsevier B.V. All rights reserved.

In the last two decades, several new box-TDI systems were exhibited. Chen, Ding, and Zang [6] characterized box-Mengerian matroid ports. In [7], they provided a box-TDI system describing the 2-edge-connected spanning subgraph polyhedron for series–parallel graphs. Ding, Tan, and Zang [8] characterized the graphs for which the Edmonds system for defining the matching polytope [9], which is always TDI as shown by Cunningam and Marsh [10], is box-TDI. Ding, Zang, and Zhao [11] introduced new subclasses of box-perfect graphs. Cornaz, Grappe, and Lacroix [12] provided several box-TDI systems in series–parallel graphs. More recently, these graphs have also been characterized by the box-TDIness of their flow cone [13] and that of their k-edge-connected polyhedron [14]. These last two results use characterizations of box-TDI polyhedra given by Chervet, Grappe, and Robert [15].

As stated before, every integer polyhedron can be defined by a TDI system. Yet, the statement no longer holds if we replace TDI by box-TDI. A polyhedron that can be described by a box-TDI system is a *box-TDI polyhedron*, and every TDI system describing it is actually box-TDI [3]. Box-TDI polyhedra characterize the following generalization of TU matrices. A matrix is *totally equimodular (TE)* if every subset of linearly independent rows forms an equimodular matrix, a matrix being *equimodular* if it has full row rank and all its nonzero maximal minors have the same absolute value. A matrix A is TE if and only if the polyhedron $\{x : Ax \leq b\}$ is box-TDI for each rational vector b [15].

Several complexity results relative to TDIness and box-TDIness are known. Deciding whether a system $Ax \leq b$ is TDI or whether it is box-TDI are two co-NP-complete problems [6]. The first problem remains co-NP-complete even for conic systems [16], that is, when $b = \mathbf{0}$. A tractable case for the recognition of box-TDI systems is when A is TU, since total unimodularity can be tested in polynomial time [17]. We continue along this line by providing two new hardness results.

Contributions. In this paper, we prove that the problem of deciding whether a given polyhedron is box-TDI is co-NP-complete. Our proof builds upon the hardness result of Ding et al. [6] about the recognition of box-TDI systems.

We also prove that the edge-vertex incidence matrix of any graph is TE. This implies that the edge relaxation of the stable set problem is a box-TDI polyhedron. From the NP-hardness of the maximum stable set problem, it follows that optimizing a linear function over $\{x \in \mathbb{Z}^n : Ax \leq \mathbf{1}\}$ is NP-hard when A is TE. Since the latter problem is polynomial when A is TU, this unveils a major difference between TE and TU matrices. Moreover, this hardness result also implies that integer optimization over box-TDI polyhedra is NP-hard.

Another difference between TE and TU matrices is that the transpose of a TE matrix is not always TE. We highlight this fact by characterizing the equimodularity and the total equimodularity of the vertex-edge incidence matrix of a graph.

Outline. In Section 2, we provide the definitions and some results needed throughout the paper. In Section 3, we characterize the equimodularity and the total equimodularity of the edge-vertex and of the vertex-edge incidence matrix of a graph. Based on these results, in Section 4, we characterize the box-TDIness of the stable set polytope and that of the edge cover dominant polyhedron of a graph. As a consequence, we prove that Integer Programming over box-TDI polyhedra is NP-complete and that recognizing whether a polyhedron is box-TDI is co-NP-complete.

2. Preliminaries

2.1. Matrices and polyhedra

In a given matrix, a *minor* is the determinant of any square submatrix. When the latter has maximal size, the associated minor is *maximal*.

Recall that an integer matrix is unimodular if it has full row rank and all its nonzero maximal minors are ± 1. More generally, a rational matrix is equimodular if it has full row rank and all its nonzero maximal minors have the same absolute value. As observed in [15], checking equimodularity can be done in polynomial time. Indeed, equimodular matrices are TU up to a basis change, and checking total unimodularity can be done in polynomial time [17].

A *face* of a polyhedron $P = \{x : Ax \leq b\}$ is the polyhedron obtained by imposing equality on some inequalities in the description of P. A matrix M is *face-defining* for a face F of P if it has full row rank and the affine space generated by F can be written as $\{x : Mx = d\}$ for some vector d of appropriate size. These matrices characterize box-TDI polyhedra as follows.

Theorem 1 (*Chervet et al. [15]*). *A polyhedron P is box-TDI if and only if every face-defining matrix of P is equimodular.*

In our proofs, we will use Theorem 1 combined with the following observation.

Observation 2 (*See, for Instance, Chervet et al. [15]*). *A full row rank matrix M is face-defining for a face F of a polyhedron $P \subseteq \mathbb{R}^n$ if and only if there exist a vector d and a set $H \subseteq F \cap \{x : Mx = d\}$ of $\dim(F)+1$ affinely independent points such that $|H|+rank(M) = n+1$.*

Recall that a matrix is TE if every subset of linearly independent rows forms an equimodular matrix. By Theorem 1, every polyhedron whose constraint matrix is TE is box-TDI. It turns out that this characterizes TE matrices.

Theorem 3 (*Chervet et al. [15]*). *A matrix A of $\mathbb{Q}^{m\times n}$ is totally equimodular if and only if the polyhedron $\{x : Ax \leq b\}$ is box-TDI for all $b \in \mathbb{Q}^m$.*

TE matrices are to box-TDI polyhedra what TU matrices are to box-TDI systems.

Theorem 4 (*Hoffman et al. [5]*). *A matrix A of $\mathbb{Z}^{m\times n}$ is totally unimodular if and only if the system $Ax \leq b$ is box-TDI for all $b \in \mathbb{Z}^m$.*

2.2. Matrices and graphs

In this paper, all graphs are undirected. Without loss of generality, we assume that they are simple, connected, and have at least one edge, as our results extend immediately to general undirected loopless graphs.

Let $G = (V, E)$ be a graph. Given $W \subseteq V$, let $\delta(W)$ (respectively $E(W)$) be the set of edges with exactly one extremity (respectively both extremities) in W. An edge uv is said to *cover* u and v. Given $F \subseteq E$, $V(F)$ is the union of the vertices covered by each edge of F. A graph $G' = (V', E')$ is a *subgraph of* G if $E' \subseteq E$ and $V' = V(E')$. A subgraph $G' \subseteq G$ is a *spanning subgraph* of G if $V' = V$. The *degree* of a vertex v of G is the number of edges of G covering v and is denoted by $d_G(v)$. A set of edges $C \subseteq E$ is a *circuit* if the subgraph $(V(C), C)$ is connected and all its vertices have degree 2. A *hole* is a circuit for which $E(V(C)) = C$.[1] An *odd circuit* is a circuit with an odd number of edges, similarly, an *odd hole* is a hole with an odd number of edges. A graph is *bipartite* if it does not contain any odd circuit. A *perfect matching* of a graph is a set of pairwise nonadjacent edges covering all the vertices.

Let A_G denote the *edge-vertex incidence matrix* of G, that is the matrix whose rows are the characteristic vectors of the edges of G, where the *characteristic vector of an edge* $e = uv$ is the vector $\chi^e \in \{0,1\}^V$ with $\chi^e_w = 1$ if $w \in \{u, v\}$ and $\chi^u_w = 0$ otherwise. Similarly, $A_G^\top$ is the *vertex-edge incidence matrix*. When a result applies to both the edge-vertex and the vertex-edge incidence matrices, we simply write incidence matrix. For $F \subseteq E$, let A_F be the edge-vertex incidence matrix of the graph $(V(F), F)$. The *characteristic vector of a vertex* u is the vector $\chi^u \in \{0,1\}^V$ with $\chi^u_w = 1$ if $w = u$ and $\chi^u_w = 0$ otherwise.

Odd circuits are involved in the value of the determinants of incidence matrices.

Theorem 5 (*Grossman et al. [18]*). *For a connected graph G with n vertices and n edges, $|\det(A_G)|$ is equal to 0 if G is bipartite, and 2 otherwise.*

Theorem 5 comes from the fact that since G is connected, it has exactly one circuit, and then the value of the determinant of its incidence matrix depends on the parity of that circuit. Theorem 5 can be used to deduce a well-known result characterizing bipartite graphs, generally referred to as Hoffman and Kruskal's Theorem [5].

Theorem 6 (*Hoffman et al. [5]*). *The incidence matrix of a graph is totally unimodular if and only if the graph is bipartite.*

In our proofs, we will use the following lemma to show that a matrix is not equimodular.

Lemma 7. *For an odd circuit C, and for every $u \in V(C)$, the matrix $\left[A_C^\top, \chi^u\right]$ has full row rank but is not equimodular.*

Proof. Reordering the rows and the columns of $\left[A_C^\top, \chi^u\right]$, we may assume that the matrix is as follows.

$$\begin{bmatrix} \underbrace{\begin{matrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \\ 1 & & & & 1 \end{matrix}}_{A_C^\top} & \underbrace{\begin{matrix} \\ \\ \\ \\ 1 \end{matrix}}_{\chi^u} \end{bmatrix}$$

Since C is an odd circuit, $|\det(A_C^\top)| = 2$, hence $\left[A^\top, \chi^u\right]$ has full row rank. Moreover, the last $|C|$ columns form a lower triangular matrix with 1s on the main diagonal, thus they have determinant 1. Therefore, the matrix is not equimodular. □

The definition of bipartite graphs can be generalized as follows. A graph G is *quasi-bipartite* if for each odd circuit C of G, the graph $G \setminus V(C)$ has at least one isolated vertex. These graphs characterize the box-TDIness of the system given in the following theorem, where K_4 denotes the complete graph with 4 vertices.

Theorem 8 (*Ding et al. [6]*). *Given a connected graph G, the system $A_G^\top x \geq \mathbf{1}, x \geq \mathbf{0}$ is box-TDI if and only if G is a quasi-bipartite graph different from K_4.*

3. Incidence matrices and total equimodularity

In this section, we characterize when the incidence matrix of a graph is TE. Since total equimodularity is not preserved under taking the transpose, this section is divided into two parts: edge-vertex incidence matrices and vertex-edge incidence matrices.

[1] In this paper, triangles are considered holes.

3.1. Edge-vertex incidence matrices

Recall that the edge-vertex incidence matrix of a graph is TU if and only if the graph is bipartite. This extends to all graphs as follows in the more general context of total equimodularity.

Theorem 9. *The edge-vertex incidence matrix of a graph is totally equimodular.*

Proof. Let $G = (V, E)$ be a graph and let M be a full row rank matrix formed by a subset of k rows of the edge-vertex incidence matrix of G. Let us prove that M is equimodular by induction on its number of rows: the base case is when M has one row, and then M is equimodular since a row has only values in $\{0, 1\}$. The matrix M encodes a subgraph $H = (V, F)$ of G with $k = |F|$ edges.

We have $|V(F)| \geq |F|$, as otherwise M would have too many columns of zeros to have full row rank. If $|V(F)| = |F|$, then M has exactly one $k \times k$ submatrix which is nonsingular, hence M is equimodular. If $|V(F)| > |F|$, then H has a vertex u of degree one. Indeed, if every vertex of $V(F)$ had degree at least two we would have $2|F| = \sum_{w \in V(F)} d_H(w) \geq 2|V(F)|$, a contradiction.

The column of u in M contains a single one, in uv's row, where v is the neighbor of u in H. Let M' be the matrix obtained from M by removing uv's row. Note that M' has full row rank since M has it. A nonsingular $k \times k$ submatrix N of M has to contain at least one of u and v, as otherwise it has only zeros in uv's row. When N contains exactly one of them, then, expand along uv's row using the cofactor expansion. When N contains both of them, then develop, as before, with respect to u's column. In both cases, the determinant of N is equal to a maximal minor of M', up to the sign. By the induction hypothesis, M' is equimodular, so all these determinants are equal in absolute value. Therefore, so are the nonzero $k \times k$ determinants of M, and M is equimodular. □

In [18], the authors proved that the problem of determining the maximum absolute value of a minor of a given incidence matrix is NP-hard. Hence, Theorem 9 implies the following.

Corollary 10. *Determining the maximum absolute value of a minor of a totally equimodular matrix is NP-hard.*

3.2. Vertex-edge incidence matrices

In contrast to edge-vertex incidence matrices, vertex-edge incidence matrices of graphs are rarely TE. We characterize below the classes of graphs for which they are. We also characterize when these matrices are equimodular. Note that when the graph G is bipartite the incidence matrix of G does not have full row rank by Theorem 5. Otherwise, the determinant of a square incidence matrix is 2^k, where $k \geq 1$ is the number of vertex-disjoint odd circuits [18]. Therefore, to get an equimodular vertex-edge incidence matrix, one should forbid vertex-disjoint odd circuits. It turns out that it is an equivalence, as proved below.

Theorem 11. *The vertex-edge incidence matrix of a connected nonbipartite graph $G = (V, E)$ is equimodular if and only if G has no pair of vertex-disjoint odd circuits.*

Proof. Note that every maximal square submatrix of a vertex-edge incidence matrix induces a spanning subgraph of G having $|V|$ edges. Since a spanning tree of G has $|V| - 1$ edges, this subgraph contains a circuit.
($\Rightarrow$) Suppose that G has two vertex-disjoint odd circuits C_1 and C_2, and let e_1 and e_2 be edges of C_1 and C_2, respectively. Since G is connected, there exists a spanning tree T of G containing $C_1 \cup C_2 \setminus \{e_1, e_2\}$. Since C_1 and C_2 are vertex-disjoint, there exists an edge e of T whose removal splits T into two trees T_1 and T_2 with $C_1 \setminus \{e_1\} \subseteq T_1$ and $C_2 \setminus \{e_2\} \subseteq T_2$.

By Theorem 5, $|\det(A^{\top}_{T_i \cup \{e_i\}})| = |\det(A^{\top}_{T \cup \{e_i\}})| = 2$, for $i = 1, 2$. By construction, $|\det(A^{\top}_{T \cup \{e_1, e_2\} \setminus \{e\}})| = |\det(A^{\top}_{T_1 \cup \{e_1\}}) \det(A^{\top}_{T_2 \cup \{e_2\}})| = 4$. The determinants of the maximal nonsingular square submatrices $A^{\top}_{T \cup \{e_1, e_2\} \setminus \{e\}}$ and $A^{\top}_{T \cup \{e_1\}}$ of $A^{\top}_G$ differ in absolute value, thus $A^{\top}_G$ is not equimodular.
($\Leftarrow$) Suppose that G is not bipartite and has no two vertex-disjoint odd circuits. Note that since G is connected, it contains a nonbipartite connected spanning subgraph H with $|V|$ edges. By Theorem 5, we have $|\det(A^{\top}_H)| = 2$ and $A^{\top}_G$ has full row rank. This holds for every nonbipartite connected spanning subgraph with $|V|$ edges. The other spanning subgraphs of G with $|V|$ edges are either connected and bipartite or a product of smaller minors corresponding to connected subgraphs. In the first case, the associated minor is zero by Theorem 5. In the second case, by Theorem 5 and the fact that G has no two vertex-disjoint odd circuits, one of these smaller minors is zero. Therefore, every maximal minor of $A^{\top}_G$ belongs to $\{-2, 0, 2\}$, and $A^{\top}_G$ is equimodular. □

Theorem 11 gives a characterization of graphs having two vertex-disjoint odd circuits in terms of total equimodularity. A graph-theoretic characterization of these graphs was given by Lovász — see [19, page 546], or [20] for a proof without matroid decomposition. They also appear in the context of extended formulations [21] and unimodular covers [22]. In particular, since equimodularity can be tested in polynomial time [15, Section 4.1], Theorem 11 provides another polynomial-time algorithm for their recognition [23].

Theorem 12. *The vertex-edge incidence matrix of a connected graph $G = (V, E)$ is totally equimodular if and only if G is an odd hole or a bipartite graph.*

Proof. ($\Rightarrow$) Suppose that G is neither bipartite nor an odd hole. Then, G contains an odd hole C and two edges uv and uw in C and $\delta(V(C))$, respectively.

The submatrix of $A_G^\top$ restricted to the rows associated with $V(C)$ can be reordered such that the first $|C|+1$ columns form the matrix $\left[A_C, \chi^u\right]$. By Lemma 7, it has full row rank but is not equimodular. This implies that $A_G^\top$ is not TE.
($\Leftarrow$) If G is bipartite, then $A_G^\top$ is TU by Theorem 6, and hence TE. Now, if G is an odd hole, then $A_G^\top$ is also the edge-vertex incidence matrix of an odd hole, and hence is TE by Theorem 9. $\square$

By Theorem 12, deciding whether a vertex-edge incidence matrix is TE can be done in polynomial time. This might be a first step towards the complexity of recognizing TE matrices, which is an open problem raised in [15].

4. Box-TDIness and complexity consequences

In this section, we provide several complexity results based on the characterization of total equimodularity of incidence matrices devised in the previous section.

4.1. Edge relaxation of the stable set polytope

Given a graph $G=(V,E)$, a *stable set* is a set of pairwise nonadjacent vertices. The polytope $\{x \in \mathbb{R}^V : A_G x \leq \mathbf{1}, x \geq \mathbf{0}\}$ is called the *edge relaxation of the stable set polytope* of G and its integer points are precisely the characteristic vectors of the stable sets of G.

By Theorems 3 and 9, every polyhedron of the form $\{x \in \mathbb{R}^V : A_G x \leq b\}$ with b rational is box-TDI. As adding $x \geq \mathbf{0}$ preserves box-TDIness, we have the following.

Corollary 13. *The edge relaxation of the stable set polytope is box-TDI.*

Since finding a maximum stable set in a given graph is NP-hard [24], Corollary 13 implies that integer programming over a box-TDI polyhedron is NP-hard.

Corollary 14. *Given a box-TDI polyhedron P and a cost vector c, finding an integer point x maximizing $c^\top x$ over P is NP-hard.*

4.2. Edge relaxation of the edge cover dominant

Since multiplying a row by -1 preserves total equimodularity, by Theorems 3 and 12, when G is an odd hole or a bipartite graph, the polyhedron $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}\}$ is box-TDI. It turns out that the converse holds.

Theorem 15. *Given a connected graph $G=(V,E)$, the polyhedron $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}\}$ is box-TDI if and only if G is an odd hole or a bipartite graph.*

Proof. To prove the reverse direction, suppose that G is neither an odd hole nor a bipartite graph. Let us build a subgraph $H=(V,F)$ of G for which the polytope is not box-TDI. Since $\{x \in \mathbb{R}^F : A_H^\top x \geq \mathbf{1}\}$ is the projection onto F of $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}\}$ intersected with $\{x \in \mathbb{R}^E : x_e = 0, \text{ for all } e \in E \setminus F\}$, this will imply that $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}\}$ is not box-TDI.

Since G is connected, nonbipartite, and different from an odd hole, it contains an odd hole C with $\delta(V(C))$ nonempty. Denote by U the set of vertices of $V \setminus V(C)$ whose neighbors are all in $V(C)$. Let S be a subset of $\delta(U)$ such that each vertex of U is covered by exactly one edge of S. Let $F=(E \setminus \delta(U)) \cup S$, and let $H=(V,F)$. This graph is obtained from G by deleting edges so that every vertex of U has exactly one neighbor in $V(C)$, whereas the vertices of $V \setminus (V(C) \cup U)$ have at least one neighbor but none in $V(C)$.

Let M be the $|V(C)| \times |F|$ matrix formed by the rows of $A_H^\top$ associated with the vertices of $V(C)$. By considering the columns of M associated to C and an edge of $\delta(V(C))$, observe that M contains a matrix of the type $\left[A_C^\top, \chi^u\right]$ for some $u \in V(C)$. Therefore, by Lemma 7, M has full row rank but is not equimodular.

We now show that M is face-defining for $P=\{x \in \mathbb{R}^F : A_H^\top x \geq \mathbf{1}\}$. Since M has full row rank, by Observation 2 it is sufficient to exhibit $|F|-|V(C)|+1$ affinely independent points of the face $Q=P \cap \{x : Mx=\mathbf{1}\}$ of P. Let $K=F \setminus (C \cup \delta(V(C)))$, we define:

$$p^0 = \frac{1}{2}\chi^C + \chi^{S \cup K} + \frac{1}{2}\sum_{u \in V(C)} |\delta(u) \cap S| (\chi^{L_u} - \chi^{C \setminus L_u}),$$

where L_u is the unique perfect matching of the path $C \setminus \delta(u)$. Then, we define two types of points:

- $p^e = p^0 + \chi^e$, for each $e \in K$,
- $q^{uv} = p^0 + \chi^{uv} + \frac{1}{2}(\chi^{L_u} - \chi^{C \setminus L_u})$, for each $uv \in \delta(V(C))$ with $u \in V(C)$.

Together with p^0, the points p are affinely independent because $p^e - p^0 = \chi^e$, for each e in K. Adding the points q maintains affine independence since q^{uv} is the only point with uv's coordinate different from 1.

Moreover, all these points belong to Q since they satisfy $x(\delta(u)) = 1$ for all $u \in V(C)$ and $x(\delta(v)) \geq 1$ for all $v \in V \setminus V(C)$. To see this, note that for each u in $V(C)$, $\chi^{L_u} - \chi^{C \setminus L_u}$ satisfies $x(\delta(u)) = -2$ and $x(\delta(v)) = 0$ for all $v \neq u$. The number of points p is $|K|+1 = |F| - |V(C)| - |\delta(V(C))| + 1$ and the number of points q is $|\delta(V(C))|$, hence M is face-defining for F.

The matrix M is nonequimodular and face-defining for $\{x \in \mathbb{R}^F : A_H^\top x \geq \mathbf{1}\}$. Therefore, the latter is not box-TDI by Theorem 1, and neither is $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}\}$. $\square$

Given a graph $G = (V, E)$, an *edge cover* is a set of edges covering each vertex. The polyhedron $\{x \in \mathbb{R}^E : A_G^\top x \geq \mathbf{1}, x \geq \mathbf{0}\}$ is called the *edge relaxation of the edge cover dominant* of G and its binary points are precisely the characteristic vectors of the edge covers of G.

Since adding box constraints preserves box-TDIness, by Theorem 15, the edge relaxation of the edge cover dominant of an odd hole or a bipartite graph is box-TDI. The converse does not hold, because adding $x \geq \mathbf{0}$ might cut off faces defined by nonequimodular matrices, such as the one given in the proof of Theorem 15. The larger class of graphs to be considered to get the converse is given in Theorem 16 below.

The parallel between Theorems 8 and 16 highlights once more the subtle difference between the TDIness of a system and that of a polyhedron. In particular, for an odd hole C_n, the system $A_{C_n}^\top x \geq \mathbf{1}, x \geq \mathbf{0}$ is not box-TDI while the associated polyhedron is box-TDI. This means that this system is not TDI. This can be seen as the right-hand side is integer but, since n is odd, the point $\frac{1}{2}\mathbf{1}$ is a noninteger vertex of the associated polyhedron [2]. A box-TDI system describing this polyhedron is obtained by adding the inequality $\mathbf{1}^\top x \geq \frac{|C|}{2}$, which is one half of the sum of every inequality in $A_{C_n}^\top x \geq \mathbf{1}$.

Theorem 16. *The edge relaxation of the edge cover dominant of a connected graph G is box-TDI if and only if G is an odd hole or a quasi-bipartite graph different from K_4.*

Proof. Let P_G denote the edge relaxation of the edge cover dominant of G.
($\Leftarrow$) By Theorem 8, if G is a quasi-bipartite graph different from K_4, then the system $A_G^\top x \geq \mathbf{1}, x \geq \mathbf{0}$ is box-TDI, hence P_G is box-TDI.

If G is an odd hole, P_G is the intersection of the polyhedron stated in Theorem 15 with the box $\{x : x \geq \mathbf{0}\}$. Theorem 15 and the definition of box-TDI polyhedra imply that P_G is box-TDI.
($\Rightarrow$) Let us show that P_{K_4} is not box-TDI. By definition, $P_{K_4} = \{x : A_{K_4}^\top x \geq \mathbf{1}, x \geq \mathbf{0}\}$, where

$$A_{K_4}^\top = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

The full row rank matrix formed by the last three rows of $A_{K_4}^\top$, say B, is not equimodular because the determinant of the first three columns is 1, whereas that of the last three is 2. Moreover, the four points $z_1 = (1,0,0,0,0,1)^\top$, $z_2 = (0,1,0,0,1,0)^\top$, $z_3 = (0,0,1,1,0,0)^\top$ and $z_4 = (1,1,1,0,0,0)^\top$ belong to P_G, satisfy $Bx = \mathbf{1}$, and are affinely independent. Therefore, by Observation 2, B is a face-defining matrix of P_{K_4}. This implies that P_{K_4} is not box-TDI by Theorem 1.

To complete the proof there remains to show that P_G is not box-TDI when G is neither quasi-bipartite nor an odd hole. In this case, there exists an odd circuit C such that $G \setminus V(C)$ is nonempty and has no isolated vertices. If C has a chord e, then $C \cup \{e\}$ contains a smaller odd circuit C'. Since $C \setminus C'$ is a path of length at least two, $G \setminus V(C')$ has no isolated vertices. Therefore, we may assume that C is an odd hole.

Let M be the submatrix of $A_G^\top$ formed by the rows associated with the vertices of $V(C)$. By construction, $\delta(V(C))$ is nonempty, hence M contains $\left[A_C^\top, \chi^u\right]$, for some $u \in V(C)$. By Lemma 7, M is not equimodular.

We show that M is face-defining for P_G. Since M has full row rank, by Observation 2 it is sufficient to exhibit $|E| - |V(C)| + 1$ affinely independent points of the face $F = P_G \cap \{x : Mx = \mathbf{1}\}$ of P_G. We exhibit the same points as in the proof of Theorem 15, the difference is that, here, the set U is empty since there are no isolated vertices when removing $V(C)$:

- $p^0 = \frac{1}{2}\chi^C + \chi^K$,
- $p^e = p^0 + \chi^e$, for each $e \in K$,
- $q^{uv} = \chi^{uv} + \chi^{L_u} + \chi^K$, for each $uv \in \delta(V(C))$ with $u \in V(C)$.

As shown in the proof of Theorem 15, these points are affinely independent and satisfy $A_G^\top x \geq \mathbf{1}, Mx = \mathbf{1}$. Since these points also satisfy $x \geq \mathbf{0}$, they belong to the face $P_G \cap \{x : Mx = \mathbf{1}\}$ for which M is a face-defining matrix. By Theorem 1, P_G is not box-TDI. □

Theorem 16 implies that recognizing box-TDI polyhedra is co-NP-complete since recognizing quasi-bipartite graphs is [6].

Corollary 17. *Recognizing box-TDI polyhedra is co-NP-complete.*

Conclusion

In this paper, we provide two hardness results regarding box-TDI polyhedra, and their proofs are based on the exhibition of new classes of binary TE matrices. A natural subsequent question is the characterization of binary TE matrices, and further that of $\{0, 1, -1\}$ TE matrices.

Data availability

No data was used for the research described in the article.

Acknowledgments

We are indebted to the anonymous referees for their valuable comments. This work has been partially supported by the Jacques Hadamard Foundation, France, PGMO project P-2019-0015, ‘‘Matrices Totalement Équimodulaires’’.

References

[1] A. Schrijver, Theory of linear and integer programming, in: Wiley-Interscience Series in Discrete Mathematics and Optimization, 1986.
[2] J. Edmonds, R. Giles, A min-max relation for submodular functions on graphs, in: P. Hammer, E. Johnson, B. Korte, G. Nemhauser (Eds.), Studies in Integer Programming, in: Annals of Discrete Mathematics, vol. 1, Elsevier, 1977, pp. 185–204.
[3] W.J. Cook, On box totally dual integral polyhedra, Math. Programmming 34 (1) (1986) 48–61.
[4] D. König, Gráfok és mátrixok, Mate. Fizikai Lapok 38 (1931) 116–119.
[5] A.J. Hoffman, J.B. Kruskal, Integral boundary points of convex polyhedra, in: 50 Years of Integer Programming 1958-2008: From the Early Years To the State-of-the-Art, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 49–76.
[6] G. Ding, L. Feng, W. Zang, The complexity of recognizing linear systems with certain integrality properties, Math. Program. 114 (2) (2008) 321–334.
[7] X. Chen, G. Ding, W. Zang, The box-TDI system associated with 2-edge connected spanning subgraphs, Discrete Appl. Math. 157 (1) (2009) 118–125.
[8] G. Ding, L. Tan, W. Zang, When is the matching polytope box-totally dual integral? Math. Oper. Res. 43 (1) (2018) 64–99.
[9] J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices, J. Res. Natl. Bur. Stand. 69B (1965).
[10] W.H. Cunningham, A.B. Marsh, A primal algorithm for optimum matching, in: Polyhedral Combinatorics: Dedicated To the Memory of D.R. Fulkerson, Springer Berlin Heidelberg, Berlin, Heidelberg, 1978, pp. 50–72.
[11] G. Ding, W. Zang, Q. Zhao, On box-perfect graphs, J. Combin. Theory Ser. B 128 (2018) 17–46.
[12] D. Cornaz, R. Grappe, M. Lacroix, Trader multiflow and box-TDI systems in series-parallel graphs, Discrete Optim. 31 (1) (2019).
[13] M. Barbato, R. Grappe, M. Lacroix, E. Lancini, R. Wolfler Calvo, The Schrijver system of the flow cone in series–parallel graphs, Discrete Appl. Math. 308 (2022) 162–167.
[14] M. Barbato, R. Grappe, M. Lacroix, E. Lancini, Box-total dual integrality and edge-connectivity, Math. Program. 197 (2023) 307–336.
[15] P. Chervet, R. Grappe, L. Robert, Box-total dual integrality, box-integrality, and equimodular matrices, Math. Program. 188 (1) (2021) 319–349.
[16] J. Pap, Recognizing conic TDI systems is hard, Math. Program. 128 (1) (2011) 43–48.
[17] P.D. Seymour, Decomposition of regular matroids, J. Comb. Theory. Ser. B 28 (3) (1980) 305–359.
[18] J.W. Grossman, D.M. Kulkarni, I.E. Schochetman, On the minors of an incidence matrix and its smith normal form, Linear Algebra Appl. 218 (1995) 213–224.
[19] P.D. Seymour, Matroid minors, in: Handbook of Combinatorics, Elsevier, Amsterdam, 1995, pp. 527–550.
[20] K. Kawarabayashi, K. Ozeki, A simpler proof for the two disjoint odd cycles theorem, J. Combin. Theory Ser. B 103 (3) (2013) 313–319.
[21] M. Conforti, S. Fiorini, T. Huynh, S. Weltge, Extended formulations for stable set polytopes of graphs without two disjoint odd cycles, in: D. Bienstock, G. Zambelli (Eds.), Integer Programming and Combinatorial Optimization, Springer International Publishing, Cham, 2020, pp. 104–116.
[22] C. Haase, A. Paffenholz, L. Piechnik, F. Santos, Existence of unimodular triangulations - positive results, Mem. Amer. Math. Soc. 270 (2014).
[23] K. Kawarabayashi, B. Reed, Odd cycle packing, in: Proceedings of the Forty-Second ACM Symposium on Theory of Computing, STOC ’10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 695–704.
[24] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher, J.D. Bohlinger (Eds.), Complexity of Computer Computations, Springer US, Boston, MA, 1972, pp. 85–103.

Discrete Optimization 31 (2019) 103–114

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Trader multiflow and box-TDI systems in series–parallel graphs

Denis Cornaz[a,*], Roland Grappe[b], Mathieu Lacroix[b]

[a] *Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France*
[b] *Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS (UMR 7030), F-93430, Villetaneuse, France*

A R T I C L E I N F O

Article history:
Received 18 May 2018
Received in revised form 25 September 2018
Accepted 27 September 2018
Available online 22 November 2018

MSC:
90C27
90C57
05C70
05C85

Keywords:
Box-TDI system
Series–parallel graph
Multiflow

A B S T R A C T

Series–parallel graphs are known to be precisely the graphs for which the standard linear systems describing the cut cone, the cycle cone, the T-join polytope, the cut polytope, the multicut polytope and the T-join dominant are TDI. We prove that these systems are actually box-TDI. As a byproduct, our result yields a min–max relation for a new problem: the trader multiflow problem. The latter generalizes both the maximum multiflow and min-cost multiflow problems and we show that it is polynomial-time solvable in series–parallel graphs.

1. Introduction

Throughout the paper, all the entries will be rational. A linear system $Ax \geq b, x \geq 0$ is *totally dual integral* (*TDI* for short) if the maximum in the LP-duality equation

$$\min\{c^\top x : Ax \geq b,\ x \geq \mathbf{0}\} = \max\{b^\top y : A^\top y \leq c,\ y \geq \mathbf{0}\}$$

has an integer optimal solution for all integer vectors c for which the optimum is finite. This property is much sought-after since such systems describe integer polyhedra when b is integer and yield min–max relations [1]. An even stronger property than TDIness is box-TDIness, where a *box-TDI system* is a TDI system $Ax \geq b, x \geq \mathbf{0}$ which remains TDI when adding box-constraints $\ell \leq x \leq u$, for all rational[1] vectors ℓ, u. In other words, it is box-TDI if

$$\max\{b^\top y + \ell^\top z^1 - u^\top z^2 : A^\top y + z^1 - z^2 \leq c,\ y \geq \mathbf{0},\ z^1, z^2 \geq \mathbf{0}\}$$

* Corresponding author.
E-mail addresses: denis.cornaz@dauphine.fr (D. Cornaz), grappe@lipn.univ-paris13.fr (R. Grappe), lacroix@lipn.univ-paris13.fr (M. Lacroix).
[1] Allowed to take infinite values.

https://doi.org/10.1016/j.disopt.2018.09.003
1572-5286/© 2018 Elsevier B.V. All rights reserved.

has an integer solution for all integer vectors c and all rational vectors ℓ, u for which the optimum is finite. General properties of such systems can be found in Cook [2] and Chapter 22.4 of Schrijver [3]. Note that, although every rational polyhedron $\{x : Ax \geq b, x \geq \mathbf{0}\}$ is described by a TDI system $\frac{1}{k}Ax \geq \frac{1}{k}b, x \geq \mathbf{0}$, for some integer k, not every polyhedron is described by a box-TDI system.

The book by Schrijver [4] contains numerous min–max relations of combinatorial optimization derived from TDI systems. When such systems are box-TDI, most of the time, the matrix A is totally unimodular. The past few years, this topic has received a renewed interest [5,6], and other box-TDI systems have been studied [7–9], with matrices that are not totally unimodular. A 0–1 matrix A so that the linear system $Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ is (box-) TDI is called *(box-) Mengerian*. In 1977, Seymour [10] proved that a 0–1 matrix associated with a binary clutter is Mengerian if and only if it does not contain Q_6 as a minor. In 2008, Chen, Ding and Zang [8] proved that such matrices are box-Mengerian if and only if they contain neither Q_6 nor Q_7 as a minor. Recently, Ding, Tan and Zang [11] announced a characterization of the graphs for which a box-TDI system describes the matching polytope.

In 2009, Chen, Ding and Zang [9] proved that a graph is series–parallel if and only if the system $\frac{1}{2}Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ describing the 2-edge-connected spanning subgraph polytope is box-TDI, where A is the cut-edge incidence matrix of the graph. Another set of characterizations of series–parallel graphs given by Schrijver asserts that they are precisely the graphs for which the standard linear systems describing the cut cone, the cycle cone [12], the cut polytope [13], the T-join polytope [14] and the T-join dominant [15] are TDI — see Corollary 29.9c of [4]. Moreover, it is proved in [16] that a graph is series–parallel if and only if the standard linear system describing its multicut polytope is TDI.

Multiflows are among the most famous NP-hard problems in combinatorial optimization and have been considerably studied, see for instance [4]. We focus on integer multiflows in the present paper. Multiflow problems involve two simple undirected graphs, a *supply graph* $G = (V, E)$ and a *demand graph* $H = (V, R)$, and two vectors, a *capacity vector* $c \in \mathbb{Z}_+^E$ and a *demand vector* $d \in \mathbb{Z}_+^R$. An edge $e \in E$ is a *link* of *capacity* c_e whereas an edge $r \in R$ is a *net* of *demand* d_r. From now on, (G, H, c, d) will refer to such a quadruplet. For a net $r = st$, let $\mathcal{P}(r)$ be the set of all st-paths in G, and let $\mathcal{P}$ be the union of $\mathcal{P}(r)$ for all nets r. A *multiflow of* (G, H, c, d) is an integer vector $y \in \mathbb{Z}^{\mathcal{P}}$ satisfying the following system of linear inequalities:

$$(\textsc{mflow}) \begin{cases} \displaystyle\sum_{P \in \mathcal{P}(r)} y_P \geq d_r & \text{for each net } r \in R, \\ \displaystyle\sum_{P \in \mathcal{P}\colon e \in P} y_P \leq c_e & \text{for each link } e \in E, \\ \qquad\quad y \geq \mathbf{0}. \end{cases}$$

Two famous NP-hard problems are related to multiflows. Given G, H and c, the *maximum multiflow problem* asks for a demand vector d such that there exists a multiflow for (G, H, c, d) and $\sum_{r \in R} d_r$ is maximum.

Given (G, H, c, d) and some cost vector $w \in \mathbb{Z}_+^E$ on the links, the *min-cost multiflow problem* asks for a multiflow minimizing the sum of $w_e y_e$ over all links $e \in E$, where $y_e := \sum_{P \in \mathcal{P}\colon e \in P} y_P$ is the amount of flow through link e.

A necessary condition for the existence of a multiflow in (G, H, c, d) is the *cut condition* which requires that $d(D \cap R) \leq c(D \cap E)$ for all cuts D of $G + H$, the latter being $G + H = (V, E \cup R)$ where E and R are considered as disjoint, that is, $G + H$ may contain parallel edges. Seymour [14] proved that a graph (V, F) is series–parallel if and only if for all partitions F into E and R, and for all $c \in \mathbb{Z}_+^E$ and $d \in \mathbb{Z}_+^R$, the cut condition implies the existence of a multiflow.

Contribution. In this paper, we investigate some box-TDI systems related to multiflows. Our main result is to strengthen the TDI characterizations of series–parallel graphs mentioned earlier by proving that the standard linear systems describing the cut cone, the cycle cone, the T-join polytope, the cut polytope, the multicut polytope, and the T-join dominant are actually box-TDI systems for series–parallel graphs — see Theorem 1.

From the box-TDIness of the cut cone, we derive a min–max relation for series–parallel graphs that involves a new multiflow problem generalizing both the maximum multiflow and min-cost multiflow problems. Given (G, H, c, d), a *profit* $\ell \in \mathbb{Z}_+^R$ and a *cost* $u \in \mathbb{Z}_+^E$, the *trader multiflow problem* asks to maximize $\ell^\top z^1 - u^\top z^2$ over all $(y, z^1, z^2) \in \mathbb{Z}_+^{\mathcal{P}} \times \mathbb{Z}_+^R \times \mathbb{Z}_+^E$ such that y is a multiflow of $(G, H, \tilde{c}, \tilde{d})$ with $\tilde{c} = c + z^2$ and $\tilde{d} = d + z^1$. Therefore, in this new multiflow problem, we gain ℓ_r for each additional unit of demand on net $r \in R$ that we are able to satisfy, we pay u_e to add a unit of capacity on link $e \in E$, and the goal is to maximize the total benefit. The min–max relation we derive connects the trader multiflow problem and box-multicuts, where box-multicuts are a generalization of multicuts. We also show that the trader multiflow problem is polynomial time solvable in series–parallel graphs.

Outline. In Section 2, we establish our characterization of series–parallel graphs in terms of box-TDI systems. Section 3 is devoted to the trader multiflow problem. We first show how it generalizes both the maximum multiflow and min-cost multiflow problems. Then, we provide our min–max relation for the trader multiflow problem in series–parallel graphs and explain why this problem is polynomial in these graphs. For the sake of clarity, the most technical part of the proof of Theorem 1 is postponed to the Appendix. The rest of this section is devoted to definitions.

Definitions. Throughout, $G = (V, E)$ will denote an undirected graph and $T \subseteq V$ a set of vertices of even cardinality. A graph is *series–parallel* if it is obtained from a forest by repeating the operations of replacing one edge by two edges in parallel, or by two edges in series. Equivalently, these are the graphs without K_4 minor [17]. Then, a series–parallel graph is planar and its planar dual is also series–parallel. Following [4], a *cycle* is a subset $C \subseteq E$ so that every vertex of (V, C) has an even degree. A minimal nonempty cycle is a *circuit*. The *cut* defined by a subset of vertices U, denoted by $\delta(U)$, is the set of edges having one extremity in U and the other one in $V \setminus U$. A minimal nonempty cut is a *bond*. Note that cycles (resp. cuts) are disjoint unions of circuits (resp. bonds). A *multicut* is the set of all the edges between different classes of some partition of the vertex set. A *T-join* is a subset of edges F such that the odd degree vertices of (V, F) are the ones in T. Note that a cycle is an $\emptyset$-join. A *T-cut* is a cut $\delta(U)$ with $|U \cap T|$ odd. For $x \in \mathbb{R}^E$ and $F \subseteq E$, we use the notation $x(F) = \sum_{e \in F} x_e$. We will make no difference between combinatorial objects and their characteristic vectors, that is, for instance, we will speak of nonnegative combinations of cycles instead of nonnegative combinations of characteristic vectors of cycles.

2. Box-TDI systems of series–parallel graphs

In this section, we first provide the systems involved in our main theorem. Then, we state and prove Theorem 1, which establishes that the standard linear systems describing the cut cone, the cycle cone, the T-join polytope, the cut polytope, the multicut polytope and the T-join dominant are box-TDI if and only if the graph is series–parallel. These systems were already known to be TDI [4,16].

2.1. TDI systems of series–parallel graphs...

Let us write now the systems involved in Theorem 1. Let $G = (V, E)$ be an undirected graph and $T \subseteq V$ a set of vertices of even cardinality.

Seymour [12] proved that the *cycle cone* of G, that is, the set of nonnegative combinations of cycles of G, is described by the following set of inequalities.

$$\text{(Cycle cone)} \begin{cases} x(\delta(U) \setminus \{e\}) - x_e \geq 0 & \text{for each } U \subseteq V \text{ and each } e \in \delta(U), \\ x \geq \mathbf{0}. \end{cases}$$

The *T-join polytope* of G is the convex hull of its T-joins. Seymour [14] proved that it is described by the following set of inequalities.

$$(T\text{-join})\begin{cases} x(F) - x(\delta(U)\setminus F) \le |F| - 1 & \text{for each } U \subseteq V,\ F \subseteq \delta(U) \\ & \text{with } |U \cap T| + |F| \text{ odd,} \\ \mathbf{0} \le x \le \mathbf{1}. & \end{cases}$$

The T-join dominant of G is the set of vectors greater than or equal to some T-join of G. This dominant is described by the following set of inequalities, see Corollary 29.2b in [4].

$$(T\text{-join dominant})\begin{cases} x(C) \ge 1 & \text{for each } T\text{-cut } C, \\ x \ge \mathbf{0}. & \end{cases}$$

Sebő [18] provided a minimal TDI system describing the T-join dominant of G.

Let us assume that G is planar and let G^* denote its dual graph. Recall that the cycles of G are the cuts of G^*. Hence,

$$(\text{Cut cone})\begin{cases} x(C\setminus\{e\}) - x_e \ge 0 & \text{for each circuit } C \text{ and each edge } e \in C, \\ x \ge \mathbf{0}, & \end{cases}$$

describes the *cut cone* of G, that is, the set of nonnegative combinations of cuts of G. Moreover, by taking $T = \emptyset$ in system (T-join), and then writing the planar dual, we have the following description of the *cut polytope* of G, that is, the convex hull of its cuts.

$$(\text{Cut})\begin{cases} x(F) - x(C\setminus F) \le |F| - 1 & \text{for each circuit } C \text{ and } F \subseteq C \\ & \text{with } |F| \text{ odd,} \\ \mathbf{0} \le x \le \mathbf{1}. & \end{cases}$$

Actually, the systems (Cut cone) and (Cut) describe the cut cone and the cut polytope for a larger class than planar graphs, namely graphs with no K_5-minor — see [14] and [13], respectively.

Schrijver showed that the systems (Cycle cone), (T-join) and (T-join dominant) are TDI if and only if the graph is series–parallel— see Corollary 29.9c of [4]. A graph is series–parallel if and only if its dual is; this result, combined with the fact that cycles are $\emptyset$-joins, implies that (Cut cone) and (Cut) are TDI if and only if the graph is series–parallel.

Multicuts can be equivalently defined as arbitrary unions of cuts, or as sets of edges $D \subseteq E$ such that $|D \cap C| \ne 1$ for all cycles C. The *multicut polytope* of a graph is the convex hull of its multicuts, and is therefore contained in the polyhedron defined by the inequalities of (Cut cone) and $x \le \mathbf{1}$. Chopra [19] showed that the following system, called (Multicut), describes the multicut polytope of a graph if and only if the graph is series–parallel.

$$(\text{Multicut})\begin{cases} x(C\setminus\{e\}) - x_e \ge 0 & \text{for each circuit } C \text{ and each edge } e \in C, \\ \mathbf{0} \le x \le \mathbf{1}. & \end{cases}$$

Corollary 4.1 of [16] strengthens the result of Chopra [19] by stating that system (Multicut) is TDI if and only if the graph is series–parallel.

2.2. . . . are actually box-TDI

We now strengthen the aforementioned TDIness results. More precisely, we show that each system mentioned in Section 2.1 which is TDI for series–parallel graphs is actually box-TDI for these graphs. Our theorem implies Corollary 4.1 of [16] and Corollary 29.9c of [4].

Theorem 1. *Let $G = (V, E)$ be a graph. The following statements are equivalent.*

(i) G is series–parallel.
(ii) System (Cut cone) is box-TDI.
(iii) System (Cycle cone) is box-TDI.
(iv) System (T-join) is box-TDI, for all $T \subseteq V$ of even cardinality.
(v) System (Cut) is box-TDI.
(vi) System (Multicut) is box-TDI.
(vii) System (T-join dominant) is box-TDI, for all $T \subseteq V$ of even cardinality.

Proof. Proof. Series-parallelness is already necessary for the systems of (ii)–(vii) to be TDI — see [16] for (vi) and Corollary 29.9c of [4] for the others. A box-TDI system being TDI, the necessity of (i) follows. For the other directions, we will show that $(i) \Rightarrow (ii) \Rightarrow (iii) \Rightarrow (iv) \Rightarrow (v)$ and $(ii) \Rightarrow (vi)$ and $(iv) \Rightarrow (vii)$.

$(i) \Rightarrow (ii)$: Let $G = (V, E)$ be series–parallel, $c \in \mathbb{Z}^E$ and $\ell, u \in \mathbb{Q}^E$ with $\ell \leq u$. The primal problem is to optimize over the system (Cut cone) intersected with the box $\{x : \ell \leq x \leq u\}$. Since we have $x \geq \mathbf{0}$, we may suppose that $\ell \geq \mathbf{0}$ and we get:

$$(P)\begin{cases} \min c^\top x \\ x(C \setminus \{e\}) - x_e \geq 0 \qquad \text{for each circuit } C \text{ of } G \text{ and each edge } e \in C, \\ \mathbf{0} \leq \ell \leq x \leq u. \end{cases}$$

To prove box-TDIness, one has to show that if the dual given below has an optimal solution, then it also has an integer one.

$$(D)\begin{cases} \max \ell^\top z^1 - u^\top z^2 \\ \displaystyle\sum_{\text{circuit } C \ni e} \Big(\sum_{f \in C \setminus \{e\}} y_{C,f} \quad - \quad y_{C,e} \Big) \leq c_e - z^1_e + z^2_e \qquad \text{for each } e \in E, \\ y \geq \mathbf{0}, \quad z^1, z^2 \geq \mathbf{0}. \end{cases}$$

The feasible set for (D) has the form $Q = \{z^1, z^2 \geq \mathbf{0},\ y \geq \mathbf{0}\colon z^1 - z^2 + Ay \leq c\}$, and its projection onto the space of $z = (z^1, z^2) \in \mathbb{R}^{E \times E}$ is $\text{proj}_z(Q) = \{z^1, z^2 \geq \mathbf{0} : v^\top z^1 - v^\top z^2 \leq v^\top c, \text{ for each } v \in K\}$ where K is the projection cone $K = \{v \in \mathbb{R}^E : v^\top A \geq \mathbf{0}^\top, v \geq \mathbf{0}\}$. Observe that K is the set of $v \in \mathbb{R}^E$ satisfying the inequalities of the system (Cut cone). Since G is series–parallel, K is the cut cone of G [14]. Therefore

$$\text{proj}_z(Q) = \{(z^1, z^2) \in \mathbb{R}_+^{E \times E} : z^1(D) - z^2(D) \leq c(D), \text{ for each cut } D \text{ of } G\}.$$

The following claim states that $\text{proj}_z(Q)$ is an integer polyhedron. It is a direct corollary of a technical result whose statement and proof are postponed to the Appendix.

Claim 2. *$\text{proj}_z(Q)$ is integer whenever c is integer.*

Suppose (D) has an optimal solution. By Claim 2, there exists an integer optimal solution $(\bar{z}^1, \bar{z}^2)$ of $\max \ell^\top z^1 - u^\top z^2$ over $\text{proj}_z(Q)$. We now build an optimal solution $(\bar{y}, \bar{z}^1, \bar{z}^2)$ of (D) as follows.

Let $b := c - \bar{z}^1 + \bar{z}^2$. Then b is integer and satisfies $b(D) \geq 0$ for each cut D of G. Define R as the set of all $e \in E$ with $b_e \leq 0$ and $E' = E \setminus R$. Let $G' = (V, E')$ and $H = (V, R)$. Let $c' \in \mathbb{Z}_+^{E'}$ and $d \in \mathbb{Z}_+^R$ be defined by $c'_e = b_e$ for all $e \in E'$ and $d_r = -b_r$ for all $r \in R$. Then $d(D \cap R) \leq c'(D \cap E')$ for each cut D of $G' + H$. In other words, the cut condition is satisfied in (G', H, c', d). Hence, $G' + H = G$ being series–parallel, Theorem 8.1 of [14] implies that there exists a multiflow $\hat{y}$ of (G', H, c', d). Define $\bar{y}$ as follows:

$$\bar{y}_{C,e} := \begin{cases} \hat{y}_P & \text{if } b_e \leq 0 \text{ and } P = C \setminus \{e\}, \\ 0 & \text{otherwise.} \end{cases}$$

By construction, $(\bar{y}, \bar{z}^1, \bar{z}^2)$ is an integer optimal solution of (D), and we are done.

$(ii) \Rightarrow (iii)$: The system (Cycle cone) of a series–parallel graph is the system (Cut cone) of its planar dual which is also a series–parallel graph. As the latter system is box-TDI precisely for such graphs, we get the desired implication.

$(iii) \Rightarrow (iv)$: In the following, $Ax \leq b$ is a system whose underlying polyhedron $P = \{x : Ax \leq b\}$ is pointed. The *vertex system* associated with a vertex z of $\{x : Ax \leq b\}$ is the system $A_z x \leq b_z$ composed of the inequalities of $Ax \leq b$ satisfied with equality by z.

Claim 3. *The system $Ax \leq b$ is box-TDI if and only if the vertex system associated with each vertex of $P = \{x : Ax \leq b\}$ is box-TDI.*

Proof. Cook proves that a system is box-TDI if and only if, for each face F of the associated polyhedron, the set of active rows for F forms a box Hilbert basis [2, Proposition 2.2].

Suppose that all the vertex systems of P are box-TDI. Let F be a proper face of P and z be a vertex of F. Then, the active rows in $A_z x \leq b_z$ for the minimal face of $\{x : A_z x \leq b_z\}$ containing F are exactly the same as those in $Ax \leq b$ for F. Hence, by [2, Proposition 2.2], the set of active rows for F forms a box Hilbert basis. Since this holds for every face of P, [2, Proposition 2.2] implies that $Ax \leq b$ is box-TDI. The converse can be proved in a similar way. ■

Let $T \subseteq V$. Recall that vertices of the polytope defined by the system (T-join) correspond to T-joins of G, and conversely. Let J be any T-join of G. By Claim 3, it suffices to show that the vertex system of (T-join) associated with vertex J is box-TDI. Let $\phi_J : \mathbb{R}^E \to \mathbb{R}^E$ be defined by

$$[\phi_J(x)]_e := \begin{cases} 1 - x_e & \text{if } e \in J, \\ x_e & \text{if } e \in E \setminus J. \end{cases}$$

The next two claims exhibit properties of ϕ_J.

Claim 4. *The system obtained from (Cycle cone) by replacing x by $\phi_J(x)$ is the vertex system of (T-join) associated with J.*

Proof. Schrijver proves that replacing x by $\phi_J(x)$ in the vertex system of (T-join) associated with J gives the system (Cycle cone) — see (29.61) to (29.63) page 506 in [4] for the details. As $\phi_J(\phi_J(x)) = x$, the assertion follows. ■

Claim 5. *Replacing x by $\phi_J(x)$ preserves box-TDIness.*

Proof. From the definition of box-TDI systems, it follows that replacing some coordinates by their opposite preserves box-TDIness. So does translation, see Theorem 5.34 in [4]. ■

The (Cycle cone) being box-TDI by (iii), Claims 4 and 5 imply the box-TDIness of the vertex system of (T-join) associated with J. Since this holds for any T-join J of G, Claim 3 gives the box-TDIness of (T-join).

$(iv) \Rightarrow (v)$: We have already shown that (T-join) is box-TDI if and only if the graph is series–parallel. Recall that the cuts of a planar graph are the cycles of its planar dual, and that cycles are $\emptyset$-joins. Therefore, (Cut) is nothing but the system ($\emptyset$-join) for the planar dual of the graph, and since planar duality preserves series–parallelness, we get that (iv) implies (v).

$(ii) \Rightarrow (vi)$: This is immediate because (Multicut) is nothing but the box-TDI system (Cut cone) together with the box-constraints $x \leq \mathbf{1}$.

$(iv) \Rightarrow (vii)$: The system describing the T-join polytope being box-TDI, the TDI system (T-join dominant) describing its dominant is also box-TDI — by Theorem 22.11 of [3]. □

Box-TDI systems have the remarkable property that any TDI system describing the same polyhedron is also box-TDI [2]. This gives the following consequence of Theorem 1. The minimal TDI system describing the T-join dominant given by Sebő [18] becomes box-TDI when the graph is series–parallel.

3. Trader multiflow vs box-multicut

In this section, we first explain how the trader multiflow problem generalizes both the min-cost multiflow and maximum multiflow problems. We then provide a min–max relation involving the trader multiflow problem and the so-called box-multicuts. Finally, we briefly explain why the trader multiflow problem is polynomial in series–parallel graphs.

3.1. Related multiflow problems

Recall that an instance (G, H, c, d, ℓ, u) of the trader multiflow problem is composed of two simple undirected graphs $G = (V, E)$ and $H = (V, R)$, a capacity $c \in \mathbb{Z}_+^E$, a demand $d \in \mathbb{Z}_+^R$, a profit $\ell \in \mathbb{Z}_+^R$ and a cost $u \in \mathbb{Z}_+^E$. The trader multiflow problem aims at maximizing $\ell^\top z^1 - u^\top z^2$ over all $(y, z^1, z^2) \in \mathbb{Z}_+^{\mathcal{P}} \times \mathbb{Z}_+^R \times \mathbb{Z}_+^E$ such that y is a multiflow of $(G, H, \tilde{c}, \tilde{d})$ with $\tilde{c} = c + z^2$ and $\tilde{d} = d + z^1$.

This problem contains the maximum multiflow problem as a special case. Let (G, H, c, d, ℓ, u) be an instance of the trader multiflow problem with $d = \mathbf{0}$, $\ell = \mathbf{1}$ and $u = +\infty$. In any optimal solution $(\bar{y}, \bar{z}^1, \bar{z}^2)$, since $u = +\infty$, we have $\bar{z}^2 = \mathbf{0}$, that is, capacities remain unchanged. Since $d = \mathbf{0}$ and $\ell = \mathbf{1}$, the problem reduces to find $\bar{z}^1$ such that $\sum_{r \in R} \bar{z}_r^1$ is maximum and there exists a multiflow in $(G, H, c, \bar{z}^1)$. This is nothing but the maximum multiflow problem associated with (G, H, c).

The trader multiflow problem also contains the min-cost multiflow problem as a special case. Let (G, H, c, d, w) be an instance of the min-cost multiflow problem. It is transformed into an instance $(G', H', c', d', \ell', u')$ of the trader multiflow problem as follows. Let $G' = (V', E')$ be the graph obtained from G by subdividing every link $e \in E$ into two links e_1, e_2 in series. Then, the amount of flow passing by e_1 equals the amount of flow passing by e_2. Let $c'_{e_1} = c_e$ and $u'_{e_1} = +\infty$. The capacity of e_1 is chosen in order to limit the value of the flow passing by e_1, e_2 to c_e. Let $c'_{e_2} = 0$ and $u'_{e_2} = w_e$. The role of e_2 is to charge a fee w_e for each unit of flow passing by e_1, e_2. Let $H' = (V', R)$, $d' = d$ and $\ell' = \mathbf{0}$. In an optimal solution $(\bar{y}, \bar{z}^1, \bar{z}^2)$ of the trader multiflow problem, we may suppose without loss of generality that $\bar{z}^1 = \mathbf{0}$ since $\ell' = \mathbf{0}$. Since $u'_{e_1} = +\infty$, the amount of flow passing by e_1, e_2 is no more than $c'_{e_1} = c_e$. Since $c'_{e_2} = 0$, for each unit of flow passing by e_1, e_2, one has to increase the capacity of e_2 by one at cost $u'_{e_2} = w_e$. Hence, $\bar{y}$ defines a multiflow in (G, H, c, d) minimizing the total cost of the flow.

3.2. Min–max theorem

Given a graph and integer vectors ℓ and u indexed on its edges, the integer vectors x satisfying system (Cut cone) and $\ell \leq x \leq u$ are called *box-multicuts within* $[\ell, u]$. If we are also given a cost vector c defined on the edges, the *minimum box-multicut problem* seeks a box-multicut x within $[\ell, u]$ of minimum cost $c^\top x$.

Box-multicuts are a generalization of multicuts, these latter being box-multicuts within $[\mathbf{0}, \mathbf{1}]$. Box-multicuts also generalize separating multicuts, where, given a supply graph G and a demand graph $H = (V, R)$, a *separating multicut* is a multicut of $G + H$ containing R. Indeed, separating multicuts are box-multicuts of $G + H$ within $[\ell, \mathbf{1}]$ where ℓ equals 1 for every net of R and 0 otherwise.

The min–max relation between the trader multiflow and minimum box-multicut problems given in the following Corollary 6 is a consequence of Theorem 1. Its statement uses the following notation: given a supply graph $G = (V, E)$ and a demand graph $H = (V, R)$ and two vectors $v^1 \in \mathbb{Z}_+^E$ and $v^2 \in \mathbb{Z}_+^R$, the vector associated with the edges of $G + H$ defined by v^1 and v^2 is denoted by (v^1, v^2).

Corollary 6. *The maximum trader multiflow of (G, H, c, d, ℓ, u) equals the minimum box-multicut of $G+H$ within $[(\mathbf{0}, \ell), (u, +\infty)]$ with respect to costs $(c, -d)$, if $G+H$ is series–parallel.*

Proof. First, set $\hat{c} = (c, -d)$, $\hat{\ell} = (\mathbf{0}, \ell)$ and $\hat{u} = (u, +\infty)$. Consider the linear program (P) of the proof of Theorem 1 where G, c, ℓ and u are replaced by $G+H$, $\hat{c}$, $\hat{\ell}$ and $\hat{u}$, respectively. Since $\hat{\ell}_e = 0$, we may suppose, without loss of generality, that $\bar{z}^1_e = 0$ for all links $e \in E$ in an optimal solution $(\bar{y}, \bar{z}^1, \bar{z}^2)$ of the dual (D). Moreover, as $u_r = +\infty$, $\bar{z}^2_r = 0$ for all nets $r \in R$. The dual can then be written as:

$$(D') \begin{cases} \max \sum_{r \in R} \ell_r z^1_r - \sum_{e \in E} u_e z^2_e \\ \sum_{\text{circuit } C \ni r} \left(y_{C,r} - \sum_{f \in C \setminus \{r\}} y_{C,f} \right) \geq d_r + z^1_r & \text{for each } r \in R, \\ \sum_{\text{circuit } C \ni e} \left(\sum_{f \in C \setminus \{e\}} y_{C,f} - y_{C,e} \right) \leq c_e + z^2_e & \text{for each } e \in E, \\ y \geq \mathbf{0}, \quad z^1, z^2 \geq \mathbf{0}. \end{cases}$$

By strong duality, the optimal values of (P) and (D') are equal, when finite. In this case, we will show that there exists an integer optimal solution for both problems.

We may suppose that $\bar{y}_{C,f} = 0$ if $f \in E$. Otherwise, one may decrease $\bar{y}_{C,f}$ by some $\epsilon > 0$. If the solution becomes infeasible, then there exist a circuit $C' \ni f$ and link $f' \in C' \setminus \{f\}$ with $\bar{y}_{C',f'} \geq \epsilon$ since $c \geq \mathbf{0}$. Decreasing $\bar{y}_{C',f'}$ by ϵ and increasing $\bar{y}_{C'',f'}$ by ϵ where C'' is the circuit of $C \Delta C'$ containing f' restores its feasibility. Similarly, we may suppose that $\bar{y}_{C,f} = 0$ if $C \setminus f$ intersects R. Thus, for every $\bar{y}_{C,f} > 0$, $f \in R$ and $C \setminus f \in \mathcal{P}(r)$. Since $G+H$ is series–parallel, system (Cut cone) is box-TDI and $(\bar{y}, \bar{z}^1, \bar{z}^2)$ may be assumed integer. The latter then corresponds to an optimal solution to the trader multiflow problem. Finally, since $\hat{\ell}$ and $\hat{u}$ are integer, the box-TDIness of system (Cut cone) implies that the optimal solution of (P) is integer, that is, a box-multicut of $G+H$ within $[\hat{\ell}, \hat{u}]$. □

Min–max relations involving min-cost multiflow and maximum multiflow stem from Corollary 6 since the transformations described in Section 3.1 preserve series-parallelness. In particular, Corollary 6 implies that the two following min–max relations of [16] that hold if $G+H$ is series–parallel:

- the maximum multiflow equals the minimum separating multicut,
- the minimum multiflow loss equals the maximum multicut,

where the *minimum multiflow loss problem* asks to remove a minimum number of demands of H to ensure the existence of a multiflow in $G+H$.

Applying the arguments used in the proof of $(i) \Rightarrow (ii)$ of Theorem 1, it can be shown that optimizing over (D') amounts to optimize over an integer polyhedron similar to $\text{proj}_z(Q)$. For series–parallel graphs, optimizing over such a polyhedron is polynomial-time solvable [20,21]. It yields an increase of capacities and demands which maximizes the objective function and ensures that the cut condition is satisfied. Then, applying Theorem 8.1 of [14] provides an optimal solution to the trader multiflow problem. To sum up, we have the following complexity result.

Corollary 7. *If $G+H$ is series–parallel, then the maximum trader multiflow problem on (G, H, c, d, ℓ, u) is polynomial-time solvable for all vectors ℓ and u and for all integer vectors c and d.*

As seen in Corollary 7, our approach yields a polynomial algorithm, however it relies on the ellipsoid method. We conclude with the question: is there a combinatorial algorithm that solves the trader multiflow problem in series–parallel graphs?

Acknowledgment

The very careful reading of the paper by an anonymous referee has allowed an improvement of several technical proofs readability, we would like to thank her or him very much.

Appendix

The proof of Theorem 1 is based on Claim 2 which is a direct consequence of the following result.

Lemma 8. *Let $G = (V, E)$ be a graph. The polyhedron $P(G, c)$ defined by*

$$P(G,c) := \{(x,y) \in \mathbb{R}_+^{E\times E} : x(D) - y(D) \leq c(D), \text{ for each cut } D \text{ of } G\}$$

is integer for all integer weights $c \in \mathbb{Z}^E$ if and only if G is series–parallel.

Proof. *Necessity.* First, note that $P(\hat{G}, \hat{c})$ has a fractional extreme point if $\hat{G}$ is the complete graph K_4 with cost $\hat{c}_e = -1$ on the three edges of a triangle and $\hat{c}_e = +1$ on the remaining star. Indeed, the point $\hat{p} = (\hat{x}, \hat{y})$ defined by $\hat{y}_e = 1/2$ for the edges of the triangle and zero elsewhere is the unique optimal solution of maximizing $\hat{\ell}^\top x - \hat{u}^\top y$ over $P(\hat{G}, \hat{c})$, where $\hat{\ell}$ is zero and $\hat{u}$ is the all-one vector. Now, let $\bar{G}$ be a graph which is not series–parallel, then, by [17], it has a K_4-minor, that is we can remove and contract some edges of $\bar{G}$ to obtain K_4. Let us extend $(\hat{c}, \hat{\ell}, \hat{u})$ to $(\bar{c}, \bar{\ell}, \bar{u})$ by defining $\bar{\ell}_e = -\infty$ and $\bar{u}_e = +\infty$ for the new edges e, with $\bar{c}_e = +\infty$ if e must be contracted, and $\bar{c}_e = 0$ if it must be deleted. Clearly, the point $\bar{p}$ obtained by extending $\hat{p}$ with zero components is the unique optimal solution of maximizing $\bar{\ell}^\top x - \bar{u}^\top y$ over $P(\bar{G}, \bar{c})$.

Sufficiency. By contradiction, let (G, c) be a counter-example with a minimum number of edges. Throughout, $\bar{p} = (\bar{x}, \bar{y})$ will denote some fractional extreme point of $P(G, c)$ and

$$\bar{b} := c - \bar{x} + \bar{y}.$$

Note that $\bar{b}(D) \geq 0$, for each cut D.

First, note that G has no loops or bridges. Indeed, a loop belongs to no cut, and a bridge e appears exactly in three nonredundant constraints, namely $x_e \geq 0$, $y_e \geq 0$ and $y_e - x_e \geq c_e$, two of which are satisfied with equality by any extreme point.

Moreover, $P(G, c)$ is full-dimensional. To see this, observe that the point $p = (x, y) \in \mathbb{R}^{E\times E}$ defined by $x_e = 1$ and $y_e = +\infty$ for all $e \in E$ belongs to $P(G, c)$. Moreover, for each edge $e \in E$, the point p_e^x (resp. p_e^y) obtained from p by resetting x_e to zero (resp. y_e to zero) also belongs to $P(G, c)$ since each cut has size at least two. The $2|E| + 1$ points p, p_e^x, p_e^y, for $e \in E$, are affinely independent, hence the dimension of $P(G, c)$ is $2|E|$.

In consequence, the point $\bar{p}$ is the solution of a system of $2|E|$ equations of the following type, where the left-hand-side forms a full-rank matrix.

$$\bar{x}_e = 0 \quad \text{for some edges } e, \tag{A.1}$$

$$\bar{y}_e = 0 \quad \text{for some edges } e, \tag{A.2}$$

$$\bar{x}(D) - \bar{y}(D) = c(D) \quad \text{for some bonds } D \neq \emptyset. \tag{A.3}$$

Suppose G has two parallel edges $\bar{e}$ and $\bar{f}$. Then, replacing $(\bar{x}_{\bar{e}}, \bar{y}_{\bar{e}})$ by $(\bar{x}_{\bar{e}}, \bar{y}_{\bar{e}}) + (\bar{x}_{\bar{f}}, \bar{y}_{\bar{f}})$ and $(\bar{x}_{\bar{f}}, \bar{y}_{\bar{f}})$ by $(0, 0)$ yields a feasible point $(\tilde{x}, \tilde{y})$ because $\bar{e}$ and $\bar{f}$ belong to the same cuts. This point $(\tilde{x}, \tilde{y})$ satisfies all Eqs. (A.1)–(A.3) except possibly the Eqs. (A.1) and (A.2) associated with $\bar{e}$. But these two equations are not satisfied only if $\bar{x}_{\bar{f}} > 0$ or $\bar{y}_{\bar{f}} > 0$ respectively. This implies that $(\tilde{x}, \tilde{y})$ satisfies $2|E|$ equations among (A.1)–(A.3), $\bar{x}_{\bar{f}} = 0$, and $\bar{y}_{\bar{f}} = 0$. Hence, it is also an extreme point of $P(G, c)$. Therefore resetting $c_{\bar{e}} := c_{\bar{e}} + c_{\bar{f}}$ and removing $\bar{f}$ gives a counter-example with a smaller number of edges, a contradiction. We have just proved the following.

$$G \text{ has no parallel edges.} \tag{A.4}$$

Note that, if both $\bar{x}_e > 0$ and $\bar{y}_e > 0$ for some edge e, then one could reset $\bar{x}_e := \bar{x}_e - \varepsilon$ and $\bar{y}_e := \bar{y}_e - \varepsilon$ (for some $\varepsilon > 0$) and still satisfy (A.1)–(A.3), contradicting the extremality of $\bar{p}$. Thus,

$$\text{for all } e, \text{ either } \bar{x}_e = 0 \text{ or } \bar{y}_e = 0. \tag{A.5}$$

We can choose c so as to minimize the norm of $\bar{p}$ (*e.g.* Euclidean). Consequently, nonzero coordinates of $\bar{p}$ are fractional. Indeed, we have

$$\mathbf{0} \leq \bar{p} < \mathbf{1}, \tag{A.6}$$

as otherwise, if $\bar{x}_e \geq 1$ (resp. $\bar{y}_e \geq 1$) for some edge e, then (A.1)–(A.3) would still be satisfied after resetting $\bar{x}_e := \bar{x}_e - 1$ and $c_e := c_e - 1$ (resp. $\bar{y}_e := \bar{y}_e - 1$ and $c_e := c_e + 1$).

By (A.4) and by construction of series–parallel graphs, there are two edges $\bar{e}$ and $\bar{f}$ in series. We may assume w.l.o.g. that $\bar{b}_{\bar{e}} \leq \bar{b}_{\bar{f}}$. Since $\bar{D} = \{\bar{e}, \bar{f}\}$ is a cut, we have $\bar{b}_{\bar{f}} \geq -\bar{b}_{\bar{e}}$. Denote by $\hat{p} = (\hat{x}, \hat{y}) \in \mathbb{R}^{E\setminus\{\bar{f}\}\times E\setminus\{\bar{f}\}}$ the restriction of $\bar{p}$ to $E\setminus\{\bar{f}\}\times E\setminus\{\bar{f}\}$, and let $\hat{G}$ be the graph obtained from G by contracting $\bar{f}$, and $\hat{c}$ the restriction of c to $E \setminus \{\bar{f}\}$. Clearly, $\hat{p}$ belongs to $P(\hat{G}, \hat{c})$, and the latter is full-dimensional since neither loops nor bridges appeared in $\hat{G}$.

Moreover, since c is integer and $\bar{p}$ fractional, (A.3) and (A.5) imply that at least two edges have a fractional $\bar{x}$ or $\bar{y}$ coordinate. Therefore $\hat{p}$ is fractional, and hence, by minimality of $|E|$, $\hat{p}$ is not an extreme point of $P(\hat{G}, \hat{c})$.

Remark that in fact we have:

$$\bar{b}_{\bar{f}} = |\bar{b}_{\bar{e}}| \tag{A.7}$$

If it is not true, then $\bar{p}$ does not saturate the constraint associated to $\bar{D}$, and moreover $\bar{b}_{\bar{f}} > \bar{b}_{\bar{e}}$. Hence, except maybe for $\bar{x}_{\bar{f}} = 0$ or $\bar{y}_{\bar{f}} = 0$, the edge $\bar{f}$ appears in no equation among (A.1)–(A.3). Then $\hat{p}$ is an extreme point, a contradiction.

By the integrality of c, a direct consequence of (A.5)–(A.7) is that:

$$\text{Exactly one of } \bar{x}_{\bar{e}}, \bar{y}_{\bar{e}} \text{ is fractional} \iff \text{exactly one of } \bar{x}_{\bar{f}}, \bar{y}_{\bar{f}} \text{ is fractional.} \tag{A.8}$$

Since $\hat{p}$ is not extreme, there is a (nonzero) direction $\hat{d} = (\hat{d}^x, \hat{d}^y) \in \mathbb{R}^{E\setminus\{\bar{f}\}\times E\setminus\{\bar{f}\}}$ and an $\varepsilon > 0$ such that

$$\hat{p} = \frac{1}{2}(\hat{p} + \varepsilon \cdot \hat{d}) + \frac{1}{2}(\hat{p} - \varepsilon \cdot \hat{d})$$

where both $\hat{p} + \varepsilon \cdot \hat{d}$ and $\hat{p} - \varepsilon \cdot \hat{d}$ belong to $P(\hat{G}, \hat{c})$. Extend the direction $\hat{d} = (\hat{d}^x, \hat{d}^y) \in \mathbb{R}^{E\setminus\{\bar{f}\}\times E\setminus\{\bar{f}\}}$ to a direction $\bar{d} = (\bar{d}^x, \bar{d}^y) \in \mathbb{R}^{E\times E}$ by arbitrarily defining the two missing components $\bar{d}^x_{\bar{f}}$ and $\bar{d}^y_{\bar{f}}$. So

$$\bar{p} = \frac{1}{2}(\bar{p} + \varepsilon \cdot \bar{d}) + \frac{1}{2}(\bar{p} - \varepsilon \cdot \bar{d}) \qquad \forall \varepsilon > 0$$

where the points $\bar{p}^+ = \bar{p} + \varepsilon \cdot \bar{d}$ and $\bar{p}^- = \bar{p} - \varepsilon \cdot \bar{d}$ are different. Since $\bar{p}$ is extreme, we can assume that $\bar{p}^+ = (\bar{x}^+, \bar{y}^+) \notin P(G, c)$. Clearly, we have

$$\bar{x}_{\bar{e}} = 0 \text{ (resp. } \bar{y}_{\bar{e}} = 0) \text{ implies } \bar{d}^x_{\bar{e}} = 0 \text{ (resp. } \bar{d}^y_{\bar{e}} = 0). \tag{A.9}$$

Define $\bar{b}^+ := c - \bar{x}^+ + \bar{y}^+$. By (A.7), there are two cases.

Case 1: $\bar{b}_{\bar{e}} = \bar{b}_{\bar{f}} \geq 0$.

Define

$$\bar{d}^x_{\bar{f}} = \begin{cases} \bar{d}^x_{\bar{e}} - \bar{d}^y_{\bar{e}} & \text{if } \bar{x}_{\bar{f}} > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{d}^y_{\bar{f}} = \begin{cases} \bar{d}^y_{\bar{e}} - \bar{d}^x_{\bar{e}} & \text{if } \bar{y}_{\bar{f}} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

By definition of $\bar{d}$, and by (A.8)–(A.9), we have

$$\bar{b}^+_{\bar{e}} - \bar{b}_{\bar{e}} = (\bar{y}^+_{\bar{e}} - \bar{y}_{\bar{e}}) - (\bar{x}^+_{\bar{e}} - \bar{x}_{\bar{e}}) = \varepsilon(\bar{d}^y_{\bar{e}} - \bar{d}^x_{\bar{e}}) = (\bar{y}^+_{\bar{f}} - \bar{y}_{\bar{f}}) - (\bar{x}^+_{\bar{f}} - \bar{x}_{\bar{f}}) = \bar{b}^+_{\bar{f}} - \bar{b}_{\bar{f}}.$$

Therefore, $\bar{b}^+_{\bar{e}} = \bar{b}^+_{\bar{f}}$. By (A.9), choosing a small enough ε ensures the nonnegativity of $\bar{p}^+$. Since $\bar{p}^+$ does not belong to $P(G, c)$, we get that $\bar{p}^+$ violates $x(\bar{D}) - y(\bar{D}) \le c(\bar{D})$, that is,

$$\bar{b}^+_{\bar{e}} + \bar{b}^+_{\bar{f}} = \bar{b}_{\bar{e}} + \bar{b}_{\bar{f}} + 2\varepsilon(\bar{d}^y_{\bar{e}} - \bar{d}^x_{\bar{e}}) < 0, \qquad \forall \varepsilon > 0 \tag{A.10}$$

Notice that exactly one of $\bar{x}_{\bar{e}}$ and $\bar{y}_{\bar{e}}$ is fractional, as otherwise (A.9) would imply $\bar{d}^x_{\bar{e}} = \bar{d}^y_{\bar{e}} = 0$, and then (A.10) would give the contradiction $\bar{b}(\bar{D}) < 0$. Consequently, we have $\bar{b}_{\bar{e}} + \bar{b}_{\bar{f}} > 0$, a contradiction to the fact that (A.10) holds for all $\epsilon > 0$. This settles Case 1.

Case 2: $\bar{b}_{\bar{e}} = -\bar{b}_{\bar{f}} < 0$.

Define

$$\bar{d}^x_{\bar{f}} = \begin{cases} \bar{d}^y_{\bar{e}} - \bar{d}^x_{\bar{e}} & \text{if } \bar{x}_{\bar{f}} > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{d}^y_{\bar{f}} = \begin{cases} \bar{d}^x_{\bar{e}} - \bar{d}^y_{\bar{e}} & \text{if } \bar{y}_{\bar{f}} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

By definition of $\bar{d}$, and by (A.8)–(A.9), we have $\bar{b}^+_{\bar{e}} - \bar{b}_{\bar{e}} = \varepsilon(\bar{d}^y_{\bar{e}} - \bar{d}^x_{\bar{e}}) = (\bar{x}^+_{\bar{f}} - \bar{x}_{\bar{f}}) - (\bar{y}^+_{\bar{f}} - \bar{y}_{\bar{f}}) = \bar{b}_{\bar{f}} - \bar{b}^+_{\bar{f}}$. Therefore, $\bar{b}^+_{\bar{f}} = -\bar{b}^+_{\bar{e}}$.

In particular, $\bar{p}^+$ satisfies the constraint of the cut $\bar{D}$, and since nonnegativity is ensured, then $\bar{p}^+$ violates the constraint of a cut D containing $\bar{f}$ but not $\bar{e}$, that is

$$\bar{b}^+(D) = \bar{b}(D) + \varepsilon(\bar{d}^y(D) - \bar{d}^x(D)) < 0 \qquad (\forall \varepsilon > 0) \tag{A.11}$$

Since $D' = D \cup \{\bar{e}\} \setminus \{\bar{f}\}$ is a cut, we have $\bar{b}(D') \ge 0$, thus $\bar{b}(D) = \bar{b}(D') - \bar{b}_{\bar{e}} + \bar{b}_{\bar{f}} > 0$. This contradiction to (A.11) finishes the proof. □

References

[1] J. Edmonds, R. Giles, A min-max relation for submodular functions on graphs, Ann. Discrete Math. 1 (1977) 185–204, http://dx.doi.org/10.1016/S0167-5060(08)70734-9.

[2] W. Cook, On box totally dual integral polyhedra, Math. Program. 34 (1) (1986) 48–61, http://dx.doi.org/10.1007/BF01582162.

[3] A. Schrijver, Theory of Linear and Integer Programming, in: Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Inc., New York, NY, USA, 1986.

[4] A. Schrijver, Combinatorial Optimization Polyhedra and Efficiency, vol. 24, (January 2003) Springer, 2003, p. 1881.

[5] G. Ding, L. Feng, W. Zang, The complexity of recognizing linear systems with certain integrality properties, Math. Program. 114 (2) (2008) 321–334, http://dx.doi.org/10.1007/s10107-007-0103-y.

[6] J. Pap, Recognizing conic TDI systems is hard, Math. Program. 128 (1) (2011) 43–48, http://dx.doi.org/10.1007/s10107-009-0294-5.

[7] X. Chen, Z. Chen, W. Zang, A unified approach to Box-Mengerian hypergraphs, Math. Oper. Res. 35 (3) (2010) 655–668, http://dx.doi.org/10.1287/moor.1100.0458.

[8] X. Chen, G. Ding, W. Zang, A characterization of Box-Mengerian matroid ports, Math. Oper. Res. 33 (2) (2008) 497–512, http://dx.doi.org/10.1287/moor.1070.0306.

[9] X. Chen, G. Ding, W. Zang, The box-TDI system associated with 2-edge connected spanning subgraphs, Discrete Appl. Math. 157 (1) (2009) 118–125, http://dx.doi.org/10.1016/j.dam.2008.05.001.

[10] P.D. Seymour, The matroids with the max-flow min-cut property, J. Combin. Theory Ser. B 23 (2) (1977) 189–222, http://dx.doi.org/10.1016/0095-8956(77)90031-4.

[11] G. Ding, L. Tan, W. Zang, When is the matching polytope box-totally dual integral? Math. Oper. Res. 43 (1) (2018) 64–99, http://dx.doi.org/10.1287/moor.2017.0852.

[12] P.D. Seymour, Sums of circuits, Graph Theory Related Top. 1 (1979) 341–355.

[13] F. Barahona, A.R. Mahjoub, On the cut polytope, Math. Program. 36 (2) (1986) 157–173, http://dx.doi.org/10.1007/BF02592023.

[14] P.D. Seymour, Matroids and multicommodity flows, European J. Combin. 2 (3) (1981) 257–290, http://dx.doi.org/10.1016/S0195-6698(81)80033-9.

[15] J. Edmonds, E.L. Johnson, Matching, Euler tours and the Chinese postman, Math. Program. 5 (1) (1973) 88–124, http://dx.doi.org/10.1007/BF01580113.

[16] D. Cornaz, Max-multiflow/min-multicut for G+H series-parallel, Discrete Math. 311 (17) (2011) 1957–1967, http://dx.doi.org/10.1016/j.disc.2011.05.025.

[17] R.J. Duffin, Topology of series-parallel networks, J. Math. Anal. Appl. 10 (2) (1965) 303–318, http://dx.doi.org/10.1016/0022-247X(65)90125-3.

[18] A. Sebő, The Schrijver system of odd join polyhedra, Combinatorica 8 (1) (1988) 103–116, http://dx.doi.org/10.1007/BF02122558.

[19] S. Chopra, The Graph partitioning polytope on series-parallel and 4-wheel free graphs, SIAM J. Discrete Math. 7 (1) (1994) 16–31, http://dx.doi.org/10.1137/S0895480191199415.

[20] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (2) (1981) 169–197, http://dx.doi.org/10.1007/BF02579273.

[21] W.-K. Shih, S. Wu, Y.S. Kuo, Unifying maximum cut and minimum cut of a planar graph, IEEE Trans. Comput. 39 (5) (1990) 694–697, http://dx.doi.org/10.1109/12.53581.

Discrete Applied Mathematics 308 (2022) 162–167

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

The Schrijver system of the flow cone in series–parallel graphs

Michele Barbato [a,1], Roland Grappe [b,2], Mathieu Lacroix [b], Emiliano Lancini [b,*], Roberto Wolfler Calvo [b,c]

[a] *Università degli Studi di Milano, Dipartimento di Informatica, OptLab, Via Bramante 65, 26013, Crema (CR), Italy*
[b] *Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430, Villetaneuse, France*
[c] *Università di Cagliari, Dipartimento di Matematica e Informatica, Cagliari (CA), Italy*

A R T I C L E I N F O

Article history:
Received 12 November 2018
Received in revised form 14 January 2020
Accepted 25 March 2020
Available online 17 April 2020

Keywords:
Total dual integrality
Box-total dual integrality
Schrijver system
Hilbert basis
Flow cone
Multicuts
Series–parallel graphs

A B S T R A C T

We represent a *flow* of a graph $G = (V, E)$ as a couple (C, e) with C a circuit of G and e an edge of C, and its incidence vector is the $0/\pm 1$ vector $\chi^{C\setminus e} - \chi^{e}$. The *flow cone* of G is the cone generated by the flows of G and the unit vectors.

When G has no K_5-minor, this cone can be described by the system $x(M) \geq 0$ for all multicuts M of G. We prove that this system is box-totally dual integral if and only if G is series–parallel. Then, we refine this result to provide the Schrijver system describing the flow cone in series–parallel graphs.

This answers a question raised by Chervet et al., (2018).

1. Introduction

Totally dual integral systems were introduced in the late 70s and are strongly connected to min–max relations in combinatorial optimization [16]. A rational system of linear inequalities $Ax \leq b$ is *totally dual integral (TDI)* if the minimization problem in the linear programming duality:

$$\max\{cx : Ax \leq b\} = \min\{yb : y \geq \mathbf{0}, yA = c\}$$

admits an integer optimal solution for each integer vector c such that the maximum is finite. Such systems describe integer polyhedra when b is integer [13]. Schrijver [15] proved that every full-dimensional polyhedron is described by a unique minimal TDI system $Ax \leq b$ with A integer—its *Schrijver system* [6].

A stronger property is the box-total dual integrality, where a system $Ax \leq b$ is *box-totally dual integral (box-TDI)* if

$$Ax \leq b, \quad \ell \leq x \leq u$$

is TDI for all rational vectors ℓ and u (with possible infinite components). General properties of such systems can be found in Cook [5] and Chapter 22.4 of Schrijver [16]. Note that, although every rational polyhedron $\{x : Ax \leq b\}$ is described by a TDI system $\frac{1}{k}Ax \leq \frac{1}{k}b$, for some integer k, not every polyhedron is described by a box-TDI system. A polyhedron

* Corresponding author.
E-mail address: lancini@lipn.univ-paris13.fr (E. Lancini).

1 Michele Barbato participated in this work as a member of FCiências.ID (University of Lisbon) and was financially supported by Portuguese National Funding under Project PTDC/MAT-NAN/2196/2014.

2 Supported by ANR, France DISTANCIA (ANR-17-CE40-0015).

https://doi.org/10.1016/j.dam.2020.03.054
0166-218X/© 2020 Elsevier B.V. All rights reserved.

described by a box-TDI system is called a *box-TDI polyhedron*. As proved by Cook [5], every TDI system describing such a polyhedron is actually box-TDI.

In the last decade, several new box-TDI systems were exhibited. Chen, Ding, and Zang [1] characterized box-Mengerian matroid ports. In [2], they provided a box-TDI system describing the 2-edge-connected spanning subgraph polyhedron for series–parallel graphs. Ding, Tan, and Zang [10] characterized the graphs for which the TDI system of Cunningam and Marsh [9] describing the matching polytope is actually box-TDI. Ding, Zang, and Zhao [11] introduced new subclasses of box-perfect graphs. Cornaz, Grappe, and Lacroix [8] provided several box-TDI systems in series–parallel graphs. Recently, Chervet, Grappe, and Robert [3] gave new geometric characterizations of box-TDI polyhedra.

As mentioned by Pulleyblank [14], it is not uncommon that the minimal integer system and the Schrijver system of a polyhedron coincide. This is the case of the matching polytope and matroid polyhedra. However, this does not hold in general, as shown by Cook [4] and Pulleyblank [14] for the b-matching polyhedron, and by Sebő [18] for the T-join polyhedron.

In this paper, we are interested in TDI, box-TDI, and Schrijver systems for the flow cone of series–parallel graphs. Given a graph $G = (V, E)$, a *flow* of G is a couple (C, e) with C a circuit of G and e an edge of C. In a flow (C, e), the edge e represents a demand and $C \setminus e$ represents the path satisfying this demand. The *incidence vector* of a flow (C, e) is the $0/\pm 1$ vector $\chi^{C\setminus e} - \chi^e$. The *flow cone* of G is the cone generated by the flows of G and the unit vectors χ^e of $\mathbb{R}^E$.

The *cut* $\delta(W)$ is the set of edges having exactly one endpoint in a subset W of V. A *bond* is an inclusionwise minimal nonempty cut. Note that a nonempty cut is the disjoint union of bonds. Given a partition $\{V_1, \ldots, V_k\}$ of V, the set of edges having endpoints in two distinct V_i's is called *multicut* and is denoted by $\delta(V_1, \ldots, V_k)$. The *cut cone* of G is the cone generated by the incidence vectors of the cuts of G. Equivalently, it is the cone generated by the incidence vectors of the bonds of G, or by those of the multicuts of G.

When G has no K_5-minor, the flow cone of G is the polar of the cut cone and is described by $x(C) \geq 0$, for all cuts C of G [19]. Chervet, Grappe, and Robert [3] proved that the flow cone is a box-TDI polyhedron if and only if the graph is series–parallel. Moreover they provided the following box-TDI system:

$$\frac{1}{2}x(B) \geq 0 \quad \text{for all bonds } B \text{ of } G. \tag{1}$$

Quoting them, they "*leave open the question of finding a box-TDI system with integer coefficients, which exists by [16, Theorem 22.6(i)] and [5, Corollary 2.5]*".

Contribution. The goal of this paper is to answer the question of [3] mentioned above. Throughout, the main concept that we use is that of Hilbert basis, whose definition and connection with TDIness are given at the end of the introduction.

We first prove that

$$x(M) \geq 0 \quad \text{for all multicuts } M \text{ of } G, \tag{2}$$

is a TDI system describing the flow cone if and only if the graph is series–parallel. As the flow cone is a box-TDI polyhedron for such graphs, this implies that System (2) is a box-TDI system if and only if the graph is series–parallel. We then refine this result by providing the corresponding Schrijver system, which is composed of the so-called chordal multicuts—see Corollary 3.4.

This completely answers the question of [3].

Outline. In the next paragraph, we provide definitions and notation. In Section 2, we first characterize the graphs for which multicuts form a Hilbert basis. It follows that System (2) is box-TDI precisely for series–parallel graphs. In Section 3, we provide a minimal integer Hilbert basis for multicuts in series–parallel graphs. This gives the Schrijver system for the flow cone in series–parallel graphs.

Definitions. Given a finite set S and a subset T of S, we denote by $\chi^T \in \{0, 1\}^S$ the incidence vector of T, that is χ^T_s equals 1 if s belongs to T and 0 otherwise, for all $s \in S$. Since there is a bijection between sets and their incidence vectors, we will often use the same terminology for both.

Let $G = (V, E)$ be a loopless undirected graph. Given $U \subseteq V$, the graph $G[U]$ is obtained from G by removing all the vertices not in U. A set of edges M is a multicut if and only if $|M \cap C| \neq 1$ for all circuits C of G—see e.g. [7]. The *reduced graph* of a multicut M is the graph G_M obtained by contracting all the edges of $E \setminus M$. Note that a multicut of G_M is also a multicut of G. We denote respectively by $\mathcal{M}_G$ and $\mathcal{B}_G$ the set of multicuts and the set of bonds of G. A subset of edges of G is called a *circuit* if it induces a connected graph in which every vertex has degree 2. Given a circuit C, an edge of G is a *chord* of C if its endpoints are two nonadjacent vertices of C. A graph is 2-connected if it remains connected whenever a vertex is removed.

A graph is *series–parallel* if its 2-connected components either consist of a single edge or can be constructed from the circuit of length two C_2 by repeatedly adding edges parallel to an existing one, and subdividing edges, that is, replacing an edge by a path of length two. Series–parallel graphs are those having no K_4-minor [12]. A graph is *chordal* if every circuit of length 4 or more has a chord.

The cone $\mathcal{C}$ *generated* by a set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ of $\mathbb{R}^n$ is the set of nonnegative combinations of $\mathbf{v}_1, \ldots, \mathbf{v}_k$, that is, $\mathcal{C} = \left\{\sum_{j=1}^k \lambda_j \mathbf{v}_j : \lambda_1, \ldots, \lambda_k \geq 0\right\}$. A set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is a *Hilbert basis* if each integer vector in their cone can

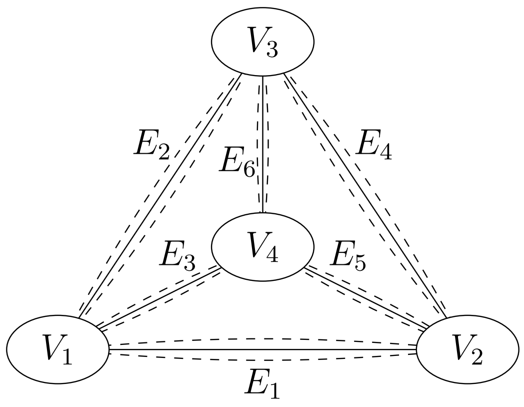

Fig. 1. Edges in the figure represent sets of edges of G having endpoints in distinct V_i's. Solid lines depict $e_1, \dots, e_6$ given in the proof of Theorem 2.1.

be expressed as a nonnegative integer combination of $\mathbf{v}_1, \dots, \mathbf{v}_k$. A Hilbert basis is *integer* if it is composed of integer vectors, and it is a *minimal integer Hilbert basis* if it has the smallest number of vectors among all integer Hilbert basis generating the same cone. Each pointed rational cone has a unique minimal integer Hilbert basis [15, Theorems 16.4]. The link between Hilbert basis and TDIness is in the following result.

Theorem 1.1 (*Corollary 22.5a of [16]*)**.** *A system $Ax \geq \mathbf{0}$ is TDI if and only if the rows of A form a Hilbert basis.*

2. When do multicuts form a Hilbert basis?

2.1. Characterization

The following result characterizes the graphs for which the multicuts form a Hilbert basis.

Theorem 2.1. *The multicuts of a graph form a Hilbert basis if and only if the graph is series–parallel.*

Proof. First, let us show that the incidence vectors of the multicuts of a non series–parallel graph do not form a Hilbert basis. Suppose that $G = (V, E)$ has K_4 as a minor. Without loss of generality, we may assume G connected. Then, V can be partitioned into four sets $\{V_1, \dots, V_4\}$ such that V_i induces a connected subgraph and at least one edge connects each pair V_i, V_j for $i, j = 1, \dots, 4$. We subdivide $\delta(V_1, V_2, V_3, V_4)$ into $E_1, \dots, E_6$ as in Fig. 1.

Let $\hat{E} = \{e_1, \dots, e_6\}$ where $e_i \in E_i$ for all $i = 1, \dots, 6$, and let $\mathbf{w} \in \mathbb{Z}^E$ be as follows:

$$\mathbf{w}_e = \begin{cases} 2 & \text{if } e \in E_1, \\ 1 & \text{if } e \in E_2, \dots, E_6, \\ 0 & \text{otherwise.} \end{cases}$$

Since $\mathbf{w} = \frac{1}{2}\chi^{\delta(V_1)} + \frac{1}{2}\chi^{\delta(V_2)} + \frac{1}{2}\chi^{\delta(V_1 \cup V_3)} + \frac{1}{2}\chi^{\delta(V_1 \cup V_4)}$, it belongs to the cut cone of G. Moreover, $\mathbf{w}^\top \chi^{\hat{E}} = 7$. Any conic combination of multicuts yielding $\mathbf{w}$ involves only multicuts contained in $\delta(V_1, \dots, V_4)$. Each of these multicuts contains between 3 and 6 edges of $\hat{E}$. Hence, if $\mathbf{w}$ is an integer combination of such multicuts, it is the sum of two multicuts containing 3 and 4 edges of $\hat{E}$, respectively. This means that $\mathbf{w}$ is the sum of $\chi^{\delta(V_i)}$ and $\chi^{\delta(V_i, V_j)}$ for some $i \neq j$. Since $\mathbf{w}_{e_1} = 2$, we have $i \in \{1, 2\}$ and $j \in \{3, 4\}$. But then $\delta(V_i) \cap \delta(V_i, V_j)$ contains an edge among $e_2, \dots, e_5$, a contradiction with $\mathbf{w}_{e_2} = \mathbf{w}_{e_3} = \mathbf{w}_{e_4} = \mathbf{w}_{e_5} = 1$.

Therefore, $\mathbf{w}$ is not an integer combination of multicuts, implying that the set of multicuts of G is not a Hilbert basis.

For the other direction, remark that each multicut of a series–parallel graph is the disjoint union of multicuts of its 2-connected components. Since they belong to disjoint spaces, if the set of multicuts of each 2-connected component forms a Hilbert basis, then so does their union. Hence, it is enough to prove that the multicuts of a 2-connected series–parallel graph form a Hilbert basis. From now on, assume the graph to be 2-connected.

We prove the result by induction on the number of edges of G. When $G = (\{u, v\}, \{e, f\})$ is the circuit of length two, the only nonempty multicut is $\{e, f\}$, and its incidence vector forms a Hilbert basis. Similarly, when G consists of a single edge, its incidence vector forms a Hilbert basis.

Now, let $\tilde{G} = (\tilde{V}, \tilde{E})$ be obtained from a 2-connected series–parallel graph $G = (V, E)$ by either adding a parallel edge or subdividing an edge. By the induction hypothesis, $\mathcal{M}_G$ is a Hilbert basis.

Suppose first that $\tilde{G}$ is obtained from G by adding an edge f parallel to an edge e of E. A subset of edges M of G containing (respectively not containing) e is a multicut if and only if $M \cup f$ (respectively M) is a multicut of $\tilde{G}$. Thus, the incidence vector of each multicut of $\tilde{G}$ is obtained by copying the component associated with e in the component of f. Since the incidence vectors of the multicuts of G are a Hilbert basis, so are the incidence vectors of the multicuts of $\tilde{G}$.

Suppose now that $\tilde{G}$ is obtained from G by subdividing an edge $\bar{e} \in E$. We denote by u the new vertex and by f and g the edges adjacent to it. A multicut M of $\tilde{G}$ can be expressed as the half-sum of the bonds of $\tilde{G}$. Moreover, as each bond is a multicut, bonds and multicuts of $\tilde{G}$ generate the same cone: the cut cone. Since System (1) is TDI in series–parallel graphs [3, end of Section 6.4], the set of vectors $\{\frac{1}{2}\chi^B : B \in \mathcal{B}_{\tilde{G}}\}$ forms a Hilbert basis.

Let $\mathbf{v}$ be an integer vector in the cut cone. There exist $\lambda_B \in \frac{1}{2}\mathbb{Z}_+$ for all $B \in \mathcal{B}_{\tilde{G}}$ such that $\mathbf{v} = \sum_{B\in\mathcal{B}_{\tilde{G}}} \lambda_B \chi^B$. The vector $\mathbf{v}$ is an integer combination of multicuts of $\tilde{G}$ if and only if $\mathbf{v} - \lfloor \lambda_{\delta(u)} \rfloor \chi^{\delta(u)}$ is, thus we may assume that $\lambda_{\delta(u)} \in \{0, \frac{1}{2}\}$. Define $\mathbf{w} \in \mathbb{Z}^E$ by:

$$\mathbf{w}_e = \begin{cases} \mathbf{v}_f + \mathbf{v}_g - 2\lambda_{\delta(u)} & \text{if } e = \bar{e}, \\ \mathbf{v}_e & \text{otherwise.} \end{cases}$$

Remark that $(B \setminus \bar{e}) \cup f$ and $(B \setminus \bar{e}) \cup g$ are bonds of $\tilde{G}$ whenever B is a bond of G containing $\bar{e}$. Moreover, a bond B of G which does not contain $\bar{e}$ is a bond of $\tilde{G}$. Since $\delta(u)$ is the unique bond of $\tilde{G}$ containing both f and g, we have:

$$\mathbf{w} = \sum_{B\in\mathcal{B}_G:\bar{e}\in B} (\lambda_{(B\setminus\bar{e})\cup f} + \lambda_{(B\setminus\bar{e})\cup g})\chi^B + \sum_{B\in\mathcal{B}_G:\bar{e}\notin B} \lambda_B \chi^B.$$

Thus, $\mathbf{w}$ belongs to the cut cone of G. Moreover, as $\lambda_{\delta(u)}$ is half-integer, $\mathbf{w}$ is integer. By the induction hypothesis, $\mathcal{M}_G$ is a Hilbert basis, hence there exist $\mu_M \in \mathbb{Z}_+$ for all $M \in \mathcal{M}_G$ such that $\mathbf{w} = \sum_{M\in\mathcal{M}_G} \mu_M \chi^M$. Consider the family $\mathcal{N}$ of multicuts of G where each multicut M of G appears μ_M times.

Suppose first that $\lambda_{\delta(u)} = 0$. Then, $\mathbf{v}_f + \mathbf{v}_g$ multicuts of $\mathcal{N}$ contain $\bar{e}$. Let $\mathcal{P}$ be a family of $\mathbf{v}_f$ multicuts of $\mathcal{N}$ containing $\bar{e}$ and $\mathcal{Q} = \{M \in \mathcal{N} : \bar{e} \in M\} \setminus \mathcal{P}$. Then, we have

$$\mathbf{v} = \sum_{M\in\mathcal{N}:\bar{e}\notin M} \chi^M + \sum_{M\in\mathcal{P}} \chi^{(M\setminus\bar{e})\cup f} + \sum_{M\in\mathcal{Q}} \chi^{(M\setminus\bar{e})\cup g},$$

hence $\mathbf{v}$ is a nonnegative integer combination of multicuts of $\tilde{G}$.

Suppose now that $\lambda_{\delta(u)} = \frac{1}{2}$. Then, $\mathbf{v}_f + \mathbf{v}_g - 1$ multicuts of $\mathcal{N}$ contain $\bar{e}$. Let $\mathcal{P}$ be a family of $\mathbf{v}_f - 1$ multicuts of $\mathcal{N}$ containing $\bar{e}$, let $\mathcal{Q}$ be a family of $\mathbf{v}_g - 1$ multicuts in $\{M \in \mathcal{N} : \bar{e} \in M\} \setminus \mathcal{P}$, and denote by N the unique multicut of $\mathcal{N}$ containing $\bar{e}$ which is not in $\mathcal{P} \cup \mathcal{Q}$. Then, we have

$$\mathbf{v} = \sum_{M\in\mathcal{N}:\bar{e}\notin M} \chi^M + \sum_{M\in\mathcal{P}} \chi^{(M\setminus\bar{e})\cup f} + \sum_{M\in\mathcal{Q}} \chi^{(M\setminus\bar{e})\cup g} + \chi^{N\setminus\bar{e}\cup\{f,g\}}.$$

Hence $\mathbf{v}$ is a nonnegative integer combination of multicuts of $\tilde{G}$. This proves that $\mathcal{M}_{\tilde{G}}$ is a Hilbert basis. □

2.2. An integer box-TDI system for the flow cone in series–parallel graphs

Combining the box-TDIness of the flow cone and Theorems 1.1 and 2.1 yields a box-TDI system for the flow cone of a series–parallel graph with only integer coefficients. This provides a first answer to the question of [3].

Corollary 2.2. *The following statements are equivalent:*

i. G is a series–parallel graph,
ii. System (2) is TDI,
iii. System (2) is box-TDI.

Proof (*i.*⇔*ii.*)**.** This equivalence follows by combining Theorems 1.1 and 2.1.

(*ii.*⇔*iii.*) If G is series–parallel, then System (1) is box-TDI [3, end of Section 6.4]. Hence, the flow cone of G is box-TDI. Since a TDI system describing a box-TDI polyhedron is a box-TDI system [5], point *ii.* implies point *iii.*. A box-TDI system being TDI by definition, point *iii.* implies point *ii.*. □

3. Which multicuts form Hilbert basis?

3.1. A minimal integer Hilbert basis

Theorem 2.1 provides the set of graphs whose multicuts form a Hilbert basis. The following theorem refines this result by characterizing the multicuts which form the minimal Hilbert basis.

A multicut is *chordal* when its reduced graph is 2-connected and chordal. Note that bonds are chordal multicuts.

Theorem 3.1. *The chordal multicuts of a series–parallel graph form a minimal integer Hilbert basis.*

Proof. Let $G = (V, E)$ be a series–parallel graph. By Theorem 2.1, the multicuts of G form an integer Hilbert basis. Hence, the minimal integer Hilbert basis is composed of the multicuts which are not disjoint union of other multicuts. These multicuts are characterized in the following lemma, from which stems the desired theorem.

Lemma 3.2. *A multicut of a series–parallel graph G is chordal if and only if it cannot be expressed as the disjoint union of other nonempty multicuts.*

Proof. Let M be a multicut of G. Recall that every multicut of G_M is a multicut of G. Besides, since the disjoint union of multicuts is a multicut, a disjoint union of nonempty multicuts is actually the disjoint union of two nonempty multicuts.

We first prove that, if G_M is 2-connected and chordal, then M is not the disjoint union of two nonempty multicuts. By contradiction, suppose that G_M is 2-connected and chordal, and $M = M_1 \cup M_2$ where M_1, M_2 are disjoint multicuts of G_M. If C is a circuit of length at most three in G_M, then $C \subseteq M_i$ for some $i = 1, 2$. Indeed, the edges of C are partitioned by M_1 and M_2, and a multicut and a circuit intersect in either none or at least two edges.

Since G_M is 2-connected and M_i is nonempty for $i = 1, 2$, there exists at least a circuit containing edges of both M_1 and M_2. Let C be such a circuit, of smallest length. Then, C has length at least 4, as otherwise it would be contained in one of M_1 and M_2. Since G_M is chordal, there exists a chord c of C. Denote by P_1 and P_2 the two paths of C between the endpoints of c. For $i = 1, 2$, the circuit $P_i \cup \{c\}$ is strictly shorter than C. Since C is the shortest circuit intersecting both M_1 and M_2, we get that $P_i \cup \{c\} \subseteq M_i$ for $i = 1, 2$. But then $c \in M_1 \cap M_2$, a contradiction.

To prove the other direction, first suppose that G_M is not 2-connected. Then, the set of edges of each 2-connected component of G_M is a multicut of G, and M is the disjoint union of these multicuts. Now, suppose that G_M is not chordal, that is, G_M contains a chordless circuit C of length at least 4. We will apply the following.

Claim 3.3. *Let C be a circuit of length at least 4 in a series–parallel graph G. Then, there exists a pair of vertices nonadjacent in $G[V(C)]$ whose removal disconnects G.*

Proof. We can assume that there are two nonadjacent vertices u and v of $G[V(C)]$ such that there exists a path P between u and v that has no internal vertex in C. Indeed, otherwise, removing any two nonadjacent vertices of $G[V(C)]$ would disconnect G.

Let us show that removing u and v disconnects G. Denote by Q and R the two paths of C between u and v. By contradiction, suppose that $G\setminus\{u, v\}$ is connected. Then, there exists a path containing neither u nor v between an internal vertex of R and an internal vertex of either P or Q. Let S be a minimal path of this kind. Then, no internal vertex of S belongs to P, Q, or R, and the subgraph composed of P, Q, R and S is a subdivision of K_4. This contradicts the hypothesis that G is series–parallel. □

By Claim 3.3 there exist two vertices u and v of C, nonadjacent in $G[V(C)]$, whose removal disconnects G. Denote by $V_1, \ldots, V_k$ the sets of vertices of the connected components of $G \setminus \{u, v\}$. Let $G_i = G[V_i \cup \{u, v\}]$ and denote by $E(G_i)$ the set of edges of G_i, for $i = 1, \ldots, k$. Note that, since u and v are not adjacent, $E(G_i) \cap E(G_j) = \emptyset$ for all distinct i and j. Thus, M is the disjoint union of $E(G_1), \ldots, E(G_k)$.

Let us prove that $E(G_i)$ is a multicut of G_M, for $i = 1, \ldots, k$. Consider a circuit D of G_M. If D is contained in one of the G_i's, then $|D \cap E(G_j)| \neq 1$ for $j = 1, \ldots, k$. Otherwise, D is the union of two paths from u to v, these paths being contained in two different G_i's. Without loss of generality, let these paths be $P_1 \in G_1$ and $P_2 \in G_2$. Then, we have $D \cap E(G_i) = P_i$ if $i = 1, 2$, and $\emptyset$ otherwise. Since u and v are not adjacent, the shortest path from u to v in each G_i is of length at least two, hence $|P_i| \geq 2$. Therefore $|D \cap E(G_i)| \neq 1$ for $i = 1, \ldots, k$.

Therefore, $E(G_i)$ is a multicut of G_M, and hence of G, for $i = 1, \ldots, k$. Hence, M is the disjoint union of multicuts of G. □

□

3.2. The Schrijver system of the flow cone in series–parallel graphs

Corollary 2.2 provides an integer box-TDI description of the flow cone in series–parallel graphs. However, this box-TDI description is not minimal: there are redundant inequalities whose removal preserves box-TDIness. Here, we provide the minimal integer box-TDI system for this cone. This completely answers the question of [3, end of Section 6.4].

Corollary 3.4. *The Schrijver system for the flow cone of a series–parallel graph G is the following:*

$$x(M) \geq 0 \quad \text{for all chordal multicuts } M \text{ of } G. \tag{3}$$

Moreover, this system is box-TDI.

Proof. By Theorems 1.1 and 3.1, System (3) is a minimal integer TDI system. Since every bond is a chordal multicut, this system describes the flow cone for series–parallel graphs. Therefore, by [5, Corollary 2.5] and by the flow cone being box-TDI for series–parallel graphs, System (3) is box-TDI. □

We mention that, by planar duality, Corollary 3.4 provides the Schrijver system for the cone of conservative functions [17, Corollary 29.2h] in series–parallel graphs.

Acknowledgments

We are indebted to the anonymous referees for their valuable comments which greatly helped to improve the presentation of the paper.

References

[1] X. Chen, G. Ding, W. Zang, A characterization of box-Mengerian matroid ports, Math. Oper. Res. 33 (2) (2008) 497–512.
[2] X. Chen, G. Ding, W. Zang, The box-TDI system associated with 2-edge connected spanning subgraphs, Discrete Appl. Math. 157 (1) (2009) 118–125.
[3] P. Chervet, R. Grappe, L.-H. Robert, Box-total dual integrality, box-integrality, and equimodular matrices., 2020, to appear in Math. Program..
[4] W. Cook, A minimal totally dual integral defining system for the b-matching polyhedron, SIAM J. Algebr. Discrete Methods 4 (2) (1983) 212–220.
[5] W. Cook, On box totally dual integral polyhedra, Math. Program. 34 (1) (1986) 48–61.
[6] W. Cook, W.R. Pulleyblank, Linear systems for constrained matching problems, Math. Oper. Res. 12 (1) (1987) 97–120.
[7] D. Cornaz, Max-multiflow/min-multicut for G+H series-parallel, Discrete Math. 311 (17) (2011) 1957–1967.
[8] D. Cornaz, R. Grappe, M. Lacroix, Trader multiflow and box-TDI systems in series–parallel graphs, Discrete Optim. 31 (2019) 103–114.
[9] W.H. Cunningham, A. Marsh, A primal algorithm for optimum matching, in: Polyhedral Combinatorics, Springer, 1978, pp. 50–72.
[10] G. Ding, L. Tan, W. Zang, When is the matching polytope box-totally dual integral? Math. Oper. Res. 43 (1) (2017) 64–99.
[11] G. Ding, W. Zang, Q. Zhao, On box-perfect graphs, J. Combin. Theory Ser. B 128 (2018) 17–46.
[12] R.J. Duffin, Topology of series-parallel networks, J. Math. Anal. Appl. 10 (2) (1965) 303–318.
[13] J. Edmonds, R. Giles, A min-max relation for submodular functions on graphs, in: Annals of Discrete Mathematics, Vol. 1, Elsevier, 1977, pp. 185–204.
[14] W.R. Pulleyblank, Total dual integrality and b-matchings, Oper. Res. Lett. 1 (1) (1981) 28–30.
[15] A. Schrijver, On total dual integrality, Linear Algebra Appl. 38 (1981) 27–32.
[16] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, 1998.
[17] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Vol. 24, Springer Science & Business Media, 2003.
[18] A. Sebő, The Schrijver system of odd join polyhedra, Combinatorica 8 (1) (1988) 103–116.
[19] P.D. Seymour, Sums of circuits, in: J.A. Bondy, U.S.R. Murty (Eds.), Graph Theory and Related Topics, Vol. 1, Academic Press, 1979, pp. 341–355.

Mathematical Programming (2023) 197:307–336
https://doi.org/10.1007/s10107-021-01743-x

FULL LENGTH PAPER

Series A

Box-total dual integrality and edge-connectivity

Michele Barbato[1] · **Roland Grappe**[2] · **Mathieu Lacroix**[2] · **Emiliano Lancini**[2,3]

Received: 15 May 2020 / Accepted: 19 November 2021 / Published online: 30 January 2022

Abstract

Given a graph $G = (V, E)$ and an integer $k \geq 1$, the graph $H = (V, F)$, where F is a family of elements (with repetitions allowed) of E, is a k-edge-connected spanning subgraph of G if H cannot be disconnected by deleting any $k - 1$ elements of F. The convex hull of incidence vectors of the k-edge-connected subgraphs of a graph G forms the k-edge-connected subgraph polyhedron of G. We prove that this polyhedron is box-totally dual integral if and only if G is series–parallel. In this case, we also provide an integer box-totally dual integral system describing this polyhedron.

Keywords Box-total dual integrality · k-edge connected subgraph · Polyhedron · Series–parallel graph

Mathematics Subject Classification 90C27 · 90C05 · 90C57

1 Introduction

Totally dual integral systems, introduced in the late 70's, are strongly connected to min–max relations in combinatorial optimization [34]. A rational system of linear

A preliminary version has been published in the proceedings of the conference ISCO 2020.

Emiliano Lancini
emiliano.lancini@eseo.fr

Michele Barbato
michele.barbato@unimi.it

Roland Grappe
grappe@lipn.univ-paris13.fr

Mathieu Lacroix
lacroix@lipn.univ-paris13.fr

1 Department of Computer Science, University of Milan, 20133 Milan, Italy

2 Université Sorbonne Paris Nord, LIPN, CNRS, UMR 7030, 93430 Villetaneuse, France

3 Eseo, 78140 Vélizy-Villacoublay, France

inequalities $Ax \geq b$ is *totally dual integral (TDI)* if the maximization problem in the linear programming duality

$$\min\{c^\top x : Ax \geq b\} = \max\{b^\top y : A^\top y = c, y \geq \mathbf{0}\}$$

admits an integer optimal solution for each integer vector c such that the optimum is finite. Every rational polyhedron can be described by a TDI system [28]. For instance, the polyhedron $\{x : Ax \geq b\}$ can be described by TDI systems of the form $\frac{1}{q}Ax \geq \frac{1}{q}b$ for certain positive q. However, a polyhedron is integer if and only if it can be described by a TDI system with only integer coefficients [23,28]. Integer TDI systems yield min–max results that may have combinatorial interpretation.

A stronger property is box-total dual integrality: a system $Ax \geq b$ is *box-totally dual integral (box-TDI)* if $Ax \geq b, \ell \leq x \leq u$ is TDI for all rational vectors ℓ and u (possibly with infinite components). General properties of such systems can be found in Cook [12] and Chapter 22.4 of Schrijver [34]. Note that, although every rational polyhedron can be described by a TDI system, not every polyhedron can be described by a box-TDI system. A polyhedron which can be described by a box-TDI system is called a *box-TDI polyhedron*. As proved by Cook [12], every TDI system describing such a polyhedron is actually box-TDI.

Recently, several new box-TDI systems have been exhibited. Chen et al. [6] characterized box-Mengerian matroid ports. Ding et al. [18] characterized the graphs for which the Edmonds' system defining the matching polytope [21] is box-TDI. Ding et al. [19] exhibited new subclasses of box-perfect graphs. Cornaz et al. [14] provided several box-TDI systems in series–parallel graphs. Barbato et al. [3] gave the minimal box-TDI system with integer coefficients for the flow cone for series–parallel graphs. For these graphs, Chen et al. [7] provided a box-TDI system describing the 2-edge-connected spanning subgraph polyhedron.

In this paper, we are interested in integrality properties of systems related to k-edge-connected spanning subgraphs. A *k-edge-connected spanning subgraph* of a graph $G = (V, E)$ is a graph $H = (V, F)$, with F being a collection of elements of E where each element can appear several times, that remains connected after the removal of any $k - 1$ edges.

These objects model a kind of failure resistance of telecommunication networks. More precisely, they represent networks which remain connected when $k - 1$ links fail. The underlying network design problem is the *k-edge-connected spanning subgraph problem* (*k-ECSSP*): given a graph G and positive edge costs, find a k-edge-connected spanning subgraph of G of minimum cost. Special cases of this problem are related to classical combinatorial optimization problems. The 2-ECSSP is a well-studied relaxation of the traveling salesman problem [24] and the 1-ECSSP is nothing but the well-known minimum spanning tree problem. While this latter is polynomial-time solvable, the k-ECSSP is **NP**-hard for every fixed $k \geq 2$ [27].

Different algorithms have been devised in order to deal with the k-ECSSP, such as branch-and-cut procedures [4,15], approximation algorithms [8,26], cutting plane algorithms [30], and heuristics [11]. In [36], Winter introduced a linear-time algorithm solving the 2-ECSSP on series–parallel graphs. Most of these algorithms rely on polyhedral considerations.

Given a graph $G = (V, E)$, the convex hull of incidence vectors of all the families of E inducing a k-edge-connected spanning subgraph of G forms a polyhedron, hereafter called the *k-edge-connected spanning subgraph polyhedron* of G and denoted by $P_k(G)$. Cornuéjols et al. [16] gave a system describing $P_2(G)$ when G is series–parallel. Vandenbussche and Nemhauser [35] characterized in terms of forbidden minors the graphs for which this system describes $P_2(G)$. Chopra [10] described $P_k(G)$ for outerplanar graphs when k is odd. Didi Biha and Mahjoub [17] extended these results to series–parallel graphs for all $k \geq 2$. By a result of Baïou et al. [1], the inequalities in these descriptions can be separated in polynomial time, which implies that the k-ECSSP is solvable in polynomial time for series–parallel graphs.

When studying k-edge-connected spanning subgraphs of a graph G, we can add the constraint that each edge of G can be taken at most once. We denote the corresponding polyhedron by $Q_k(G)$. Barahona and Mahjoub [2] described $Q_2(G)$ for Halin graphs. Further polyhedral results for the case $k = 2$ have been obtained by Boyd and Hao [5] and Mahjoub [32,33]. Grötschel and Monma [29] described several classes of facets of $Q_k(G)$. Moreover, Fonlupt and Mahjoub [25] extensively studied the extremal points of $Q_k(G)$ and characterized the class of graphs for which this polytope is described by cut inequalities and $\mathbf{0} \leq x \leq \mathbf{1}$.

The polyhedron $P_1(G)$ is known to be box-TDI for all graphs [31]. For series–parallel graphs, the system given in [16] describing $P_2(G)$ is not TDI. Chen et al. [7] showed that dividing it by 2 yields a TDI system for such graphs. Actually, they proved that this system is box-TDI if and only if the graph is series–parallel.

Contributions. Our starting point is the result of Chen et al. [7]. First, their result implies that $P_2(G)$ is a box-TDI polyhedron for series–parallel graphs. However, this leaves open the question of the box-TDIness of $P_2(G)$ for non series–parallel graphs. More generally, for which integers k and graphs G is $P_k(G)$ a box-TDI polyhedron?

We answer this question by proving that, for $k \geq 2$, $P_k(G)$ is a box-TDI polyhedron if and only if G is series–parallel. Note that this work is one of the first ones that proves the box-TDIness of a polyhedron without giving a box-TDI system describing it. Instead, our proof is based on the recent matricial characterization of box-TDI polyhedra given by Chervet et al. [9].

By [34, Theorem 22.6], there exists a TDI system with integer coefficients describing $P_k(G)$. For series–parallel graphs, the system provided by Chen et al. [7] has noninteger coefficients. Moreover, the system given by Didi Biha and Mahjoub [17] describing $P_k(G)$ when k is even is not TDI. When $k \geq 2$ and G is series–parallel, which combinatorial objects yield an integer TDI system describing $P_k(G)$?

We answer this question by exhibiting integer TDI systems based on multicuts. When k is even, we use multicuts to provide an integer TDI system for $P_k(G)$ when G is series–parallel. Our proof relies on the standard constructive characterization of series–parallel graphs. When k is odd, we prove that the description of $P_k(G)$ given by Didi Biha and Mahjoub [17] based on multicuts is TDI if and only if the graph is series–parallel. For this case, our proof relies on new properties of the set of degree 2 vertices in simple series–parallel graphs stated in Lemma 2.3.

The box-totally dual integral characterization of $P_k(G)$ implies that these systems are actually box-TDI if and only if G is series–parallel. By definition of box-TDIness, adding $x \leq \mathbf{1}$ to these systems yields box-TDI systems for $Q_k(G)$ for series–parallel graphs.

Outline. In Sect. 2, we give the definitions and preliminary results used throughout the paper. In Sect. 3, we prove that, for $k \geq 2$, $P_k(G)$ is a box-TDI polyhedron if and only if G is series–parallel. In Sect. 4, we provide a TDI system with integer coefficients describing $P_k(G)$ when G is series–parallel and $k \geq 2$ is even. In Sect. 5, we show the TDIness of the system given by Didi Biha and Mahjoub [17] that describes $P_k(G)$ for G series–parallel and $k \geq 3$ odd.

2 Definitions and preliminary results

This section is devoted to the definitions, notation, and preliminary results used throughout the paper.

2.1 Graphs and combinatorial objects

Given a set E, a *family* of E is a collection of elements of E where each element can appear multiple times. The incidence vector of a family F of E is the vector χ^F of $\mathbb{Z}_+^E$ such that e's coordinate is the multiplicity of e in F for all e in E. Since there is a bijection between families and their incidence vectors, we will often use the same terminology for both.

Given a graph $G = (V, E)$ and the incidence vector $z \in \mathbb{Z}_+^E$ of a family F of E, $G(z)$ denotes the graph (V, F).

Let $G = (V, E)$ be a loopless undirected graph. Two edges of G are *parallel* if they share the same endpoints, and G is *simple* if it does not have parallel edges. A graph is 2-*connected* if it cannot be disconnected by removing a vertex. The graph obtained from two disjoint graphs by identifying two vertices, one of each graph, is called a *1-sum*. A 2-connected graph is *trivial* if it is composed of a single edge. We denote by K_n the complete graph on n vertices, that is the simple graph with n vertices and one edge between each pair of vertices. Given an edge e of G, we denote by $G\backslash e$ (respectively G/e) the graph obtained by removing (respectively contracting) the edge e, where *contracting* an edge uv consists in removing it and identifying u and v. Similarly, we denote by $G\backslash v$ the graph obtained from G by removing the vertex v, and by $G[W]$ the *graph induced by* W, that is, the graph obtained by removing all vertices not in the vertex subset W. Given a vector $x \in \mathbb{R}^E$ and a subgraph H of G, we denote by $x_{|H}$ the vector obtained by restricting x to the components associated with the edges of H.

A subset of edges of G is called a *circuit* if it induces a connected graph in which every vertex has degree 2. Given a subset U of V, the *cut* $\delta(U)$ is the set of edges having exactly one endpoint in U. A *bond* is a minimal nonempty cut. Given a partition $\{V_1, \ldots, V_n\}$ of V, the set of edges having endpoints in two distinct V_i's is called a *multicut* and is denoted by $\delta(V_1, \ldots, V_n)$. We denote respectively by $\mathcal{M}_G$ and $\mathcal{B}_G$ the set of multicuts and the set of bonds of G. For every multicut M, there exists a unique partition $\{V_1, \ldots, V_{d_M}\}$ of vertices of V such that $M = \delta(V_1, \ldots, V_{d_M})$, and $G[V_i]$ is connected for all $i = 1, \ldots, d_M$. We say that d_M is the *order* of M and $V_1, \ldots, V_{d_M}$ are the *classes* of M. Multicuts are characterized in terms of circuits, as stated in the following.

Lemma 2.1 [13] *A set of edges M is a multicut if and only if $|M \cap C| \neq 1$ for all circuits C of G.*

We denote the symmetric difference of two sets S and T by $S \triangle T$. It is well-known that the symmetric difference of two cuts is a cut. Moreover, the following result holds.

Observation 2.2 *Let G be a graph, v be a degree 2 vertex of G, and M be a multicut such that $|M \cap \delta(v)| = 1$. Then, $M \cup \delta(v)$ and $M \triangle \delta(v)$ are multicuts. Moreover, $d_{M \cup \delta(v)} = d_M + 1$, and $d_{M \triangle \delta(v)} = d_M$.*

A graph is *series–parallel* if its nontrivial 2-connected components can be constructed from a circuit of length 2 by repeatedly adding edges parallel to an existing one, and subdividing edges, that is, replacing an edge by a path of length two. Equivalently, series–parallel graphs are those having no K_4-minor [20].

By construction, simple nontrivial 2-connected series–parallel graphs have at least one degree 2 vertex. Moreover, these vertices satisfy the following.

Lemma 2.3 *For a simple nontrivial 2-connected series–parallel graph, at least one of the following holds:*

(i) *Two degree 2 vertices are adjacent,*
(ii) *A degree 2 vertex belongs to a circuit of length 3,*
(iii) *Two degree 2 vertices belong to the same circuit of length 4.*

Proof We proceed by induction, the base case is K_3 for which (i) holds.

Let G be a simple 2-connected series–parallel graph. Since G is simple, it can be built from a series–parallel graph H by subdividing an edge e into a path f, g. Let v be the degree 2 vertex added with this operation. By the induction hypothesis, either H is not simple, or one among (i), (ii), and (iii) holds for H. Hence, there are four cases.

Case 1 H is not simple. Since G is simple, e is parallel to exactly one edge h. Hence, f, g, h is a circuit of G length 3 containing v, thus (ii) holds for G.

Case 2 (i) Holds for H. Then, it holds for G.

Case 3 (ii) Holds for H. Let C be a circuit of H of length 3 containing a degree 2 vertex, say w. If $e \notin C$, then (ii) holds for G. Otherwise, by subdividing e, we obtain a circuit of length 4 containing v and w, and hence (iii) holds for G.

Case 4 (iii) Holds for H. Let C be a circuit of H of length 4 containing two degree 2 vertices. If $e \notin C$, then (iii) holds for G. Otherwise, by subdividing e, we obtain a circuit of length 5 containing three degree 2 vertices. Then, at least two of them are adjacent, and so (i) holds for G. □

2.2 Box-total dual integrality

Let $A \in \mathbb{R}^{m\times n}$ be a full-row rank matrix. This matrix is *equimodular* if all its $m \times m$ non-zero determinants have the same absolute value. The matrix A is *face-defining for* a face F of a polyhedron $P \subseteq \mathbb{R}^n$ if $\text{aff}(F) = \{x \in \mathbb{R}^n : Ax = b\}$ for some $b \in \mathbb{R}^m$, where $\text{aff}(F)$ denotes the affine hull of F. Such matrices are the *face-defining matrices of P*.

Theorem 2.4 [9, Theorem 1.4] *Let P be a polyhedron. Then, the following statements are equivalent:*

(i) *P is box-TDI.*
(ii) *Every face-defining matrix of P is equimodular.*
(iii) *Each face of P has an equimodular face-defining matrix.*

In Theorem 2.4, the equivalence of conditions (ii) and (iii) follows from the following observation.

Observation 2.5 ([9, Observation 4.10]) *Let F be a face of a polyhedron. If a face-defining matrix for F is equimodular, then so are all the face-defining matrices for F.*

We will also use the following.

Observation 2.6 *Let $A \in \mathbb{R}^{I\times J}$ be a full row rank matrix and $j \in J$. If A is equimodular, then so are following two matrices:*

(i) $\begin{bmatrix} A \\ \pm\chi^j \end{bmatrix}$ *if it is full row-rank,*
(ii) $\begin{bmatrix} A & \mathbf{0} \\ \pm\chi^j & \pm 1 \end{bmatrix}$.

Observation 2.7 [9, Observation 4.11] *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let $F = \{x \in P : Bx = b\}$ be a face of P. If B has full-row rank and $n - \dim(F)$ rows, then B is face-defining for F.*

2.3 The *k*-edge-connected spanning subgraph polyhedron

Note that $P_k(G)$ is the dominant of the convex hull of incidence vectors of all the families of E containing at most k copies of each edge and inducing a k-edge-connected spanning subgraph of G. Since the dominant of a polyhedron is a polyhedron, $P_k(G)$ is a full-dimensional polyhedron even though it is the convex hull of an infinite number of points.

From now on, we assume that $k \geq 2$. Didi Biha and Mahjoub [17] gave a complete description of $P_k(G)$ for all k, when G is series–parallel.

Theorem 2.8 ([17]) *Let G be a series–parallel graph and h be a positive integer. Then, $P_{2h}(G)$ is described by:*

$$(1)\begin{cases} x(D) \geq 2h \quad \text{for all cuts } D \text{ of } G, & \text{(1a)} \\ x \geq \mathbf{0}, & \text{(1b)} \end{cases}$$

and $P_{2h+1}(G)$ is described by:

$$(2)\begin{cases} x(M) \geq (h+1)d_M - 1 \quad \textit{for all multicuts } M \textit{ of } G, & \text{(2a)} \\ x \geq \mathbf{0}. & \text{(2b)} \end{cases}$$

Since the incidence vector of a multicut $\delta(V_1, \ldots, V_\ell)$ of order ℓ is the half-sum of the incidence vectors of the bonds $\delta(V_1), \ldots, \delta(V_\ell)$, we can deduce another description of $P_{2h}(G)$.

Corollary 2.9 *Let G be a series–parallel graph and h be a positive integer. Then, $P_{2h}(G)$ is described by:*

$$(3)\begin{cases} x(M) \geq hd_M \quad \textit{for all multicuts } M \textit{ of } G, & \text{(3a)} \\ x \geq \mathbf{0}. & \text{(3b)} \end{cases}$$

We call constraints (2a) and (3a) *partition constraints*. A multicut M is *tight for a point* of $P_k(G)$ if this point satisfies with equality the partition constraint (2a) (respectively (3a)) associated with M when k is odd (respectively even). Moreover, M is *tight for a face F* of $P_k(G)$ if it is tight for all the points of F.

The following results give some insights on the structure of tight multicuts.

Theorem 2.10 [17, Theorem 2.3 and Lemma 3.1] *Let x be a point of $P_{2h+1}(G)$, and let $M = \delta(V_1, \ldots, V_{d_M})$ be a multicut tight for x. Then, the following hold:*

(i) *If $d_M \geq 3$, then $x\left(\delta(V_i) \cap \delta(V_j)\right) \leq h+1$ for all $i \neq j \in \{1, \ldots, d_M\}$.*
(ii) *$G \backslash V_i$ is connected for all $i = 1, \ldots, d_M$.*

Lemma 2.11 *Let v be a degree 2 vertex of G and M be a multicut of G strictly containing $\delta(v) = \{uv, vw\}$. If M is tight for a point of $P_k(G)$ with $k \geq 2$, then both $M \backslash uv$ and $M \backslash vw$ are multicuts of G of order $d_M - 1$.*

Proof It suffices to show that u and w belong to different classes of the multicut $M = \delta(v, V_2, \ldots, V_{d_M})$. Suppose that $u, w \in V_2$. Then M is the union of the two multicuts $\delta(v)$ and $M' = \delta(v \cup V_2, \ldots, V_{d_M})$. Since $d_{\delta(v)} + d_{M'} = d_M + 1$, the sum of the partition inequalities associated with $\delta(v)$ and M' implies that the partition inequality associated with M is tight for no point of $P_k(G)$ for every $k \geq 2$. □

Chopra [10] gave sufficient conditions for an inequality to be facet-defining for $P_k(G)$. The following proposition is a direct consequence of Theorems 2.4 and 2.6 of [10].

Lemma 2.12 *Let G be a connected graph having a K_4-minor. Then, there exist two disjoint nonempty subsets of edges of G, E' and E'', and a rational b such that*

$$x(E') + 2x(E'') \geq b, \tag{4}$$

is a facet-defining inequality of $P_{2h+1}(G)$.

Chen et al. [7] provided a box-TDI system for $P_2(G)$ for series–parallel graphs.

Theorem 2.13 [7, Theorem 1.1] *The system:*

$$\begin{cases} \frac{1}{2}x(D) \geq 1 \ \textit{for all cuts } D \textit{ of } G, \\ x \geq \mathbf{0} \end{cases} \tag{5}$$

is box-TDI if and only if G is a series–parallel graph.

This result proves that the polyhedron $P_2(G)$ is box-TDI for all series–parallel graphs, and gives a TDI system describing this polyhedron in this case. However, Theorem 2.13 is not sufficient to state that $P_2(G)$ is a box-TDI polyhedron if and only if G is series–parallel.

3 Box-TDIness of $P_k(G)$

In this section we show that, for $k \geq 2$, $P_k(G)$ is a box-TDI polyhedron for a connected graph G if and only if G is series–parallel. Since $P_k(G) = \emptyset$ when G is not connected, we assume from now on that G is connected.

When $k \geq 2$, $P_k(G)$ is not always box-TDI, as stated in Lemma 3.1. Indeed, by Theorem 2.4, if a polyhedron has a nonequimodular face-defining matrix, then it is not box-TDI. The proof of Lemma 3.1 exhibits such a matrix when G has a K_4-minor. This follows from the existence of a particular facet-defining inequality when k is odd, as shown by Chopra [10]. When k is even, we build a nonequimodular face-defining matrix based on the structure of cuts in a K_4-minor.

Lemma 3.1 *For $k \geq 2$, if $G = (V, E)$ has a K_4-minor, then $P_k(G)$ is not box-TDI.*

Proof When $k = 2h + 1$ is odd, Lemma 2.12 shows that there exists a facet-defining inequality that is described by a nonequimodular matrix as $P_k(G)$ is full-dimensional. Thus, $P_k(G)$ is not box-TDI by Statement (ii) of Theorem 2.4.

We now prove the case when k is even. Since G has a K_4-minor, there exists a partition $\{V_1, \ldots, V_4\}$ of V such that $G[V_i]$ is connected and $\delta(V_i, V_j) \neq \emptyset$ for all $i < j \in \{1, \ldots, 4\}$. We now prove that the matrix A whose three rows are $\chi^{\delta(V_i)}$ for $i = 1, 2, 3$ is a face-defining matrix of $P_k(G)$ which is not equimodular. This will end the proof by Statement (ii) of Theorem 2.4.

Let e_{ij} be an edge in $\delta(V_i, V_j)$ for all $i < j \in \{1, \ldots, 4\}$. The submatrix of A formed by the columns associated with edges e_{ij} is the following:

$$\begin{array}{c} \\ \chi^{\delta(V_1)} \\ \chi^{\delta(V_2)} \\ \chi^{\delta(V_3)} \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{13} & e_{23} & e_{14} & e_{24} & e_{34} \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

The matrix A is not equimodular as the first three columns form a matrix of determinant -2 whereas the last three ones give a matrix of determinant 1.

By Observation 2.7, to show that A is face-defining, it is enough to exhibit $|E|-2$ affinely independent points of $P_k(G)$ satisfying $x(\delta(V_i))=k$ for $i=1,2,3$.

Let $D_1=\{e_{12},e_{14},e_{23},e_{34}\}$, $D_2=\{e_{12},e_{13},e_{24},e_{34}\}$, $D_3=\{e_{13},e_{14},e_{23},e_{24}\}$ and $D_4=\{e_{14},e_{24},e_{34}\}$. First, we define the points $S_j=\sum_{i=1}^{4}k\chi^{E[V_i]}+\frac{k}{2}\chi^{D_j}$, for $j=1,2,3$, and $S_4=\sum_{i=1}^{4}k\chi^{E[V_i]}+k\chi^{D_4}$. Note that they are affinely independent.

Now, for each edge $e\notin\{e_{12},e_{13},e_{14},e_{23},e_{24},e_{34}\}$, we construct the point S_e as follows. When $e\in E[V_i]$ for some $i=1,\dots,4$, we define $S_e=S_4+\chi^e$. Adding the point S_e maintains affine independence as S_e is the only point not satisfying $x_e=k$. When $e\in\delta(V_i,V_j)$ for some i,j, we define $S_e=S_\ell-\chi^{e_{ij}}+\chi^e$, where S_ℓ is S_1 if $e\in\delta(V_1,V_4)\cup\delta(V_2,V_3)$ and S_2 otherwise. Affine independence comes because S_e is the only point involving e.

In total, we built $4+|E|-6=|E|-2$ affinely independent points. □

The following theorem characterizes the class of graphs for which $P_k(G)$ is box-TDI. The case k even is obtained using the box-TDIness for $k=2$ and the fact that integer dilations maintain box-TDIness. For the case k odd, on the contrary to what is generally done, the proof does not exhibit a box-TDI system describing $P_k(G)$. For this case, the proof is by induction on the number of edges of G. We prove that series–parallel operations preserve the box-TDIness of the polyhedron. The most technical part of the proof is the subdivision of an edge uw into two edges uv and vw. We proceed by contradiction: by Theorem 2.4, we suppose that there exists a face F of $P_k(G)$ defined by a nonequimodular matrix. We study the structure of the inequalities corresponding to this matrix. In particular, we show that they are all associated with multicuts, and that these multicuts contain either both uv and vw, or none of them—see Claims 3.1, 3.2, and 3.3. These last results allow us to build a nonequimodular face-defining matrix for the smaller graph, which contradicts the induction hypothesis.

Theorem 3.2 *For $k\geq 2$, $P_k(G)$ is a box-TDI polyhedron if and only if G is series–parallel.*

Proof Necessity follows from Lemma 3.1. Let us now prove sufficiency. When $k=2$, the box-TDIness of System (5) has been shown by Chen et al. [7]. This implies box-TDIness for all even k: multiplying the right-hand side of a box-TDI system by a positive rational preserves its box-TDIness [34, Section 22.5]. The system obtained by multiplying by $\frac{k}{2}$ the right-hand side of System (5) describes $P_k(G)$ when k is even. Hence, the latter is a box-TDI polyhedron.

The rest of the proof is devoted to the case where $k=2h+1$ for some $h\geq 1$. To this end, we prove that for every face of $P_{2h+1}(G)$ there exists an equimodular face-defining matrix. The characterization of box-TDIness given in Theorem 2.4 concludes. We proceed by induction on the number of edges of G.

If G is trivial, then $P_{2h+1}(G)=\{x\in\mathbb{R}_+ : x\geq 2h+1\}$ is box-TDI. If G is the circuit $\{e,f\}$, then $P_{2h+1}(G)=\{x_e,x_f\in\mathbb{R}_+ : x_e+x_f\geq 2h+1\}$ is also box-TDI.

(1-sum) Let G be the 1-sum of two series–parallel graphs $G'=(W',E')$ and $G''=(W'',E'')$. By induction, there exist two box-TDI systems $A'y\geq b'$ and $A''z\geq b''$ describing respectively $P_{2h+1}(G')$ and $P_{2h+1}(G'')$. If v is the vertex of G obtained by the identification, $G\backslash v$ is not connected, hence, by Statement (ii) of Theorem 2.10,

a multicut M of G is tight for a face of $P_{2h+1}(G)$ only if $M \subseteq E'$ or $M \subseteq E''$. It follows that for every face F of $P_{2h+1}(G)$ there exist faces F' and F'' of $P_{2h+1}(G')$ and $P_{2h+1}(G'')$ respectively, such that $F = F' \times F''$. Then $P_{2h+1}(G) = \{(y, z) \in \mathbb{R}_+^{E'} \times \mathbb{R}_+^{E''} : A'y \geq b', A''z \geq b''\}$ and so it is box-TDI.

(Parallelization) Let $G = (V, E)$ be obtained from a series–parallel graph G' by adding an edge g parallel to an edge f of G' and suppose that $P_{2h+1}(G')$ is box-TDI. Let $A'x \geq b$ be a box-TDI system describing $P_{2h+1}(G')$. Note that $P_{2h+1}(G)$ is described by $Ax \geq b, x_f \geq 0, x_g \geq 0$, where A is the matrix obtained by duplicating f's column. By Theorem 22.10 of [34], the system $Ax \geq b$ is box-TDI, hence so is $Ax \geq b, x_f \geq 0, x_g \geq 0$. Thus, $P_{2h+1}(G)$ is a box-TDI polyhedron.

(Subdivision) Let $G = (V, E)$ be obtained by subdividing an edge uw of a series–parallel graph $G' = (V', E')$ into a path of length two uv, vw. By contradiction, suppose there exists a nonempty face $F = \{x \in P_{2h+1}(G) : A_F x = b_F\}$ such that A_F is a face-defining matrix for F which is not equimodular. Take such a face with maximum dimension. Then, every submatrix of A_F which is face-defining for a face of $P_{2h+1}(G)$ is equimodular. We may assume that A_F is defined by the partition constraints (2a) associated with the set of multicuts $\mathcal{M}_F$ and the nonnegativity constraints associated with the set of edges $\mathcal{E}_F$.

Claim 3.1 $\mathcal{E}_F = \emptyset$.

Proof Suppose there exists an edge $e \in \mathcal{E}_F$. Let $H = G \backslash e$ and let $A_{F_H} x = b_{F_H}$ be the system obtained from $A_F x = b_F$ by removing the column and the nonnegativity constraint associated with e. Since the matrix A_F is of full row rank, so is A_{F_H}. Since $e \in \mathcal{E}_F$, for all multicuts M tight for F not containing e, $M \cup e$ is not a multicut. Hence $M \backslash e$ is a multicut of H of order d_M, for all M in $\mathcal{M}_F$. Hence, the set $F_H = \{x \in P_{2h+1}(H) : A_{F_H} x = b_{F_H}\}$ is a face of $P_{2h+1}(H)$. Moreover, deleting e's coordinate of $\operatorname{aff}(F)$ gives $\operatorname{aff}(F_H)$ so A_{F_H} is face-defining for F_H. By the induction hypothesis, A_{F_H} is equimodular. Since maximal invertible square submatrices of A_F are in bijection with those of A_{F_H} and have the same determinant in absolute value, A_F is equimodular, a contradiction. □

Claim 3.2 *For $e \in \{uv, vw\}$, at least one multicut of $\mathcal{M}_F$ different from $\delta(v)$ contains e.*

Proof By contradiction, suppose for instance that uv belongs to no multicut of $\mathcal{M}_F$ different from $\delta(v)$.

First, suppose that $\delta(v)$ does not belong to $\mathcal{M}_F$. Then, the column of A_F associated with uv is zero. Let A'_F be the matrix obtained from A_F by removing this column. Every multicut of G not containing uv is a multicut of G' (relabelling vw by uw), so the rows of A'_F are associated with multicuts of G'. Thus, $F' = \{x \in P_k(G') : A'_F x = b_F\}$ is a face of $P_{2h+1}(G')$. Removing uv's coordinate from the points of F gives a set of points of F' of affine dimension at least $\dim(F) - 1$. Since A'_F has the same rank as A_F and has one column fewer than A_F, then A'_F is face-defining for F' by Observation 2.7. By the induction hypothesis, A'_F is equimodular. Since adding a column of zeros preserves equimodularity, A_F is also equimodular.

Suppose now that $\delta(v)$ belongs to $\mathcal{M}_F$. Then, the column of A_F associated with uv has zeros in each row but $\chi^{\delta(v)}$. Let $A_F^\star x = b_F^\star$ be the system obtained from $A_F x = b_F$

by removing the equation associated with $\delta(v)$. Then $F^\star = \{x \in P_k(G) : A^\star_F x = b^\star_F\}$ is a face of $P_k(G)$ of dimension $\dim(F) + 1$. Indeed, it contains F and $z + \alpha\chi^{uv} \notin F$ for every point z of F and $\alpha > 0$. Hence, $A^\star_F$ is face-defining for $F^\star$. This matrix is equimodular by the maximality assumption on F, and so is A_F by Statement (ii) of Observation 2.6. □

Claim 3.3 $|M \cap \delta(v)| \neq 1$ *for every multicut* $M \in \mathcal{M}_F$.

Proof Suppose there exists a multicut M tight for F such that $|M \cap \delta(v)| = 1$. Without loss of generality, suppose that M contains uv but not vw. Then, $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} \geq x_{uv}\}$ because of the partition inequality (2a) associated with the multicut $M \triangle \delta(v)$. Moreover, the partition inequality associated with $\delta(v)$ and the integrality of $P_{2h+1}(G)$ imply $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} \geq h+1\}$. The proof is divided into two cases.

Case 1 $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} = h+1\}$. We prove this case by exhibiting an equimodular face-defining matrix for F. By Observation 2.5, this implies that A_F is equimodular, which contradicts the assumption on F.

Equality $x_{vw} = h+1$ can be expressed as a linear combination of equations of $A_F x = b_F$. Let $A'_F x = b'_F$ denote the system obtained by replacing an equation of $A_F x = b_F$ by $x_{vw} = h+1$ in such a way that the underlying affine space remains unchanged. Denote by $\mathcal{N}$ the set of multicuts of $\mathcal{M}_F$ containing vw but not uv. If $\mathcal{N} \neq \emptyset$, then let N be in $\mathcal{N}$. We now modify the system $A'_F x = b'_F$ by performing the following operations.

1. For all $M \in \mathcal{M}_F$ strictly containing $\delta(v)$, replace the equation associated with M by the partition constraint (2a) associated with $M \backslash vw$ set to equality, that is, $x(M\backslash vw) = (h+1)d_{M\backslash vw} - 1$.
2. If $\delta(v) \in \mathcal{M}_F$, then replace the equation associated with $\delta(v)$ by the constraint $x_{uv} = h$.
3. If $\mathcal{N} \neq \emptyset$, then replace the equation associated with N by the constraint $x_{uv} = h+1$.
4. For all $M \in \mathcal{N}\backslash N$, replace the equation associated with M by the partition constraint (2a) associated with $M \triangle \delta(v)$ set to equality, that is, $x(M \triangle \delta(v)) = (h+1)d_{M\triangle\delta(v)} - 1$.

These operations do not change the underlying affine space. Indeed, for every multicut M strictly containing $\delta(v)$ and tight for F, the set $M\backslash vw$ is a multicut tight for F by Lemma 2.11 and $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} = h+1\}$. If $\delta(v)$ is tight for F, then $F \subseteq \{x \in P_{2h+1}(G) : x_{uv} = h\}$ because $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} = h+1\}$. For $M \in \mathcal{N}$, by Observation 2.2, the set $M \triangle \delta(v)$ is a multicut of order d_M. The tightness of the constraint (2a) associated with N and the constraint (2a) associated with $M \triangle \delta(v)$ imply that $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} \leq x_{uv}\}$. Since $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} \geq x_{uv}\}$, we have $F \subseteq \{x \in P_{2h+1}(G) : x_{uv} = h+1\}$ and $M \triangle \delta(v)$ is tight for F. It follows that, if $\delta(v) \in \mathcal{M}_F$, then $\mathcal{N} = \emptyset$. Therefore, at most one among Operations 2 and 3 is applied so the rank of the matrix remains unchanged.

Let $A''_F x = b''_F$ be the system obtained by removing the equation $x_{vw} = h+1$ from $A'_F x = b'_F$. By construction, $A''_F x = b''_F$ is composed of constraints (2a) set to equality and possibly $x_{uv} = h$ or $x_{uv} = h+1$. Moreover, the column of A''_F associated with vw is zero. Let $F'' = \{x \in P_{2h+1}(G) : A''_F x = b''_F\}$. For every point z of F and $\alpha \geq 0$, $z + \alpha\chi^{vw}$ belongs to F'' because the column of A''_F associated with vw is zero, and $z + \alpha\chi^{vw} \in P_{2h+1}(G)$. This implies that $\dim(F'') \geq \dim(F) + 1$.

If F'' is a face of $P_{2h+1}(G)$, then A''_F is face-defining for F'' by Observation 2.7 and because A'_F is face-defining for F. By the maximality assumption on F, A''_F is equimodular, and hence so is A'_F by Statement (i) of Observation 2.6.

Otherwise, by construction, $F'' = F^\star \cap \{x \in \mathbb{R}^E : x_{uv} = t\}$ where $F^\star$ is a face of $P_{2h+1}(G)$ strictly containing F and $t \in \{h, h+1\}$. Therefore, there exists a face-defining matrix for F'' given by a face-defining matrix for $F^\star$ and the row χ^{uv}. Such a matrix is equimodular by the maximality assumption of F and Statement (i) of Observation 2.6. Hence, A''_F is equimodular by Observation 2.5, and so is A'_F by Statement (i) of Observation 2.6.

Case 2 $F \nsubseteq \{x \in P_{2h+1}(G) : x_{vw} = h+1\}$. Thus, there exists $z \in F$ such that $z_{vw} > h+1$. By Claim 3.2, there exists a multicut $N \neq \delta(v)$ containing vw which is tight for F. By Statement (i) of Theorem 2.10, the existence of z implies that N is a bond, hence it does not contain uv. The set $L = N \triangle \delta(v)$ is a bond of G. The partition inequality (2a) associated with L implies that $F \subseteq \{x \in P_{2h+1}(G) : x_{vw} = x_{uv}\}$ and L is tight for F. Moreover, N is the unique multicut tight for F containing vw. Suppose indeed that there exists a multicut B containing vw tight for F. Then, B is a bond by Statement (i) of Theorem 2.10 and the existence of z. Moreover, $B \triangle N$ is a multicut not containing vw. This implies that no point x of F satisfies the partition constraint associated with $B \triangle N$ because $x(B \triangle N) = x(B) + x(N) - 2x(B \cap N) = 2(2h+1) - 2x(B \cap N) \leq 4h + 2 - 2x_{vw} \leq 2h$, a contradiction.

Consider the matrix $A^\star_F$ obtained from A_F by removing the row associated with N. Matrix $A^\star_F$ is a face-defining matrix for a face $F^\star \supseteq F$ of $P_{2h+1}(G)$ because $F^\star$ contains F and $z + \alpha\chi^{uv}$ for every point z of F and $\alpha > 0$. By the maximality assumption, the matrix $A^\star_F$ is equimodular. Let B_F be the matrix obtained from A_F by replacing the row χ^N by the row $\chi^N - \chi^L$. Then, B_F is face-defining for F. Moreover, B_F is equimodular by Statement (ii) of Observation 2.6—a contradiction. □

Let $A'_F x = b'_F$ be the system obtained from $A_F x = b_F$ by removing uv's column from A_F and subtracting $h+1$ times this column to b_F. We now show that $\{x \in P_{2h+1}(G') : A'_F x = b'_F\}$ is a face of $P_{2h+1}(G')$ if $\delta(v) \notin \mathcal{M}_F$, and of $P_{2h+1}(G') \cap \{x : x_{uw} = h\}$ otherwise. Indeed, consider a multicut M in $\mathcal{M}_F$. If $M = \delta(v)$, then the equation of $A'_F x = b'_F$ induced by M is nothing but $x_{uw} = h$. Otherwise, by Lemma 2.11 and Claim 3.3, the set $M \backslash uv$ is a multicut of G' (relabelling vw by uw) of order d_M if $uv \notin M$ and $d_M - 1$ otherwise. Thus, the equation of $A'_F x = b'_F$ induced by M is the partition constraint (2a) associated with $M \backslash uv$ set to equality.

By construction and Claim 3.3, A'_F has full row rank and one column less than A_F. We prove that A'_F is face-defining by exhibiting $\dim(F)$ affinely independent points of $P_{2h+1}(G')$ satisfying $A'_F x = b'_F$. Because of the integrality of $P_{2h+1}(G)$, there exist $n = \dim(F) + 1$ affinely independent integer points $z^1, \ldots, z^n$ of F. By Claims 3.2 and 3.3, there exists a multicut strictly containing $\delta(v)$. Then, Statement (i) of Theorem 2.10 implies that $F \subseteq \{x \in \mathbb{R}^E : x_{uv} \leq h+1, x_{vw} \leq h+1\}$. Combined with the partition inequality $x_{uv} + x_{vw} \geq 2h+1$ associated with $\delta(v)$, this implies that at least one of z^i_{uv} and z^i_{vw} is equal to $h+1$ for $i = 1, \ldots, n$. Since exchanging the uv

and vw coordinates of any point of F gives a point of F by Claim 3.3, the hypotheses on $z^1, \dots, z^n$ are preserved under the assumption that $z^i_{uv} = h+1$ for $i = 1, \dots, n-1$. Let $y^1, \dots, y^{n-1}$ be the points obtained from $z^1, \dots, z^{n-1}$ by removing uv's coordinate. Since every multicut of G' is a multicut of G with the same order, $y^1, \dots, y^{n-1}$ belong to $P_{2h+1}(G')$. By construction, they satisfy $A'_F x = b'_F$ so they belong to a face of $P_{2h+1}(G')$ or $P_{2h+1}(G') \cap \{x : x_{uw} = h\}$. This implies that A'_F is a face-defining matrix of $P_{2h+1}(G')$ if $\delta(v) \notin \mathcal{M}_F$, and of $P_{2h+1}(G') \cap \{x : x_{uw} = h\}$ otherwise.

By induction, $P_{2h+1}(G')$ is a box-TDI polyhedron and hence so is $P_{2h+1}(G') \cap \{x : x_{uw} = h\}$. Hence, A'_F is equimodular by Theorem 2.4. Since A_F is obtained from A'_F by copying a column, then A_F is also equimodular—a contradiction. □

By definition of box-TDIness and $Q_k(G)$, Theorem 3.2 implies that $Q_k(G)$ is box-TDI when G is series–parallel. The converse does not hold. Indeed, for instance, when $G = (V, E)$ is a minimal k-edge-connected graph, $Q_k(G)$ is nothing but the single point χ^E so it is a box-TDI polyhedron.

4 An integer TDI system for $P_{2h}(G)$

Let G be a series–parallel graph. In this section we provide an integer TDI system for $P_{2h}(G)$ with h positive and integer.

The proof of the main result of the section is based on the characterization of TDIness by means of Hilbert bases. A set of vectors $\{v^1, \dots, v^k\}$ is a *Hilbert basis* if each integer vector that is a nonnegative combination of $v^1, \dots, v^k$ can be expressed as a nonnegative integer combination of them. The link between Hilbert basis and TDIness is stated in the following theorem.

Theorem 4.1 (Theorem 22.5 of [34]) *A system $Ax \geq b$ is TDI if and only if for every face F of $P = \{x : Ax \geq b\}$, the rows of A associated with tight constraints for F form a Hilbert basis.*

In the previous theorem, we could restrict to minimal faces: indeed, the cone generated by the constraints tight for a face F is a face of the cone generated by the constraints active for a face $F' \subseteq F$ [34].

Remark 4.2 A system $Ax \geq b$ is TDI if and only if, for each minimal face F of $P = \{x : Ax \geq b\}$, the rows of A associated with constraints tight for F form a Hilbert basis.

The rest of the section is devoted to prove that the system given by the partition constraints and the nonnegativity constraints, which describes $P_k(G)$ when k is even, is TDI when G is series–parallel.

The proof is based on the TDIness of System (5) and the structure of inequalities (3a). Their right-hand sides are proportional to k, hence it is enough to prove the case $k = 2$. This allows us to use Theorem 2.13 to obtain a TDI system for $P_2(G)$. In terms of Hilbert bases, the TDIness of this system implies that, given a face F of $P_2(G)$, the integer points of the associated cone are the half sum of the cuts tight for

F. The technical part of the proof is to show that each integer point of this cone is also the sum of incidence vectors of the multicuts tight for F.

Theorem 4.3 *For a series–parallel graph G and a positive integer h, System (3) is TDI.*

Proof We only prove the case $h = 1$ since multiplying the right hand side of a system by a positive constant preserves its TDIness [34, Section 22.5].

The proof is done by induction on the number of edges of the graph $G = (V, E)$. When G consists of two vertices connected by a single edge ℓ, System (3) is $x_\ell \geq 2$, $x_\ell \geq 0$ and is TDI. If G is the circuit $\{e, f\}$, System (3) is $x_e + x_f \geq 2$, $x \geq \mathbf{0}$ and is TDI.

(Parallelization) Let now G be obtained from a series–parallel graph H by adding an edge g parallel to an edge f of H. System (3) associated with G is obtained from that associated with H by duplicating f's column in constraints (3a) and adding the nonnegativity constraint $x_g \geq 0$. By Lemma 3.1 of [7], System (3) is TDI.

For the other cases, we prove the TDIness of System (3) associated with G using Remark 4.2. More precisely, we prove that for any extreme point z of $P_2(G)$, the set of vectors $\{\chi^M : M \in \mathcal{T}_z\} \cup \{\chi^e : e \in E, z_e = 0\}$ is a Hilbert basis, where $\mathcal{T}_z$ is the set of multicuts tight for z.

(1-sum) Let G be the 1-sum of two series–parallel graphs $G^1 = (W^1, E^1)$ and $G^2 = (W^2, E^2)$ and let z be an extreme point of $P_2(G)$. By construction, we have $z = (z^1, z^2)$ where $z^i \in P_2(G^i)$ for $i = 1, 2$. Moreover, for each multicut $M \in \mathcal{T}_z$, the graph obtained from $G(z)$ by contracting the edges of $E \backslash M$ is a circuit. Indeed, it is 2-edge-connected since $G(z)$ is, and it has $z(M) = d_M$ edges and d_M vertices. Therefore M is either a multicut of G^1 tight for z^1 or one of G^2 tight for z^2.

By induction, Systems (3) associated with G^1 and G^2 are TDI. Thus, $\{\chi^M : M \in \mathcal{T}_z \cap \mathcal{M}(G^i)\} \cup \{\chi^e : e \in E^i, z_e = 0\}$ is a Hilbert basis for $i = 1, 2$ by Theorem 4.1. Since they belong to disjoint spaces, their union is a Hilbert basis. By Theorem 4.1, System (3) is TDI.

(Subdivision) Let $G = (V, E)$ be obtained by subdividing an edge uw of a series–parallel graph $G' = (V', E')$ into a path of length two uv, vw, and let z be an extreme point of $P_2(G)$.

Without loss of generality, suppose $z_{uv} \geq z_{vw}$. Define $z' \in \mathbb{Z}^{E'}$ by $z'_{uw} = z_{vw}$ and $z'_e = z_e$ for all edges e in $E' \backslash uw$. Note that z' belongs to $P_2(G')$ since $G'(z')$ is obtained by contracting the edge uv in $G(z)$, and this contraction preserves 2-edge-connectivity.

Note that for all $e \in E$, $z_e \in \{0, 1, 2\}$. Indeed, since z is an extreme point of $P_2(G)$ which is also described by System (1), if $z_e > 0$, then e belongs to a cut D tight for z. Moreover, as $z_{uv} \geq z_{vw}$, the partition constraint (3a) associated with $\delta(v)$ implies that $z_{uv} \in \{1, 2\}$. We now consider two different cases depending on the value of z_{uv}.

Case 1 $z_{uv} = 2$. We first show that every multicut of $\mathcal{T}_z$ containing uv is a bond. Indeed, note that every multicut M with $d_M = 2$ is a bond. If a multicut $M = \delta(V_1, \ldots, V_{d_M}) \in \mathcal{T}_z$ satisfies $d_M \geq 3$ and $uv \in \delta(V_1, V_2)$, then $M' = \delta(V_1 \cup V_2, V_3, \ldots, V_{d_M})$ is a multicut and satisfies

$$z(M') \leq z(M) - 2 < d_M - 1 = d_{M'}.$$

Hence, the partition constraint (3a) associated with M' is violated, a contradiction.

Moreover, there exists at most one bond of $\mathcal{T}_z$, say N, containing uv. As otherwise suppose there exist two bonds B_1 and B_2 in $\mathcal{T}_z$ containing uv. Then, $z(B_1 \triangle B_2) \leq z(B_1) + z(B_2) - 2z_{uv} = 0$, which contradicts the constraint (3a) associated with the multicut $B_1 \triangle B_2$. For a multicut M not containing uv, $M \in \mathcal{T}_z$ if and only if $M \in \mathcal{T}_{z'}$. This implies that $\mathcal{T}_z = \mathcal{T}_{z'} \cup N$. By induction and Theorem 4.1, $\mathcal{T}_{z'} \cup \mathcal{E}_{z'}$ is a Hilbert basis. As $\mathcal{E}_z = \mathcal{E}_{z'}$ (identifying uv and vw) and N is the only member of $\mathcal{T}_z \cup \mathcal{E}_z$ containing uv, $\mathcal{T}_z \cup \mathcal{E}_z$ is also a Hilbert basis.

Case 2 $z_{uv} = 1$. Let $\mathbf{v}$ be an integer point of the cone generated by $\mathcal{T}_z \cup \mathcal{E}_z$. We prove that $\mathbf{v}$ can be expressed as an integer nonnegative combination of the vectors of $\mathcal{T}_z \cup \mathcal{E}_z$. This implies that $\mathcal{T}_z \cup \mathcal{E}_z$ is a Hilbert basis.

Let $\mathcal{B}_z$ be the set of bonds of $\mathcal{T}_z$. Since System (5) is a TDI system describing $P_2(G)$ in series–parallel graphs, the set of vectors $\{\frac{1}{2}\chi^B : B \in \mathcal{B}_z\} \cup \mathcal{E}_z$ forms a Hilbert basis by Theorem 4.1. Then, there exist $\lambda_B \in \frac{1}{2}\mathbb{Z}_+$ for all $B \in \mathcal{B}_z$ and $\mu_e \in \mathbb{Z}_+$ for all $e \in \mathcal{E}_z$ such that $\mathbf{v} = \sum_{B \in \mathcal{B}_z} \lambda_B \chi^B + \sum_{e \in \mathcal{E}_z} \mu_e \chi^e$.

Since $z_{uv} \geq z_{vw}$, the partition inequality (3a) associated with $\delta(v)$ implies that $z_{vw} = 1$ and $\delta(v) \in \mathcal{T}_z$. In particular, $vw \notin \mathcal{E}_z$. The vector $\mathbf{v}$ is an integer combination of vectors of $\mathcal{T}_z \cup \mathcal{E}_z$ if and only if $\mathbf{v} - \lfloor \lambda_{\delta(v)} \rfloor \chi^{\delta(v)}$ is, thus we may assume that $\lambda_{\delta(v)} \in \{0, \frac{1}{2}\}$. Define $\mathbf{w} \in \mathbb{Z}^{E'}$ by:

$$\mathbf{w}_e = \begin{cases} \mathbf{v}_{uv} + \mathbf{v}_{vw} - 2\lambda_{\delta(v)} & \text{if } e = uw, \\ \mathbf{v}_e & \text{otherwise.} \end{cases}$$

Note that $(B \backslash uw) \cup uv$ and $(B \backslash uw) \cup vw$ are bonds of $\mathcal{T}_z$ whenever B is a bond of $\mathcal{T}_{z'}$ containing uw because $z'_{uw} = z_{uv} = z_{vw} = 1$. Moreover, a bond B of $\mathcal{T}_{z'}$ which does not contain uw is a bond of $\mathcal{T}_z$. Since $\delta(v)$ is the unique bond of G containing both uv and vw and $\mathcal{E}_z = \mathcal{E}_{z'}$, we have:

$$\mathbf{w} = \sum_{B \in \mathcal{B}_{z'} : uw \in B} (\lambda_{(B \backslash uw) \cup uv} + \lambda_{(B \backslash uw) \cup vw}) \chi^B + \sum_{B \in \mathcal{B}_{z'} : uw \notin B} \lambda_B \chi^B + \sum_{e \in \mathcal{E}_{z'}} \mu_e \chi^e.$$

Thus, $\mathbf{w}$ belongs to the cone generated by $\mathcal{T}_{z'} \cup \mathcal{E}_{z'}$. By the induction hypothesis, $\mathcal{T}_{z'} \cup \mathcal{E}_{z'}$ is a Hilbert basis, hence there exist $\lambda'_M \in \mathbb{Z}_+$ for all $M \in \mathcal{T}_{z'}$ and $\mu'_e \in \mathbb{Z}_+$ for all $e \in \mathcal{E}_{z'}$ such that $\mathbf{w} = \sum_{M \in \mathcal{T}_{z'}} \lambda'_M \chi^M + \sum_{e \in \mathcal{E}_{z'}} \mu'_e \chi^e$.

Consider the family $\mathcal{N}$ of multicuts of $\mathcal{T}_{z'}$ where each multicut M of $\mathcal{T}_{z'}$ appears λ'_M times. Suppose first that $\lambda_{\delta(v)} = 0$. Then, $\mathbf{v}_{uv} + \mathbf{v}_{vw}$ multicuts of $\mathcal{N}$ contain uw. Let $\mathcal{P}$ be a family of $\mathbf{v}_{uv}$ multicuts of $\mathcal{N}$ containing uw and $\mathcal{Q} = \{F \in \mathcal{N} : uw \in F\} \backslash \mathcal{P}$. Then, we have

$$\mathbf{v} = \sum_{M \in \mathcal{N} : uw \notin M} \chi^M + \sum_{M \in \mathcal{P}} \chi^{(M \backslash uw) \cup uv} + \sum_{M \in \mathcal{Q}} \chi^{(M \backslash uw) \cup vw} + \sum_{e \in \mathcal{E}_{z'}} \mu'_e \chi^e. \quad (6)$$

Suppose now that $\lambda_{\delta(v)} = \frac{1}{2}$. Then, $\mathbf{w}_{uw} = \mathbf{v}_{uv} + \mathbf{v}_{vw} - 1$ multicuts of $\mathcal{N}$ contain uw. Let $\mathcal{P}$ be a family of $\mathbf{v}_{uv} - 1$ multicuts of $\mathcal{N}$ containing uw, let $\mathcal{Q}$ be a family of $\mathbf{v}_{vw} - 1$ multicuts in $\{F \in \mathcal{N} : uw \in F\} \backslash \mathcal{P}$, and denote by N the unique multicut of $\mathcal{N}$ containing uw which is not in $\mathcal{P} \cup \mathcal{Q}$. Then, we have

$$\mathbf{v} = \sum_{M \in \mathcal{N}: uw \notin M} \chi^M + \sum_{M \in \mathcal{P}} \chi^{(M \backslash uw) \cup uv} + \sum_{M \in \mathcal{Q}} \chi^{(M \backslash uw) \cup vw} + \chi^{(N \backslash uw) \cup \delta(v)} + \sum_{e \in \mathcal{E}_{z'}} \mu'_e \chi^e. \tag{7}$$

Every $M \in \mathcal{T}_{z'}$ not containing uw is in $\mathcal{T}_z$. For every $M \in \mathcal{T}_{z'}$ containing uw, $(M \backslash uw) \cup uv$, $(M \backslash uw) \cup vw$ and $(M \backslash uw) \cup \delta(v)$ belong to $\mathcal{T}_z$ since $z'_{uw} = z_{uv} = z_{vw} = 1$. Since $\mathcal{E}_z = \mathcal{E}_{z'}$, then $\mathbf{v}$ is a nonnegative integer combination of vectors of $\mathcal{T}_z \cup \mathcal{E}_z$ in both (6) and (7). This proves that $\mathcal{T}_z \cup \mathcal{E}_z$ is a Hilbert basis. Therefore by Remark 4.2, System (3) is TDI. □

The box-TDIness of $P_k(G)$ and the TDIness of System (3) give the following result.

Corollary 4.4 *System (3) is box-TDI if and only if G is series–parallel.*

Proof If G is series–parallel, then System (3) is box-TDI by Theorems 3.2 and 4.3, since a TDI system describing a box-TDI polyhedron is box-TDI [12]. If G is not series–parallel, Theorem 3.2 ensures that $P_k(G)$ is not box-TDI, therefore System (3) is not box-TDI. □

Theorem 4.3 leaves open the following problem:

Open Problem 4.5 *Characterize the classes of graphs such that System (3) is TDI.*

5 An integer TDI system for $P_{2h+1}(G)$

In this section, we prove that System (2) is TDI if and only if G is a series–parallel graph—see Theorem 5.1. Proving the TDIness for k odd is considerably more involved than for k even. The first difference with the even case is the lack of a known TDI system describing $P_k(G)$ when k is odd, even a noninteger one. Thus, no property of the Hilbert bases associated with $P_k(G)$ is known, and the approach used to prove Theorem 4.3 cannot be applied. Instead, following the definition of TDIness, we prove the existence of an integer optimal solution to each feasible dual problem.

Another difference with the case k even follows from the structure of the partition inequalities (2a). In particular, the presence of the constant "-1" in the right-hand sides perturbs the structure of tight multicuts. Indeed, when k is odd, the tightness of $\delta(V_1, \dots, V_n)$ does not imply that of $\delta(V_1), \dots, \delta(V_n)$. Consequently, it is not clear how the contraction of an edge impacts the tightness of a multicut $\delta(V_1, \dots, V_n)$: merging adjacent V_i's is not sufficient to obtain new tight multicuts. Due to the link between tight multicuts and positive dual variables, the structure of the optimal solutions to the dual problem is completely modified when subdividing an edge. Proving directly that subdivision preserves TDIness turned out to be challenging, and we overcome this difficulty by deriving new properties of series–parallel graphs—see Lemma 2.3.

The proof of Theorem 5.1 focuses on properties of vertices of degree 2 in a minimal counterexample to the TDIness of System (2). In particular, we prove that no two vertices of degree 2 are adjacent (Claim 5.9), or in the same circuit of length 4 (Claim 5.11). Moreover, no triangle contains vertices of degree 2 (Claim 5.10). By Lemma 2.3, this implies that the graph is not series–parallel. To derive these properties, we study the interplay between cuts associated with degree 2 vertices and dual optimal solutions—see Claims 5.3–5.8.

Theorem 5.1 *For h positive and integer, System (2) is TDI if and only if G is series–parallel.*

Proof If G is not series–parallel, then System (2) is not TDI because every TDI system with integer right-hand side describes an integer polyhedron [22], but when G has a K_4-minor, System (2) describes a noninteger polyhedron [10].

We now prove that, if G is series–parallel, then System (2) is TDI. We prove the result by contradiction. Let $G = (V, E)$ be a series–parallel graph such that System (2) is not TDI. By definition of TDIness, there exists $c \in \mathbb{Z}^E$ such that $\mathcal{D}_{(G,c)}$:

$$\max \sum_{M \in \mathcal{M}_G} b_M y_M$$

$$\text{s.t.} \quad \begin{cases} \displaystyle\sum_{M \in \mathcal{M}_G : e \in M} y_M \le c_e & \text{for all } e \in E, \quad (8a) \\ y_M \ge 0 & \text{for all } M \in \mathcal{M}_G, \quad (8b) \end{cases}$$

is feasible, bounded, but admits no integer optimal solution, where $b_M = (h+1)d_M - 1$ for all $M \in \mathcal{M}_G$. Without loss of generality, we assume that:

(i) G has a minimum number of edges,
(ii) $\sum_{e \in E} c_e$ is minimum with respect to (i).

By definition, $\mathcal{D}_{(G,c)}$ is feasible if and only if $c \ge \mathbf{0}$. Hence, by minimality assumption (ii), $\mathcal{D}_{(G,c')}$ has an optimal integer solution for every integer $c' \ne c$ such that $0 \le c' \le c$.

Let M be a multicut of G. We denote by ξ_M the vector of $\{0, 1\}^{\mathcal{M}_G}$ whose only nonzero coordinate is the one associated with M. We say that M is *active* for a solution y to $\mathcal{D}_{(G,c)}$ if $y_M > 0$. Note that, by complementary slackness, a multicut is active for an optimal solution to $\mathcal{D}_{(G,c)}$ only if it is tight for an optimal solution to the primal problem. In particular, if a multicut is tight for no point of $P_{2h+1}(G)$, then it is active for no optimal solution to $\mathcal{D}_{(G,c)}$. Thus, we will use Lemma 2.11 and Theorem 2.10 to deduce properties on the optimal solutions to $\mathcal{D}_{(G,c)}$.

Claim 5.1 *G is simple, 2-connected, and nontrivial.*

Proof Suppose for a contradiction that there exist two parallel edges e_1 and e_2 and $c_{e_1} \le c_{e_2}$. Since a multicut contains either both e_1 and e_2 or none of them, the inequality

(8a) associated with e_2 is redundant because $c_{e_1} \leq c_{e_2}$. This contradicts minimality assumption (i), so G is simple.

Assume for a contradiction that G is not 2-connected. Then G is the 1-sum of two distinct graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. By Statement (ii) of Theorem 2.10, the multicuts of G that intersect both E_1 and E_2 are not tight for the points of $P_{2h+1}(G)$, by complementary slackness, these multicuts are not active for the optimal solutions to $\mathcal{D}_{(G,c)}$. Hence, every optimal solution y to $\mathcal{D}_{(G,c)}$ is of the form:

$$y_M = \begin{cases} y^1_M & \text{if } M \in \mathcal{M}_{G_1}, \\ y^2_M & \text{if } M \in \mathcal{M}_{G_2}, \\ 0 & \text{otherwise}, \end{cases} \qquad \text{for all } M \in \mathcal{M}_G,$$

where y^i is an optimal solution to $\mathcal{D}_{(G_i, c_{|E_i})}$ for $i = 1, 2$. By minimality assumption (i), there exists an integer optimal solution $\bar{y}^i$ to $\mathcal{D}_{(G_i, c_{|E_i})}$ for $i = 1, 2$, implying that $(\bar{y}^1, \bar{y}^2)$ is an integer optimal solution to $\mathcal{D}_{(G,c)}$, a contradiction.

Finally, if $G = K_2$, $\mathcal{M}_G$ contains only one multicut, say $\{e\}$, and the optimal solution to $\mathcal{D}_{(G,c)}$ is $y^*_{\{e\}} = c_e$ which is integer. □

Claim 5.2 *For all edges $e \in E$, $c_e \geq 1$.*

Proof By hypothesis, $c \geq \mathbf{0}$ is integer and $\mathcal{D}_{(G,c)}$ has an optimal solution, say y^*. Suppose for a contradiction that there exists an edge $e \in E$ with $c_e = 0$. Set $G' = G/e$ and $c' = c_{|(E \setminus e)}$. The active multicuts for y^* do not contain the edge e so they are multicuts of G' since $\mathcal{M}_{G'} = \{M \in \mathcal{M}_G : e \notin M\}$. Hence, the point $y' \in \mathbb{R}^{\mathcal{M}_{G'}}$ defined by $y'_M = y^*_M$ for all $M \in \mathcal{M}_{G'}$ is a solution to $\mathcal{D}_{(G',c')}$.

By minimality assumption (i), there exists an integer optimal solution $\tilde{y}$ to $\mathcal{D}_{(G',c')}$. Extending $\tilde{y}$ to a point of $\mathbb{Z}^{\mathcal{M}_G}$ by setting to 0 the missing component gives an integer solution to $\mathcal{D}_{(G,c)}$ with cost $b^\top \tilde{y} \geq b^\top y' = b^\top y^*$. This is an integer optimal solution to $\mathcal{D}_{(G,c)}$ since y^* is optimal, a contradiction to the hypothesis that $\mathcal{D}_{(G,c)}$ has no integer optimal solution. □

Claim 5.3 *Every optimal solution y to $\mathcal{D}_{(G,c)}$ satisfies $0 \leq y_M < 1$ for all $M \in \mathcal{M}_G$.*

Proof By contradiction, suppose that y^* is an optimal solution to $\mathcal{D}_{(G,c)}$ such that there exists a multicut M such that $y^*_M \geq 1$. Therefore, the point y' defined by $y' = y^* - \xi_M$ is a solution to $\mathcal{D}_{(G,c')}$ where $c' = c - \chi^M$. By minimality assumption (ii), $\mathcal{D}_{(G,c')}$ admits an integer optimal solution y''. The point $\tilde{y}$ defined by $\tilde{y} = y'' + \xi_M$ is an integer solution to $\mathcal{D}_{(G,c)}$ and we have:

$$b^\top \tilde{y} = b^\top y'' + b_M \geq b^\top y' + b_M = b^\top y^*.$$

Therefore $\tilde{y}$ is an integer optimal solution to $\mathcal{D}_{(G,c)}$, a contradiction. □

From the definition of series–parallel graphs, Claim 5.1 implies that G contains at least one degree 2 vertex. Let $\widehat{V}$ be the set of degree 2 vertices of G.

Claim 5.4 *Let $v \in \widehat{V}$, $\delta(v) = \{e_1, e_2\}$, y be an optimal solution to $\mathcal{D}_{(G,c)}$, and M_1 be an active multicut for y such that $M_1 \cap \delta(v) = e_1$. If $\delta(v)$ is active for y, then no multicut whose intersection with $\delta(v)$ is e_2 is active for y.*

Proof We prove the result by contradiction. Assume that M_1 and $\delta(v)$ are active for y and that there exists a M_2 active for y with $M_2 \cap \delta(v) = e_2$. By Observation 2.2, $M_i' = M_i \cup \delta(v)$ is a multicut of G such that $d_{M_i'} = d_{M_i} + 1$ for $i = 1, 2$. Let $0 < \varepsilon \leq \min\{y_{M_1}, y_{M_2}, y_{\delta(v)}\}$. Then, the point:

$$y' = y - \varepsilon\left(\chi^{M_1} + \chi^{M_2} + \chi^{\delta(v)}\right) + \varepsilon\left(\chi^{M_1'} + \chi^{M_2'}\right)$$

is a solution to $\mathcal{D}_{(G,c)}$, and we have $b^\top y' = b^\top y + \varepsilon$, implying that y is not optimal, a contradiction. □

Claim 5.5 *For every optimal solution to $\mathcal{D}_{(G,c)}$, the constraints (8a) associated with the edges incident to a degree 2 vertex are tight.*

Proof We prove the result by contradiction. Suppose that there exist an optimal solution y^* to $\mathcal{D}_{(G,c)}$ and a vertex v with $\delta(v) = \{e_1, e_2\}$ such that the inequality (8a) associated with e_1 is not tight. For $i = 1, 2$, let s_i be the slack of the constraint associated with e_i, that is,

$$s_i = c_{e_i} - \sum_{M \in \mathcal{M}_G : e_i \in M} y_M^*.$$

Inequality (8a) associated with e_2 is tight, as otherwise there exists $0 < \eta \leq \min\{s_1, s_2\}$, such that $y^* + \eta\xi_{\delta(v)}$ is a solution to $\mathcal{D}_{(G,c)}$, a contradiction to the optimality of y^*. Hence, Claims 5.2 and 5.3 imply that there are at least two distinct multicuts M_1 and M_2 active for y^* and containing e_2. Let $0 < \varepsilon \leq \min\{y_{M_1}^*, y_{M_2}^*, s_1\}$. For $i = 1, 2$, $e_1 \in M_i$, as otherwise $y' = y^* + \varepsilon(\xi_{M_i \cup e_1} - \xi_{M_i})$ is a solution to $\mathcal{D}_{(G,c)}$. This solution is such that $b^\top y' = b^\top y^* + \varepsilon(h+1) > b^\top y^*$, a contradiction to the optimality of y^*. Thus, both M_1 and M_2 contain $\delta(v)$. Since they are distinct, at least one of them, say M_1, strictly contains $\delta(v)$. Then, $y'' = y^* + \varepsilon(-\xi_{M_1} + \xi_{M_1 \backslash e_2} + \xi_{\delta(v)})$ is a solution to $\mathcal{D}_{(G,c)}$ because $M_1 \backslash e_2$ belongs to $\mathcal{M}_G$ by Lemma 2.11. Then, $b^\top y'' = b^\top y^* + \varepsilon(-b_{M_1} + b_{M_1} - (h+1) + 2h + 1) > b^\top y^*$, a contradiction. □

Given a solution y to $\mathcal{D}_{(G,c)}$, we define for each vertex $v \in \widehat{V}$ the set $\mathcal{A}_v^y$ as the set of multicuts active for y that strictly contain $\delta(v)$. Moreover we define the value α_v^y as:

$$\alpha_v^y = \sum_{M \in \mathcal{A}_v^y} y_M.$$

Claim 5.6 *Every optimal solution y to $\mathcal{D}_{(G,c)}$ satisfies $0 < \alpha_v^y < 1$ for all $v \in \widehat{V}$.*

Proof Suppose for a contradiction that there exist an optimal solution y^* to $\mathcal{D}_{(G,c)}$ and a vertex v of $\widehat{V}$ such that either $\alpha_v^{y^*} \geq 1$ or $\alpha_v^{y^*} = 0$. Denote the two edges incident to v by e_1 and e_2 in such a way that $c_{e_1} \leq c_{e_2}$.

Suppose first that $\alpha_v^{y^*} \geq 1$. By Claim 5.3, there exist at least two multicuts in $\mathcal{A}_v^{y^*}$. Let $\mathcal{A}_v^{y^*} = \{M_1, \ldots, M_n\}$. By Lemma 2.11, for all $i = 1, \ldots, n$, $M_i' = M_i \backslash e_1$ is a multicut of G with $d_{M_i'} = d_{M_i} - 1$. Let $c' = c - \chi^{e_1}$. By $\alpha_v^{y^*} \geq 1$, there exist ϵ_i for all $i = 1, \ldots, n$, such that $0 \leq \epsilon_i \leq y^*_{M_i}$ and $\sum_{i=1}^n \epsilon_i = 1$. The point y^1 defined by:

$$y^1 = y^* + \sum_{i=1}^{n} \left(-\epsilon_i \xi_{M_i} + \epsilon_i \xi_{M_i'}\right)$$

is a solution to $\mathcal{D}_{(G,c')}$. By definition of b, we have:

$$b^\top y^1 = b^\top y^* - h - 1. \tag{9}$$

By minimality assumption (ii), $\mathcal{D}_{(G,c')}$ admits an integer optimal solution, say y^2. This solution satisfies with equality the constraint (8a) associated with e_2 as otherwise $y^2 + \xi_{\delta(v)}$ would be a solution to $\mathcal{D}_{(G,c)}$ with cost $b^\top y^2 + b_{\delta(v)} \geq b^\top y^1 + 2h + 1$, contradicting the assumption that y^* is optimal by (9) and $h \geq 1$. Hence, there exists a multicut $\bar{M}$ active for y^2 containing e_2 but not e_1 since $c'_{e_1} + 1 \leq c'_{e_2}$. By definition, $\bar{M} \cup e_1$ is a multicut of G of order $d_{\bar{M}} + 1$. Define $y^3 \in \mathbb{Z}^{\mathcal{M}_G}$ by:

$$y^3_M = y^2 - \chi^{\bar{M}} + \chi^{\bar{M} \cup e_1}$$

By definition of c' and y^2, the point y^3 is an integer solution to $\mathcal{D}_{(G,c)}$. Therefore, by (9), since y^2 is optimal in $\mathcal{D}_{(G,c')}$ and by definition of y^3, we have:

$$b^\top y^* = b^\top y^1 + h + 1 \leq b^\top y^2 + h + 1 \leq b^\top y^3.$$

Thus, y^3 is an integer optimal solution to $\mathcal{D}_{(G,c)}$, a contradiction.

Suppose now that $\alpha_v^{y^*} = 0$. First, note that $\delta(v)$ is not an active multicut for y^*. Otherwise by Claims 5.2, 5.3 and 5.5, there would be a multicut containing e_1 and not e_2, say N_1, and a multicut containing e_2 and not e_1, say N_2, which are both active for y^*. This contradicts Claim 5.4. Hence, by definition of $\alpha_v^{y^*}$, no active multicut contains $\delta(v)$.

By Observation 2.2, if a multicut M contains e_2 but not e_1, then $M \triangle \delta(v)$ is a multicut with the same order and $b_M = b_{M \triangle \delta(v)}$. Hence, we can define the point $y^4 \in \mathbb{Q}^{\mathcal{M}_G}$:

$$y^4_M = \begin{cases} 0 & \text{if } e_1 \in M, \\ y^*_M + y^*_{M \triangle \delta(v)} & \text{if } e_1 \notin M \text{ and } e_2 \in M, \\ y^*_M & \text{otherwise,} \end{cases} \qquad \text{for all } M \in \mathcal{M}_G,$$

which is a solution to $\mathcal{D}_{(G,\hat{c})}$, where $\hat{c}$ is defined by:

$$\hat{c}_e = \begin{cases} c_{e_1} + c_{e_2} & \text{if } e = e_2, \\ 0 & \text{if } e = e_1, \\ c_e & \text{otherwise,} \end{cases} \qquad \text{for all } e \in E.$$

By construction, we have:

$$b^\top y^4 = b^\top y^*. \tag{10}$$

Using the argument given in the proof of Claim 5.2, we deduce that $\mathcal{D}_{(G,\hat{c})}$ admits an integer optimal solution, say y^5. Let $\mathcal{S}$ be the family of active multicuts for y^5 containing e_2, where each multicut M appears y^5_M times in $\mathcal{S}$. We have $|\mathcal{S}| > c_{e_2}$ as otherwise y^5 would be an integer optimal solution to $\mathcal{D}_{(G,c)}$, a contradiction.

We now construct from y^5 an integer solution y^6 to $\mathcal{D}_{(G,c)}$ with the same cost by replacing e_2 by e_1 in some active multicuts for y^5. More formally, since $|\mathcal{S}| \geq c_{e_1}$, there exists $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| = c_{e_1}$. By Observation 2.2, $M \triangle \delta(v)$ is a multicut of G for all $M \in \mathcal{S}'$ and $b_M = b_{M\triangle\delta(v)}$. Let $y^6 \in \mathbb{Z}^{\mathcal{M}_G}$ be the point defined by:

$$y^6 = y^5 + \sum_{M \in \mathcal{S}'} \left(\xi_{M\triangle\delta(v)} - \xi_M\right)$$

By construction, we have:

$$b^\top y^6 = b^\top y^5. \tag{11}$$

Note that for each $M \in \mathcal{S}'$, adding $\xi_{M\triangle\delta(v)} - \xi_M$ to a point of $\mathbb{R}^{\mathcal{M}_G}$ increases (resp. decreases) by 1 the left-hand side of the inequality (8a) associated with e_1 (resp. e_2) while not changing the left-hand side of the inequalities (8a) associated with the edges of $E\backslash\{e_1, e_2\}$. Therefore, by definition of $\hat{c}$, y^6 is a solution to $\mathcal{D}_{(G,c)}$. By (11), the optimality of y^5, and we have:

$$b^\top y^6 = b^\top y^5 \geq b^\top y^4 = b^\top y^*.$$

Therefore y^6 is an integer optimal solution to $\mathcal{D}_{(G,c)}$, a contradiction. □

Claim 5.6 implies that for each optimal solution y and for each $v \in \widehat{V}$ there exists at least one multicut strictly containing $\delta(v)$ that is active for y. For the following claims we need to define a subset of optimal solutions to $\mathcal{D}_{(G,c)}$: let $\mathfrak{D}_v$ be the set of optimal solutions to $\mathcal{D}_{(G,c)}$ for which $\delta(v)$ is not active. Note that if $\mathfrak{D}_v$ is not empty, then there exists a solution y in $\mathfrak{D}_v$ maximizing α^z_v over all $z \in \mathfrak{D}_v$.

The following claim presents the structure of a specific optimal solution to $\mathcal{D}_{(G,c)}$ for which $\delta(v)$ is not active.

Claim 5.7 *Let $v \in \widehat{V}$ with $\delta(v) = \{e_1, e_2\}$ and let $y^* \in \mathfrak{D}_v$ maximize α^z_v over all $z \in \mathfrak{D}_v$. Then, there are exactly 3 multicuts active for y^* intersecting $\delta(v)$: two bonds $F \cup e_1$ and $F \cup e_2$ and a multicut $F \cup \{e_1, e_2\}$ of order 3, for some $F \subseteq E$.*

Proof By Claim 5.6, there exists at least one multicut strictly containing $\delta(v)$ which is active for y^*, say M_0. By definition of $\mathfrak{D}_v$, $\delta(v)$ is not active for y^*. Hence, by

Claim 5.5, there exists at least one multicut active for y^* which contains e_i and not $\delta(v)\backslash e_i$, for $i = 1, 2$. Let M_i be such a multicut with maximum order.

First, we prove that $d_{M_0} = 3$. By definition, $M_0 = \delta(v, V_2, V_3, \dots, V_{d_{M_0}})$. Moreover, by Lemma 2.11 and complementary slackness, the two vertices adjacent to v belong to two different classes, say V_2 and V_3. By contradiction, suppose that $d_{M_0} \geq 4$. Then, $M_0' = \delta(v \cup V_2 \cup V_3, \dots, V_{d_{M_0}})$ is a multicut of order $d_{M_0} - 2$. For $i = 1, 2$, $M_i' = M_i \cup \delta(v)$ is a multicut of order $d_{M_i} + 1$. Let $0 < \varepsilon \leq \min\{y^*_{M_0}, y^*_{M_1}, y^*_{M_2}\}$. Then, let $y' \in \mathbb{R}^{\mathcal{M}_G}$ be the point defined by:

$$y' = y^* - \varepsilon\xi_{M_0} + \varepsilon\xi_{M_0'} + \varepsilon \sum_{i=1,2} \left(-\xi_{M_i} + \xi_{M_i'}\right).$$

By construction, y' is a solution to $\mathcal{D}_{(G,c)}$ with $b^\top y^* = b^\top y'$. Hence y' is an optimal solution, but we have $\alpha_v^{y'} = \alpha_v^{y^*} + \varepsilon$ because $\delta(v) \subsetneq M_i'$ for $i = 1, 2$. This contradicts the maximality of $\alpha_v^{y^*}$. Therefore $d_{M_0} = 3$.

Now, we show that M_1 is a bond. The result for M_2 holds by symmetry. By contradiction, suppose that $M_1 = \delta(V_1, \dots, V_{d_{M_1}})$ with $d_{M_1} \geq 3$. Without loss of generality, we suppose that $e \in \delta(V_1) \cap \delta(V_2)$. Then, $M_1' = \delta(V_1 \cup V_2, \dots, V_{d_{M_1}})$ is a multicut of order $d_{M_1} - 1$. Moreover, $M_2' = M_2 \cup \delta(v)$ is a multicut of order $d_{M_2} + 1$. Let $0 < \varepsilon \leq \min\{y^*_{M_1}, y^*_{M_2}\}$ and $y' \in \mathbb{R}^{\mathcal{M}_G}$ be the point defined by:

$$y' = y^* - \varepsilon\xi_{M_1} + \varepsilon\xi_{M_1'} - \varepsilon\xi_{M_2} + \varepsilon\xi_{M_2'}.$$

By construction, y' is a solution to $\mathcal{D}_{(G,c)}$ with $b^\top y^* = b^\top y'$. Hence y' is an optimal solution, but we have $\alpha_v^{y'} = \alpha_v^{y^*} + \varepsilon$ because $\delta(v) \subsetneq M_2'$. This contradicts the maximality of $\alpha_v^{y^*}$. Therefore, $d_{M_1} = d_{M_2} = 2$.

We now prove that there exists a set F such that $M_0 = F \cup \delta(v)$, and $M_i = F \cup e_i$ for $i = 1, 2$. Note that $M_1 \cup M_2$ is a multicut so $y'' = y^* + \varepsilon(\xi_{M_1 \cup M_2} - \xi_{M_1} - \xi_{M_2})$ is a solution to $\mathcal{D}_{(G,c)}$. The optimality of y^* implies $d_{M_1 \cup M_2} \leq 3$. Since M_1 and M_2 are distinct bonds, there exists $F \subseteq E\backslash\delta(v)$ such that $M_i = F \cup e_i$, for $i = 1, 2$. Finally, let $N_0 = M_0\backslash e_2$ and $N_1 = M_1 \cup e_2$. Note that $\tilde{y} = y^* + \varepsilon(\xi_{N_0} - \xi_{M_0} + \xi_{N_1} - \xi_{M_1})$ is an optimal solution to $\mathcal{D}_{(G,c)}$ for which $\delta(v)$ is not active. Moreover, N_0 and M_2 are bonds active for $\tilde{y}$ since $d_{M_0} = 3$. This implies that $N_0 = F \cup e_1$, and hence $M_0 = F \cup \delta(v)$.

This implies that M_0, M_1, and M_2 are the only multicuts active for y^* intersecting $\delta(v)$. Indeed, if M is a multicut active for y^* strictly containing $\delta(v)$, then repeating the proof above with M, M_1, and M_2 shows that there exists F' such that $M = F' \cup \delta(v)$, and $M_i = F' \cup e_i$ for $i = 1, 2$. Since $M_i = F \cup e_i$ for $i = 1, 2$, we have $F' = F$ and hence $M = M_0$. A similar argument holds for any multicut active for y^* and intersecting $\delta(v)$. □

Claim 5.8 *Let $v \in \widehat{V}$ and y^* be an optimal solution to $\mathcal{D}_{(G,c)}$. Then,*

(i) *If $y^*_{\delta(v)} = 0$, then $c_e = 1$ for all $e \in \delta(v)$,*

(ii) *If* $y^*_{\delta(v)} > 0$, *then* $\alpha_v^{y^*} + y^*_{\delta(v)} = 1$, *and there exists* $e \in \delta(v)$ *such that* $c_e = 1$.

Proof (i) First suppose that $y^*_{\delta(v)} = 0$, then $\mathfrak{D}_v \neq \emptyset$. Let $y' \in \mathfrak{D}_v$ maximize α_v^z over all $z \in \mathfrak{D}_v$. Then, by Claim 5.7, there exist exactly two active multicuts for y' containing e_i for $i = 1, 2$. Combining Claims 5.3 and 5.5, and the integrality of c, we obtain that $c_{e_i} = 1$ for $i = 1, 2$.

(ii) Let now $y^*_{\delta(v)} > 0$. By Claim 5.4, there exists an edge $e \in \delta(v)$ such that all multicuts containing e that are active for y contain $\delta(v)$. Hence, the constraint (8a) associated with e is:

$$c_e \geq \sum_{M: e \in M} y^*_M = y^*_{\delta(v)} + \sum_{M \in \mathcal{A}_v^{y^*}} y^*_M = y^*_{\delta(v)} + \alpha_v^{y^*}. \tag{12}$$

By Claim 5.5, the constraint (8a) associated with e is tight. Thus, $y^*_{\delta(v)} + \alpha_v^{y^*} = c_e$. By Claims 5.3 and 5.6 and since c_e is integer, we have $c_e = 1$. □

The last three claims of the proof give some structural property of the graph G. In particular we focus our attention on the vertices of $\widehat{V}$.

Claim 5.9 *Vertices of degree 2 are pairwise nonadjacent.*

Proof Assume for a contradiction that there exist two adjacent vertices v_1 and v_2 in $\widehat{V}$, and denote $\delta(v_i) = \{e_0, e_i\}$ for $i = 1, 2$.

We prove that $\delta(v_1)$ is active for all optimal solutions to $\mathcal{D}_{(G,c)}$, the result for $\delta(v_2)$ is obtained by symmetry. By contradiction, suppose that $\mathfrak{D}_{v_1} \neq \emptyset$. Among all the solutions $y \in \mathfrak{D}_{v_1}$, let y^1 be one having $\alpha_{v_1}^y$ maximum. Then, by Claim 5.7, the three multicuts active for y^1 intersecting $\delta(v_1)$ are $M_0 = F \cup \delta(v_1)$, $B_0 = F \cup e_0$, and $B_1 = F \cup e_1$, where B_i are bonds for $i = 0, 1$, and $F \subseteq E \backslash \delta(v_1)$ contains no nonempty multicut. By Claim 5.6 on v_2, there exists a multicut M active for y^1 strictly containing $\delta(v_2)$. By $\delta(v_1) \cap \delta(v_2) = e_0$, M intersects $\delta(v_1)$. Since $d_M \geq 3$, Claim 5.7 for v_1 implies $M = M_0$, $F = \{e_2\}$, and $B_0 = \delta(v_2)$.

As $y^1_{\delta(v_1)} = 0$, by Statement (i) of Claim 5.8, $c_{e_0} = c_{e_1} = 1$. By Claim 5.5, the constraints associated with e_0 and e_1 are tight. Since $\mathcal{A}_{v_1}^{y^1} = \{M_0\}$ by Claim 5.7, we have:

$$c_{e_i} = y^1_{M_0} + y^1_{B_i} = 1 \quad \text{for } i = 0, 1. \tag{13}$$

Let $\{M_1, \ldots, M_n\}$ be the set of active multicuts for y^1 such that $M_i \cap \{e_0, e_1, e_2\} = e_2$, for $i = 1, \ldots, n$. By Claim 5.5, the constraint (8a) associated with e_2 is tight, hence, using (13):

$$c_{e_2} = y^1_{M_0} + y^1_{B_0} + y^1_{B_1} + \sum_{i=1}^{n} y^1_{M_i} = 1 + y^1_{B_0} + \sum_{i=1}^{n} y^1_{M_i}. \tag{14}$$

By Claim 5.3, B_0 active for y^1, and $c_{e_2} \in \mathbb{Z}$, we have $\{M_1, \dots, M_n\} \neq \emptyset$ and $c_{e_2} \geq 2$. Thus, combining (13) and (14), we have:

$$\sum_{i=1}^{n} y^1_{M_i} = c_{e_2} - 1 - y^1_{B_0} \geq y^1_{M_0}. \tag{15}$$

Then, there exist $\epsilon_1, \dots, \epsilon_n$ such that $0 \leq \epsilon_i \leq y^1_{M_i}$ for $i = 1, \dots, n$, and

$$\sum_{i=1}^{n} \epsilon_i = y^1_{M_0}.$$

For $i = 1, \dots, n$, $M_i \cup e_0$ is a multicut with order $d_{M_i} + 1$, hence we can consider the following solution to $\mathcal{D}_{(G,c)}$:

$$y^2 = y^1 - \left(y^1_{M_0} \xi_{M_0} + \sum_{i=1}^{n} \epsilon_i \xi_{M_i} \right) + \left(y^1_{M_0} \xi_{M_0 \setminus e_0} + \sum_{i=1}^{n} \epsilon_i \xi_{M_i \cup e_0} \right). \tag{16}$$

We have $b^\top y^1 = b^\top y^2$, but $\alpha^{y^2}_{v_1} = 0$, a contradiction to Claim 5.6. Therefore $\mathfrak{D}_v \neq \emptyset$, and by symmetry we deduce that both $\delta(v_1)$ and $\delta(v_2)$ are active for all optimal solutions to $\mathcal{D}_{(G,c)}$.

By Claim 5.4, for every optimal solution y to $\mathcal{D}_{(G,c)}$ and every multicut M of G, if M is active for y and contains e_i for some $i \in \{1, 2\}$, then $e_0 \in M$.

Let y^* be the optimal solution to $\mathcal{D}_{(G,c)}$ maximizing $\alpha^{y}_{v_1}$ over all y solutions to $\mathcal{D}_{(G,c)}$. We have $\mathcal{A}^{y^*}_{v_2} \subseteq \mathcal{A}^{y^*}_{v_1}$ and all the multicuts in $\mathcal{A}^{y^*}_{v_2}$ have order at most 3. Otherwise, let $M \in \mathcal{A}^{y^*}_{v_2} \setminus \mathcal{A}^{y^*}_{v_1}$ (resp. $M \in \mathcal{A}^{y^*}_{v_2}$ such that $d_M \geq 4$), and $0 < \varepsilon \leq \min\{y^*_M, y^*_{\delta(v_1)}\}$. The solution

$$y^3 = y^* - \varepsilon(\xi_M + \xi_{\delta(v_1)}) + \varepsilon(\xi_{M \setminus e_2} + \xi_{\delta(v_1) \cup e_2})$$

is optimal, but $\alpha^{y^3}_{v_1} = \alpha^{y^*}_{v_1} + \varepsilon$ by the choice of M, a contradiction to the maximality of $\alpha^{y^*}_{v_1}$. Thus, $\bar{M} = \{e_0, e_1, e_2\}$ is the only multicut in $\mathcal{A}^{y^*}_{v_2}$.

Let $\{N_1, \dots, N_m\}$ be the set of active multicuts for y^* such that $N_i \cap \{e_0, e_1, e_2\} = e_0$ for $i = 1, \dots, m$. The constraint associated with e_0 is tight by Claim 5.5, hence, by $\mathcal{A}^{y^*}_{v_2} \subseteq \mathcal{A}^{y^*}_{v_1}$, we have:

$$c_{e_0} = \alpha^{y^*}_{v_1} + y^*_{\delta(v_1)} + y^*_{\delta(v_2)} + \sum_{i=1}^{m} y^*_{N_i}. \tag{17}$$

By Statement (ii) of Claim 5.8 applied to v_1, we have $y^*_{\delta(v_1)} + \alpha^{y^*}_{v_1} = 1$, and so:

$$c_{e_0} = 1 + y^*_{\delta(v_2)} + \sum_{i=1}^{m} y^*_{N_i}. \tag{18}$$

By $\mathcal{A}^{y^*}_{v_2} = \{\bar{M}\}$ and Statement (ii) of Claim 5.8 applied to v_2, we have $y^*_{\delta(v_2)} + y^*_{\bar{M}} = 1$, hence:

$$c_{e_0} = 2 - y^*_{\bar{M}} + \sum_{i=1}^{m} y^*_{N_i}. \tag{19}$$

Since c_{e_0} is integer and since $y^*_{\bar{M}} < 1$ by Claim 5.3, by (19), we have:

$$\sum_{i=1}^{m} y^*_{N_i} \geq y^*_{\bar{M}}. \tag{20}$$

Hence, let $\lambda_1, \dots, \lambda_m$ be such that $0 \leq \lambda_i \leq y^*_{N_i}$ for $i = 1, \dots, m$, and $\sum_{i=1}^{m} \lambda_i = y^*_{\bar{M}}$. Note that $\delta(v_2) = \bar{M} \backslash e_1$. Then, the point

$$y^5 = y^* - \left(y^*_{\bar{M}} \xi_{\bar{M}} + \sum_{i=1}^{m} \lambda_i \xi_{N_i} \right) + \left(y^*_{\bar{M}} \xi_{\delta(v_2)} + \sum_{i=1}^{m} \lambda_i \xi_{N_i \cup e_1} \right)$$

is a solution to $\mathcal{D}_{(G,c)}$, and it is optimal by definition of b. Moreover,

$$y^5_{\delta(v_2)} = y^*_{\bar{M}} + y^*_{\delta(v_2)} = 1,$$

a contradiction to Claim 5.3. □

The following claim forbids a circuit of length 3 to contain a vertex of $\widehat{V}$.

Claim 5.10 *No circuit of length 3 contains a vertex of degree 2.*

Proof Assume for a contradiction that in G there exist a vertex $v \in \widehat{V}$ and a circuit $\{e_1, e_2, e_3\}$ such that $\delta(v) = \{e_1, e_2\}$. By Lemma 2.1, a multicut contains e_3 only if it intersects $\delta(v)$. On the other hand, by Lemma 2.11 and complementary slackness, each multicut strictly containing $\delta(v)$ and active for an optimal solution contains e_3. Thus, for every optimal solution y to $\mathcal{D}_{(G,c)}$, we have:

$$\sum_{M: e_3 \in M} y_M = \sum_{M: e_1 \in M, M \neq \delta(v)} y_M + \sum_{M: e_2 \in M, M \neq \delta(v)} y_M - \alpha^y_v. \tag{21}$$

Let y^* be an optimal solution to $\mathcal{D}_{(G,c)}$. By the constraint (8a) associated with e_3, (21), and Claim 5.5, we have:

$$c_{e_3} \geq \sum_{M: e_3 \in M} y^*_M = c_{e_1} + c_{e_2} - 2y^*_{\delta(v)} - \alpha^{y^*}_v. \tag{22}$$

By Claim 5.6 and Statement (ii) of Claim 5.8, we have $2y^*_{\delta(v)} + \alpha_v^{y^*} < 2$. Thus, by (22) and $c_{e_3} \in \mathbb{Z}$, we have $c_{e_3} \geq c_{e_1} + c_{e_2} - 1$.

Define $G' = G \backslash e_3$ and $c' = c_{|(E \backslash e_3)}$. Note that for each multicut $M \in \mathcal{M}_G$, $M \backslash e_3$ is a multicut of G' with order at least d_M. Hence, y^* induces a solution to $\mathcal{D}_{(G',c')}$ of cost at least $b^\top y^*$. By minimality assumption (i), there exists an integer optimal solution y' to $\mathcal{D}_{(G',c')}$, and we have $b^\top y' \geq b^\top y^*$.

Let $\mathcal{M}_1$ (resp. $\mathcal{M}_2$) be the set of multicuts $M = \delta(V_1, \ldots, V_{d_M})$ of G' active for y' such that the endpoints of e_3 belong (resp. do not belong) to a same V_i for some $i \in \{1, \ldots, d_M\}$. For each $M \in \mathcal{M}_1$ (resp. $M \in \mathcal{M}_2$), M (resp. $M \cup e_3$) is a multicut of G with the same order. Hence,

$$y'' = \sum_{M \in \mathcal{M}_1} y'_M \xi_M + \sum_{M \in \mathcal{M}_2} y'_M \xi_{M \cup e_3}$$

is a point of $\mathbb{Z}_+^{\mathcal{M}_G}$ with $b^\top y'' = b^\top y'$. Thus, $b^\top y'' \geq b^\top y^*$, and y'' is not a solution to $\mathcal{D}_{(G,c)}$. By definition, y'' respects every constraint of $\mathcal{D}_{(G,c)}$ but the constraint (8a) associated with e_3.

By definition of y'', we have:

$$\sum_{M : e_3 \in M} y''_M = \sum_{M : e_1 \in M, M \neq \delta(v)} y''_M + \sum_{M : e_2 \in M, M \neq \delta(v)} y''_M - \alpha_v^{y''}. \tag{23}$$

Therefore, by y'' violating the constraint (8a) associated with e_3, (23), Statement (ii) of Claim 5.8, and the inequalities (8a) associated with e_1 and e_2, we have:

$$c_{e_3} < \sum_{M : e_3 \in M} y''_M = \sum_{M : e_1 \in M} y''_M + \sum_{M : e_2 \in M} y''_M - \alpha_v^{y''} - 2y''_{\delta(v)} \leq c_{e_1} + c_{e_2} - \alpha_v^{y''} - 2y''_{\delta(v)}. \tag{24}$$

Thus, by (22), we have $\alpha_v^{y''} + 2y''_{\delta(v)} < \alpha_v^{y^*} + 2y^*_{\delta(v)} < 2$. By $c_{e_3} \geq c_{e_2} + c_{e_1} - 1$, the integrality of y'', and (24), we have $\alpha_v^{y''} = y''_{\delta(v)} = 0$, and so $c_{e_3} = c_{e_1} + c_{e_2} - 1$. Hence, by the integrality of y'' and equation (23):

$$c_{e_3} + 1 = \sum_{M : e_3 \in M} y''_M = \sum_{M : e_1 \in M} y''_M + \sum_{M : e_2 \in M} y''_M = c_{e_1} + c_{e_2}. \tag{25}$$

For $i = 1, 2$, since $c_{e_i} \geq 1$, there exists a multicut M_i active for y'' such that $M_i \cap \delta(v) = e_i$.

We claim that the constraint (8a) associated with e_3 is not tight for y^*. By $c_{e_3} = c_{e_1} + c_{e_2} - 1$, (22), and Claim 5.6, $\delta(v)$ is active for y^*. Hence, by Statement (ii) of Claim 5.8, we have:

$$\alpha_v^{y^*} + y^*_{\delta(v)} = 1. \tag{26}$$

Hence, by (21) and Claim 5.5, (26), (25), and $\delta(v)$ active for y^*, we have:

$$\sum_{M : e_3 \in M} y^*_M = c_{e_1} + c_{e_2} - 1 - y^*_{\delta(v)} = c_{e_3} - y^*_{\delta(v)} < c_{e_3}. \tag{27}$$

The point y'' respects all the constraints of $\mathcal{D}_{(G,c)}$ except the one associated with e_3, and this constraint is not tight for y^*. Therefore, there exists $0 < \lambda < 1$ such that

$$\tilde{y} = \lambda y^* + (1 - \lambda) y''$$

is a solution to $\mathcal{D}_{(G,c)}$. Moreover, $\tilde{y}$ is optimal because $b^\top y^* \leq b^\top y''$.

All multicuts active for at least one between y^* and y'' are active for $\tilde{y}$. Since $\delta(v)$ is active for y^* and M_1, M_2 are active for y'', the three multicuts M_1, M_2, and $\delta(v)$ are active for $\tilde{y}$, a contradiction to Claim 5.4. □

Claim 5.11 *Each circuit of length 4 contains at most one vertex of degree 2.*

Proof Assume for a contradiction that there exists a circuit $C = \{e_1, \dots, e_4\}$ in G covering two vertices of $\widehat{V}$, say v_1, v_2. By Claim 5.9, v_1 and v_2 are not adjacent, hence we assume that $\delta(v_1) = \{e_1, e_2\}$ and $\delta(v_2) = \{e_3, e_4\}$. Let v_3 and v_4 be the remaining vertices of C.

We prove that $\delta(v_1)$ is active for all optimal solutions to $\mathcal{D}_{(G,c)}$. Indeed, if $\mathfrak{D}_{v_1} \neq \emptyset$, then let $y' \in \mathfrak{D}_{v_1}$ maximize $\alpha^z_{v_1}$ over all $z \in \mathfrak{D}_{v_1}$. By Statement (ii) of Theorem 2.10, for every multicut M in $\mathcal{A}^{y'}_{v_2}$, we have $M = \delta\{v_2, V_2, \dots, V_{d_M}\}$, with v_3 and v_4 belonging to different V_i's, hence $M \cap \delta(v_1) \neq \emptyset$. However, $M \backslash \delta(v_1)$ contains $\delta(v_2)$, a contradiction to Claim 5.7 applied to v_1. Exchanging the role of v_1 and v_2, we deduce that $\delta(v_2)$ is active for all optimal solutions to $\mathcal{D}_{(G,c)}$.

Without loss of generality, there exists an optimal solution y such that $\alpha^y_{v_1} \geq \alpha^y_{v_2}$. Then, we can build from y an optimal solution y^* to $\mathcal{D}_{(G,c)}$ such that $\mathcal{A}^{y^*}_{v_2} \subseteq \mathcal{A}^{y^*}_{v_1}$. Indeed, suppose $\mathcal{A}^y_{v_2} \backslash \mathcal{A}^y_{v_1} = \{M_1, \dots, M_n\}$. Then, since $\alpha^y_{v_1} \geq \alpha^y_{v_2}$, there exist $N_1, \dots, N_m \in \mathcal{A}^y_{v_1} \backslash \mathcal{A}^y_{v_2}$ such that:

$$\sum_{i=1}^{n} y_{M_i} \leq \sum_{j=1}^{m} y_{N_j}. \tag{28}$$

Hence, there exist $\epsilon_1, \dots, \epsilon_m$ such that $0 \leq \epsilon_j \leq y_{N_j}$, for $j = 1, \dots, m$, and

$$\sum_{j=1}^{m} \epsilon_j = \sum_{i=1}^{n} y_{M_i}. \tag{29}$$

By Statement (ii) of Theorem 2.10 and complementary slackness, v_3 and v_4 belong to different classes of N_j for each $j = 1, \dots, m$, implying that $N_j \cap \delta(v_2) \neq \emptyset$. Moreover, since $N_j \notin \mathcal{A}^y_{v_2}$, we have $|N_j \cap \delta(v_2)| = 1$, for all $j = 1, \dots, m$. Furthermore, since $\delta(v_2)$ is active for y, by Claim 5.4, there exists an edge in $\delta(v_2)$, say e_3, such that $N_j \cap \delta(v_2) = e_3$ for all $j = 1, \dots, m$. Therefore, the point

$$y^* = y - \left(\sum_{i=1}^{n} y_{M_i} \xi_{M_i} - \sum_{i=1}^{n} y_{M_i} \xi_{M_i \backslash e_4} \right) + \left(\sum_{j=1}^{m} \epsilon_j \xi_{N_j \cup e_4} - \sum_{j=1}^{m} \epsilon_j \xi_{N_j} \right) \tag{30}$$

is a solution to $\mathcal{D}_{(G,c)}$ with $b^\top y^* = b^\top y$ and $\mathcal{A}_{v_2}^{y^*} \subseteq \mathcal{A}_{v_1}^{y^*}$. Let $\mathcal{A}_{v_2}^{y^*} = \{M'_1 \dots, M'_p\}$. For $i = 1, \dots, p$, since $M'_i \in \mathcal{A}_{v_1}^{y^*}$, Statement (ii) of Theorem 2.10 implies $M'_i = \delta(v_1, v_2, V_3^i, V_4^i, \dots, V_{d_{M'_i}}^i)$, where V_3^i and V_4^i contain respectively v_3 and v_4. Then, $M''_i = \delta(v_1, v_2 \cup V_3^i \cup V_4^i, \dots, V_{d_{M'_i}}^i)$ is a multicut of order $d_{M'_i} - 2$ for $i = 1, \dots, p$. Since $\delta(v_2)$ is active for y^*, by Statement (ii) of Claim 5.8, we have $\alpha_{v_2}^{y^*} + y^*_{\delta(v_2)} = 1$. Then, the point $y^1 \in \mathbb{Q}^{\mathcal{M}_G}$ defined by:

$$y^1 = y^* - \left(y^*_{\delta(v_2)} \xi_{\delta(v_2)} + \sum_{i=1}^{p} y^*_{M'_i} \xi_{M'_i} \right) + \left(\sum_{i=1}^{p} y^*_{M'_i} \xi_{M''_i} \right),$$

is a solution to $\mathcal{D}_{(G,c')}$, where $c' = c - \chi^{\delta(v_2)}$.

By $d_{M''_i} = d_{M'_i} - 2$ for all $i = 1, \dots, p$, and $\alpha_{v_2}^{y^*} + y^*_{\delta(v_2)} = 1$, we have:

$$b^\top y^1 = b^\top y^* - \alpha_{v_2}^{y^*}(2h+2) - y^*_{\delta(v_2)}(2h+1) = b^\top y^* - (2h+1) - \alpha_{v_2}^{y^*}. \quad (31)$$

By minimality assumption (ii), $\mathcal{D}_{(G,c')}$ admits an integer optimal solution, say y^2. The point $y^3 \in \mathbb{Z}^{\mathcal{M}_G}$ defined by $y^3 = y^2 + \xi_{\delta(v_2)}$ is a solution to $\mathcal{D}_{(G,c)}$ such that:

$$b^\top y^3 = b^\top y^2 + 2h + 1. \quad (32)$$

Therefore, by (31), the optimality of y^2, and (32), we have:

$$b^\top y^* = b^\top y^1 + 2h + 1 + \alpha_{v_2}^{y^*} \leq b^\top y^2 + 2h + 1 + \alpha_{v_2}^{y^*} = b^\top y^3 + \alpha_{v_2}^{y^*}. \quad (33)$$

By integrality of $P_{2h+1}(G)$ and duality, we have $b^\top y^* \in \mathbb{Z}$. Furthermore, y^3 is integer by construction, so $b^\top y^3 \in \mathbb{Z}$. Then, by (33) and Claim 5.6, we have $b^\top y^* \leq b^\top y^3$, and so y^3 is an integer optimal solution to $\mathcal{D}_{(G,c)}$, a contradiction. □

Claims 5.1, 5.9, 5.10, 5.11 and Lemma 2.3 imply that G is not series–parallel, a contradiction. □

The box-TDIness of $P_k(G)$ and the TDIness of System (2) give the following result.

Corollary 5.2 *System (2) is box-TDI if and only if G is series–parallel.*

Proof By Theorem 5.1, when G is not series–parallel, System (2) is not TDI. Whenever G is series–parallel, $P_k(G)$ is box-TDI by Theorem 3.2 and System (2) is TDI by Theorem 5.1. □

Acknowledgements The authors are indebted to the anonymous referees for their careful reading and their valuable comments which greatly helped to improve the presentation of the paper.

References

1. Baïou, M., Barahona, F., Mahjoub, A.R.: Separation of partition inequalities. Math. Oper. Res. **25**(2), 243–254 (2000)
2. Barahona, F., Mahjoub, A.R.: On two-connected subgraph polytopes. Discrete Math. **147**(1–3), 19–34 (1995)
3. Barbato, M., Grappe, R., Lacroix, M., Lancini, E., Wolfler Calvo, R.: The Schrijver system of the flow cone in series-parallel graphs. Discrete Appl. Math. (2020)
4. Bendali, F., Diarrassouba, I., Mahjoub, A.R., Didi Biha, M., Mailfert, J.: A branch-and-cut algorithm for the k-edge connected subgraph problem. Networks **55**(1), 13–32 (2010)
5. Boyd, S.C., Hao, T.: An integer polytope related to the design of survivable communication networks. SIAM J. Discrete Math. **6**(4), 612–630 (1993)
6. Chen, X., Ding, G., Zang, W.: A characterization of box-Mengerian matroid ports. Math. Oper. Res. **33**(2), 497–512 (2008)
7. Chen, X., Ding, G., Zang, W.: The box-TDI system associated with 2-edge connected spanning subgraphs. Discrete Appl. Math. **157**(1), 118–125 (2009)
8. Cheriyan, J., Thurimella, R.: Approximating minimum-size k-connected spanning subgraphs via matching. SIAM J. Comput. **30**(2), 528–560 (2000)
9. Chervet, P., Grappe, R., Robert, L.H.: Box-total dual integrality, box-integrality, and equimodular matrices. Math. Program. **188**, 319–349 (2021)
10. Chopra, S.: The k-edge-connected spanning subgraph polyhedron. SIAM J. Discrete Math. **7**(2), 245–259 (1994)
11. Clarke, L.W., Anandalingam, G.: A bootstrap heuristic for designing minimum cost survivable networks. Comput. Oper. Res. **22**(9), 921–934 (1995)
12. Cook, W.: On box totally dual integral polyhedra. Math. Program. **34**(1), 48–61 (1986)
13. Cornaz, D.: Max-multiflow/min-multicut for $G + H$ series-parallel. Discrete Math. **311**(17), 1957–1967 (2011)
14. Cornaz, D., Grappe, R., Lacroix, M.: Trader multiflow and box-TDI systems in series-parallel graphs. Discrete Optim. **31**, 103–114 (2019)
15. Cornaz, D., Magnouche, Y., Mahjoub, A.R.: On minimal two-edge-connected graphs. In: International Conference on Control, Decision and Information Technologies (CoDIT), pp. 251–256. IEEE (2014)
16. Cornuéjols, G., Fonlupt, J., Naddef, D.: The traveling salesman problem on a graph and some related integer polyhedra. Math. Program. **33**(1), 1–27 (1985)
17. Didi Biha, M., Mahjoub, A.R.: k-edge connected polyhedra on series-parallel graphs. Oper. Res. Lett. **19**(2), 71–78 (1996)
18. Ding, G., Tan, L., Zang, W.: When is the matching polytope box-totally dual integral? Math. Oper. Res. **43**(1), 64–99 (2017)
19. Ding, G., Zang, W., Zhao, Q.: On box-perfect graphs. J. Comb. Theory Ser. B **128**, 17–46 (2018)
20. Duffin, R.J.: Topology of series–parallel networks. J. Math. Anal. Appl. **10**(2), 303–318 (1965)
21. Edmonds, J.: Maximum matching and a polyhedron with 0, 1-vertices. J. Res. Natl. Bur. Stand. B **69**(125–130), 55–56 (1965)
22. Edmonds, J., Giles, R.: A min–max relation for submodular functions on graphs. In: Annals of Discrete Mathematics, vol. 1, pp. 185–204. Elsevier (1977)
23. Edmonds, J., Giles, R.: Total dual integrality of linear inequality systems. In: Pulleyblank, W.R. (ed.) Progress in Combinatorial Optimization, pp. 117–129. Academic Press, Cambridge (1984)
24. Erickson, R.E., Monma, C.L., Veinott, A.F., Jr.: Send-and-split method for minimum-concave-cost network flows. Math. Oper. Res. **12**(4), 634–664 (1987)
25. Fonlupt, J., Mahjoub, A.R.: Critical extreme points of the 2-edge connected spanning subgraph polytope. Math. Program. **105**(2–3), 289–310 (2006)
26. Gabow, H.N., Goemans, M.X., Tardos, É., Williamson, D.P.: Approximating the smallest k-edge connected spanning subgraph by LP-rounding. Netw.: Int. J. **53**(4), 345–357 (2009)
27. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
28. Giles, F.R., Pulleyblank, W.R.: Total dual integrality and integer polyhedra. Linear Algebra Appl. **25**, 191–196 (1979)
29. Grötschel, M., Monma, C.L.: Integer polyhedra arising from certain network design problems with connectivity constraints. SIAM J. Discrete Math. **3**(4), 502–523 (1990)

30. Grötschel, M., Monma, C.L., Stoer, M.: Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. Oper. Res. **40**(2), 309–330 (1992)
31. Lancini, E.: TDIness and multicuts. Université Sorbonne, Paris Nord (2019). Ph.D. thesis
32. Mahjoub, A.R.: Two-edge connected spanning subgraphs and polyhedra. Math. Program. **64**(1–3), 199–208 (1994)
33. Mahjoub, A.R.: On perfectly two-edge connected graphs. Discrete Math. **170**(1–3), 153–172 (1997)
34. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Hoboken (1998)
35. Vandenbussche, D., Nemhauser, G.L.: The 2-edge-connected subgraph polyhedron. J. Comb. Optim. **9**(4), 357–379 (2005)
36. Winter, P.: Generalized Steiner problem in series–parallel networks. J. Algorithms **7**(4), 549–566 (1986)

Dependency Parsing with Bounded Block Degree and Well-nestedness via Lagrangian Relaxation and Branch-and-Bound

Caio Corro Joseph Le Roux Mathieu Lacroix
Antoine Rozenknop Roberto Wolfler Calvo
Laboratoire d'Informatique de Paris Nord,
Université Paris 13 – SPC, CNRS UMR 7030,
F-93430, Villetaneuse, France
{corro,leroux,lacroix,rozenknop,wolfler}@lipn.fr

Abstract

We present a novel dependency parsing method which enforces two structural properties on dependency trees: bounded block degree and well-nestedness. These properties are useful to better represent the set of admissible dependency structures in treebanks and connect dependency parsing to context-sensitive grammatical formalisms. We cast this problem as an Integer Linear Program that we solve with Lagrangian Relaxation from which we derive a heuristic and an exact method based on a Branch-and-Bound search. Experimentally, we see that these methods are efficient and competitive compared to a baseline unconstrained parser, while enforcing structural properties in all cases.

1 Introduction

We address the problem of enforcing two structural properties on dependency trees, namely bounded block degree and well-nestedness, without sacrificing algorithmic efficiency. Intuitively, bounded block degree constraints force each subtree to have a yield decomposable into a limited number of blocks of contiguous words, while well-nestedness asserts that every two distinct subtrees must not interleave: either the yield of one subtree is entirely inside some gap of the other or they are completely separated. These two types of constraints generalize the notion of projectivity: projective trees actually have a block degree bounded to one and are well-nested.

Our first motivation is the fact that most dependency trees in NLP treebanks are well-nested and have a low block degree which depends on the language and the linguistic representation, as shown in (Pitler et al., 2012). Unfortunately, although polynomial algorithms exist for this class of trees (Gómez-Rodríguez et al., 2009), they are not efficient enough to be of practical use in applications requiring syntactic structures. In addition, if either property is dropped, but not the other, then the underlying decision problem becomes harder. That is why practical parsing algorithms are either completely unconstrained (McDonald et al., 2005) or enforce strict projectivity (Koo and Collins, 2010). This work is, to the best of our knowledge, the first attempt to build a discriminative dependency parser that enforces well-nestedness and/or bounded block degree and to use it on treebank data.

We base our method on the following observation: a non-projective dependency parser, thus not requiring neither well-nestedness nor bounded block degree, returns dependency trees satisfying these constraints in the vast majority of sentences. This would tend to indicate that the heavy machinery involved to parse with these constraints is only needed in very few cases.

We consider arc-factored dependency parsing with well-nestedness and bounded block degree constraints. We formulate this problem as an Integer Linear Program (ILP) and apply Lagrangian Relaxation where the dualized constraints are those associated with bounded block degree and well-nestedness. The Lagrangian dual objective then reduces to a maximum spanning arborescence and can be solved very efficiently. This provides an efficient heuristic for our problem. An exact method can be derived by embedding this Lagrangian Relaxation in a Branch-and-Bound procedure to solve the problem with an optimality certificate. Despite the exponential worst-time complexity of the Branch-and-Bound procedure, it is tractable in practice. Our formulation can enforce both types of constraints or only one of them without changing the resolution method.

As stated in (Bodirsky et al., 2009), well-nested dependency trees with 2-bounded block degree are structurally equivalent to derivations in Lexicalized Tree Adjoining Grammars (LTAGs) (Joshi and Schabes, 1997).[1][2] While LTAGs can be parsed in polynomial time, developing an efficient parser for these grammars remains an open problem (Eisner and Satta, 2000) and we believe that this work could be a useful step in that direction.

Related work is reviewed in Section 2. We define arc-factored dependency parsing with block degree and well-nestedness constraints in Section 3. We derive an ILP formulation of this problem in Section 4 and then present our method based on Lagrangian Relaxation in Section 5 and Branch-and-Bound in Section 6. Section 7 contains experimental results on several languages.

2 Related Work

A dynamic programming algorithm has been proposed for parsing well-nested and k-bounded block degree dependency trees in (Gómez-Rodríguez et al., 2009; Gómez-Rodríguez et al., 2011). Unfortunately, it has a prohibitive $O(n^{3+2k})$ time complexity, equivalent to Lexicalized TAG parsing when $k = 2$. Variants of this algorithm have also been proposed for further restricted classes of dependency trees: 1-inherit ($O(n^6)$) (Pitler et al., 2012), head-split ($O(n^6)$) (Satta and Kuhlmann, 2014) and both 1-inherit and head-split ($O(n^5)$) (Satta and Kuhlmann, 2014). Although those restricted classes have good empirical coverage, they do not cover the exact search space of Lexicalized TAG derivation and their time complexity is still prohibitive. Spinal TAGs, described as a dependency parsing task in (Carreras et al., 2008), weaken even more the empirical coverage in practice, restricted to projective trees, but still remain hardly tractable with a complexity of $O(n^4)$. On the contrary, the present work does not restrict the search space.

Parsing mildly context-sensitive languages with dependencies has been explored in (Fernández-González and Martins, 2015) but the resulting parser cannot guarantee compliance with strict structural properties. On the other hand, the present method enforces the well-nestedness and bounded block degree of solutions.

The methods mentioned above all use the graph-based approach and rely on dynamic programming to achieve tractability. There is also a line of of work in transition-based parsing for various dependency classes. Systems have been proposed for projective dependency trees (Nivre, 2003), non-projective, or even unknown classes (Attardi, 2006). Pitler and McDonald (2015) propose a transition system for crossing interval trees, a more general class than well-nested trees with bounded block degree. In the case of spinal TAGs, we can mention the work of Ballesteros and Carreras (2015) and Shen and Joshi (2007). Transition-based algorithms offer low space and time complexity, typically linear in the length of sentences usually by relying on local predictors and beam strategies and thus do not provide any optimality guarantee on the produced structures. The present work follows the graph-based approach, but replaces dynamic programming with a greedy algorithm and Lagrangian Relaxation.

The use of Lagrangian Relaxation to elaborate sophisticated parsing models based on plain maximum spanning arborescence solutions originated in (Koo et al., 2010) where this method was used to parse with higher-order features. This technique has been explored to parse CCG dependencies in (Du et al., 2015) without a precise definition of the class of trees. We can also draw connections between our problem reduction procedure and the use of Lagrangian Relaxation to speed up dynamic programming and beam search with exact pruning in (Rush et al., 2013).

In this work we rely on Non-Delayed Relax-and-Cut for lazy constraint generation (Lucena, 2006). This can be linked to (Riedel, 2009) which uses a cutting plane algorithm to solve MAP inference in Markov Logic and (Riedel et al., 2012) which uses column and row generation for higher-order dependency parsing.

In NLP, the Branch-and-Bound framework (Land and Doig, 1960) has previously been used for dependency parsing with high order features in (Qian and Liu, 2013), and Das et al. (2012) combined Branch-and-Bound to Lagrangian Relaxation in order to retrieve integral solutions for shallow semantic parsing.

[1]It is possible to express a wider class of dependencies with LTAG if we allow dependencies direction to be different from the derivation tree (Kallmeyer and Kuhlmann, 2012).

[2]In order to be fully compatible with LTAGs, we must ensure that the root has only one child. For algorithmic issues see (Fischetti and Toth, 1992) or (Gabow and Tarjan, 1984).

3 Dependency Parsing

We model the dependency parsing problem using a graph-based approach. Given a sentence $s = \langle s_0, \dots, s_n \rangle$ where s_0 is a dummy root symbol, we consider the directed graph $D = (V, A)$ with $V = \{0, \dots n\}$ and $A \subseteq V \times V$. Vertex $i \in V$ corresponds to word s_i and arc $(i, j) \in A$ models a dependency from word s_i to word s_j. In the rest of the paper, we denote $V \setminus \{0\}$ by V^+.

An *arborescence* is a set of arcs T inducing a connected graph with no circuit such that every vertex has at most one entering arc. The set of vertices incident with any arc of T is denoted by $V[T]$. If $V[T] = V$, then T is a *spanning arborescence*. Among the vertices of $V[T]$, the one with no entering arc is called the *root* of T. A vertex t is *reachable* from a vertex s with respect to T if there exists a path from s to t using only arcs of T. The *yield* of a vertex $v \in V$ corresponds to the set of vertices reachable from v with respect to T.

It is well-known that there is a bijection between dependency trees for s and spanning arborescences with root 0 (McDonald et al., 2005). In what follows, the term dependency tree will refer to both the dependency tree of s and its associated spanning arborescence of D with root 0.

In the dependency parsing problem, one has to find a dependency tree with maximal score. Several scores can be associated with each dependency tree and different conditions can restrict the set of valid dependency trees.

In this paper, we consider an *arc-factored* model: each arc $(i, j) \in A$ is assigned a score w_{ij}; the score of a dependency tree is defined as the sum of the scores of the arcs it contains. This model can be computed in $O(n^2)$ with Chu–Liu-Edmonds' algorithm for Maximum Spanning Arborescence (MSA) (McDonald et al., 2005). Unfortunately, this algorithm forbids any modification of the score function, for example adding score contribution for combinations of arcs (i.e. grand-parent or sibling models). Moreover, adding score contribution for combinations of couple of arcs makes the problem NP-hard (McDonald and Pereira, 2006), although several methods have been developed to tackle this problem, for instance dual decomposition (Koo et al., 2010).

Likewise, restrictions on the tree structure such as the well-nestedness and bounded block degree conditions are not permitted in the MSA algorithm. We first give a precise definition of these structural properties, equivalent to (Bodirsky et al., 2009), before we present a method to take them into account. From now on, we suppose that instances are equipped with a positive integer k and we call *valid dependency trees* those satisfying the k-bounded block degree and well-nestedness conditions. A graph-theoretic definition of these two conditions can be given as follows.

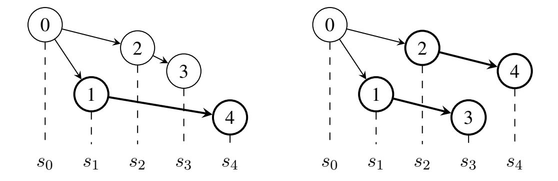

Figure 1: (Left) A 2-BBD arborescence: the block degree of vertex 1 is 2 (its yield is $\{1, 4\}$) whereas the block degree of all other vertices is 1. (Right) A not well-nested arborescence: the yields of vertices 1 and 2 interleave.

Block degree The *block degree* of a vertex set $W \subseteq V$ is the number of vertices of W without a predecessor[3] inside W. Given an arborescence T, the *block degree* of a vertex is the block degree of its yield and the *block degree* of T is the maximum block degree of its incident vertices. An arborescence satisfies the k-bounded block degree condition if its block degree is less than or equal to k. We then say it is *k-BBD* for short. Figure 1 (left) gives an example of a 2-BBD arborescence.

Well-nestedness Two disjoint subsets $I_1, I_2 \subset V^+$ *interleave* if there exist $i, j \in I_1$ and $k, l \in I_2$ such that $i < k < j < l$. An arborescence is *well-nested* if it is not incident to two vertices whose yields interleave. Figure 1 (right) shows an arborescence which is not well-nested.

4 ILP Formulation

In this section we formulate the dependency parsing problem described in Section 3 as an ILP. We start with some notation and two theorems characterizing k-BBD and well-nested dependency trees.

Given a subset $W \subseteq V$, the set of arcs entering W is denoted by $\delta^{\text{in}}(W)$ and the set of arcs leaving W is denoted by $\delta^{\text{out}}(W)$. The set $\delta(W) = \delta^{\text{in}}(W) \cup \delta^{\text{out}}(W)$ is called the *cut* of W. Given a positive integer l, let $\mathcal{W}^{\geq l}$ be the family of vertex subsets of V^+ with block degree greater than or equal to l. For instance, given any sentence with more than 6 words, $\{1, 3, 5, 6\} \in \mathcal{W}^{\geq 3}$,

[3]The predecessor of a vertex $v \in V$ is $v - 1$.

while $\{1,2,5,6\} \notin \mathcal{W}^{\geq 3}$. We also denote by $\mathcal{I}$ the family of couples of disjoint interleaving vertex subsets of V^+. For instance, $(\{1,4\},\{2,3,5\})$ belongs to $\mathcal{I}$. Finally, given a vector $x \in \mathbb{R}^A$ and a subset $B \subseteq A$, $x(B)$ corresponds to $\sum_{a \in B} x_a$.

Theorem 1. *A dependency tree T is not k-BBD iff there exists a vertex subset $W \in \mathcal{W}^{\geq k+1}$ whose cut $\delta(W)$ contains a unique arc of T.*

Proof. By definition of block degree, a dependency tree is not k-BBD iff it is incident with a vertex whose yield W belongs to $\mathcal{W}^{\geq k+1}$. It is equivalent to say that T contains a subarborescence T' such that $V[T']$ equals W. This holds iff W has one entering arc (since $0 \notin W$) and no leaving arc belonging to T. □

Theorem 2. *A dependency tree T is not well-nested iff there exists $(I_1, I_2) \in \mathcal{I}$ such that $\delta(I_1) \cap T$ and $\delta(I_2) \cap T$ are singletons.*

Proof. $\delta(I_1)$ and $\delta(I_2)$ both intersect T only once iff T contains two arborescences T_1 and T_2 such that $V[T_1] = I_1$ and $V[T_2] = I_2$. This means that T is incident with two vertices whose yields are I_1 and I_2, respectively. Result follows from the definition of $\mathcal{I}$ and well-nested arborescences. □

The dependency parsing problem can be formulated as follows. A dependency tree will be represented by its incidence vector. Hence, we use variables $z \in \mathbb{R}^A$ such that $z_a = 1$ if arc a belongs to the dependency tree and 0 otherwise.

$$\max_z \sum_{a \in A} w_a z_a \tag{1}$$

$$z(\delta^{\text{in}}(v)) = 1 \qquad \forall v \in V^+ \tag{2}$$

$$z(\delta^{\text{in}}(W)) \geq 1 \qquad \forall W \subseteq V^+ \tag{3}$$

$$z(\delta(W)) \geq 2 \qquad \forall\, W \in \mathcal{W}^{\geq k+1} \tag{4}$$

$$z(\delta(I_1)) + z(\delta(I_2)) \geq 3 \quad \forall (I_1, I_2) \in \mathcal{I} \tag{5}$$

$$z \in \{0,1\}^A \tag{6}$$

The objective function (1) maximizes the score of the dependency tree. Inequalities (2) ensure that all vertices but the root have one entering arc. Inequalities (3) force the set of arcs associated with z to induce a connected graph. Inequalities (2) and (3), together with $z \geq 0$, give a linear description of the convex hull of the incidence vectors of the spanning arborescences with root 0 — see *e.g.,* (Schrijver, 2003). Inequalities (4) ensure that the dependency tree is k-BBD and inequalities (5) impose well-nestedness. The validity of (4) and (5) follows from Theorems 1 and 2, respectively.

Remark that (3) could be replaced by a polynomial number of additional flow variables and constraints, see (Martins et al., 2009).[4]

5 Lagrangian Relaxation

Solving this ILP using an off-the-shelf solver is ineffective due to the huge number of constraints. We tackle this problem with Lagrangian Relaxation, which has become popular in the NLP community, see for instance (Rush and Collins, 2012). Note that contrary to most previous work on Lagrangian Relaxation for NLP, we do not use it to derive a decomposition method.

We note that optimizing objective (1) subject to constraints (2), (3) and (6) amounts to finding a MSA and can be solved combinatorially (McDonald et al., 2005). Thus, since formulation (1)–(6) is based only on arc variables, by relaxing constraints (4) and (5), one obtains a Lagrangian dual objective which is nothing but a MSA problem with reparameterized arc scores. Our Lagrangian approach relies on a subgradient descent where a MSA problem is solved at each iteration. We give more details in the rest of the section.

5.1 Dual Problem

Let Z be the set of the incidence vectors of dependency trees. Keeping tree shape constraints (2), (3) and (6) while dualizing k-bounded block degree constraints (4) and well-nestedness constraints (5), we build the following Lagrangian (Lemaréchal, 2001):

$$\begin{aligned} L(z,u) = &\sum_{a \in A} w_a z_a \\ &+ \sum_{W \in \mathcal{W}^{\geq k}} u_1^W \times (z(\delta(W)) - 2) \\ &+ \sum_{(I_1,I_2) \in \mathcal{I}} u_2^{I_1,I_2} \times (z(\delta(I_1)) + z(\delta(I_2)) - 3) \end{aligned} \tag{7}$$

[4]Based on this remark, we also developed a formulation of this problem with a polynomial number of variables and constraints. However it requires adding many more variables than (Martins et al., 2009). This leads to a formulation which is not tractable, see Section 7.2. Moreover, it cannot be tackled by our Lagrangian Relaxation approach.

with $z \in Z$ and $u = (u_1, u_2) \geq 0$ is a vector of Lagrangian multipliers. We refactor to:

$$L(z, u) = \sum_{a \in A} \theta_a z_a + c \tag{8}$$

where θ are modified scores and c a constant term.

The dual objective is $L^*(u) = \max_z L(z, u)$ with $z \in Z$. Note that computing $L^*(u)$ amounts to solving the MSA problem with modified scores θ and can be efficiently computed. The dual problem is $\min_{u \geq 0} L^*(u)$. L^* is a non-differentiable convex piece-wise linear function and one can find its minimum via subgradient descent. For any vector u, we use the following subgradient. Denote $Mz \leq b$ the set of constraints given by (4) and (5) and $z^* = \arg\max_z L(z, u)$. Let $g = b - Mz^*$ be a subgradient at u, see (Lemaréchal, 2001) for more details. From this subgradient, we compute the descent direction following (Camerini et al., 1975), which aggregates information during the iteration of the subgradient descent algorithm. Unfortunately, optimizing the dual is expensive with so many relaxed constraints. We handle this problem in the next subsection.

5.2 Efficient Optimization with Many Constraints

The *Non Delayed Relax-and-Cut* (NDRC) method (Lucena, 2005) tackles the problem of optimizing a Lagrangian dual problem with exponentially many relaxed constraints. In standard subgradient descent, at each iteration p of the descent, the Lagrangian update can be formulated as:

$$u^{p+1} = (u^p - s^p \times g^p)_+ \tag{9}$$

where $s^p > 0$ is the stepsize[5] and $()_+$ denotes the projection onto $\mathbb{R}^+$, which replaces each negative component by 0. If all Lagrangian multipliers are initialized to 0, the compononent corresponding to a constraint will not be changed until this constraint is violated for the first time. Indeed, by definition of g, we have $[g^p]_i \geq 0$ if constraint i is satisfied at iteration p: the projection on $\mathbb{R}^+$ ensure that $[u^{p+1}]_i$ stays at 0.[6] Thus we do not need to know constraints that have not been violated yet in order to correctly update the Lagrangian multipliers: this is the main intuition behind the NDRC method. However, s^p may depend on the full subgradient information. A common step size (Fisher, 1981) is:

$$s^p = \alpha^p \times \frac{L^*(u^p) - LB^p}{\|g^p\|^2} \tag{10}$$

with α^p a scalar and LB^p the best known lower bound. This is also the case with more recent approaches like AdaGrad (Duchi et al., 2011) and AdaDelta (Zeiler, 2012). As reported in (Beasley, 1993; Lucena, 2006), when dealing with many relaxed constraints, the $\|g^p\|^2$ term can result in each Lagrangian update being almost equal to 0. Therefore, a good practice is to modify the subgradient such that if $[g^p]_i > 0$ and $[u^p]_i = 0$, then we set $[g^p]_i = 0$: this has the same effect on the multipliers as the projection on $\mathbb{R}^+$ in (9), but it prevents the stepsize from becoming too small. Hence, instead of generating a full subgradient at each iteration, which is an expensive operation because we would need to consider all multipliers associated with constraints, we process only a subpart, namely the one associated with constraints that have been violated.

Following (Lucena, 2005), at each iteration p of the subgradient descent we define two sets: *Currently Violated Active Constraints* (CA^p) and *Previously Violated Active Constraints* (PA^p). CA^p and PA^p are not necessarily disjoint. The subgradient is computed only for constraints in $CA^p \cup PA^p$. At each iteration p, we update $PA^p = PA^{p-1} \cup CA^{p-1}$ and a *violation detection step*, similar to the separation step in a cutting plane algorithm, generates CA^p. Two strategies are possible for the detection: (1) adding to CA^p all the constraints violated by the current dual solution; (2) adding only a subset of them. The latter is justified by the fact that many constraints may overlap thus leading to exageration of modified scores on some arcs. We found that strategy (2) gives better convergence results.

Detection for violated block degree constraints (4) can be done with the algorithm described in (Möhl, 2006) in $O(n^2)$. If no violated block degree constraint is found, we search for violated well-nestedness constraints (5) using the $O(n^2)$ algorithm described in (Havelka, 2007).

5.3 Lagrangian Heuristic

We derive a heuristic from the Lagragian Relaxation. First, a dependency tree is computed with

[5]As stated above, instead of the subgradient we follow an improved descent direction which aggregates information of previous iterations. However, this does not change the proposal of this subsection.

[6]$[x]_i$ denotes the ith component of vector x.

the MSA algorithm. If it is valid, it then corresponds to the optimal solution. Otherwise, we proceed as follows. The computation of the step size in (10) in the subgradient descent needs a lower bound which can be given by the score of any valid dependency tree. In our experiments, we compute the best projective spanning arborescence (Eisner, 2000). Each iteration of the subgradient descent computes a spanning arborescence. Since violating (4) and (5) is penalized in the objective function, it tends to produce valid dependency trees. The heuristic returns the highest scoring one.

6 Branch and Bound

Solving the Lagrangian dual problem may not always give an optimal solution to the original problem because of the potential duality gap. Still, we always obtain an upper bound on the optimal solution and if a dual solution satisfies constraints (4) and (5), its score with the original weights provides a lower bound.[7]

Moreover, the subgradient descent algorithm theoretically converges but we have no guarantee that this will happen in a realistic number of iterations. Therefore, in order to retrieve an optimal solution in all cases, we embed the Lagrangian Relaxation of the problem within a Branch-and-Bound procedure (Land and Doig, 1960).

The search space is recursively split according to an arc variable, creating two subspaces, one where it is fixed to 1 and the other to 0 (branching step). The procedure returns a candidate solution when all arc variables are fixed and constraints are satisfied, and the optimal solution is the highest-scoring candidate solution.

For each subspace, we estimate an upper bound using the Lagrangian Relaxation (bounding step). The recursive exploration of a subspace stops (pruning step) if (1) we can prove that all candidate solutions it contains have a score lower than the best found so far, or (2) we detect an unsastifiable constraint.

The branching strategy is built upon Lagrangian multipliers: we branch on the variable z_a with highest value $\theta_a - w_a$. Intuitively, if the branching step sets $z_a = 1$, it indicates that we add a *hard* constraint on an arc which has been strongly promoted by Lagrangian Relaxation. This strategy, compared to other variants, gave the best parsing time on development data.

[7]Because relaxed constraints are inequalities, constraint satisfaction does not guarantee optimality (Beasley, 1993).

6.1 Problem Reduction

The efficiency of the Branch-and-Bound procedure crucially depends on the number of free variables. To prune the search space, we rely on problem reduction (Beasley, 1993), once again based on duality and Lagrangian Relaxation, which provides certificates on optimal variable settings.

We fix a variable to 1 (resp. 0), and compute an upper bound on the optimal score with this new constraint. If it is lower than the score of the best solution found so far without this constraint, we can guarantee that this variable cannot (resp. must) be in the optimal solution and safely set it to 0 (resp. 1).

Problem reduction is performed at each node of the Branch-and-Bound tree after computing the upper bound with subgradient descent.

6.2 Fixing Variables to 1

Since a node in V^+ must have exactly one parent, fixing $z_{ij} = 1$ for an arc $a = (i, j)$ greatly reduces the problem size, as it will also fix $z_{hj} = 0, \forall h \neq i$. Among all arc variables that can be set to 1, promising candidates are the arcs in a solution of the unconstrained MSA and the arcs obtained in a solution after the subgradient descent.

There are exactly n such arcs in each set of candidates, so we test fixation for less than $2n$ variables. In this case, we are ready to pay the price of a quadratic computation for each of these arcs.

Hence, for each candidate arc we obtain an upper bound by seeking the (unconstrained) MSA on the graph where this arc is removed. If this upper bound is lower than the score of the best solution found so far, we can safely say that this arc is in the optimal solution.

6.3 Fixing Variables to 0

We could apply the same strategy for fixing variables to 0. However, this reduction is less rewarding and there are many more variables set to 0 than 1 in a MSA solution. Instead, we solve an easier problem, at the expense of a looser upper bound.

For each arc a which is not in the MSA, we compute a maximum directed graph that contains this arc and where all nodes but the root have one parent. Remark that if this graph is connected then it corresponds to a dependency tree. Therefore, the score of this directed graph provides an upper bound on a solution containing arc a. If this upper bound is lower than the score of the best solution found so far, we can fix the variable z_a to 0.

Note that this whole fixing procedure can be done in $O(n^2)$.

7 Experiments

We ran a series of experiments to test our method in the case of unlabelled dependency parsing. Our prototype has been developped in Python with some parts in Cython and C++. We use the MSA implementation available in the LEMON library.[8]

7.1 Datasets

We ran experiments on 5 different corpora:

English: Dependencies were extracted from the WSJ part of the Penn Treebank (PTB) with additional NP bracketings (Vadas and Curran, 2007) with the *LTH* converter[9] (default options). Sections 02-21 are used for training, 22 for development and 23 for testing. POS tags were predicted by the Stanford tagger[10] trained with 10-jackkniffing.[11]

German: We used dependencies from the SPMRL dataset (Seddah et al., 2014), with predicted POS tags and the official split. We removed sentences of length greater than 100 in the test set.

Dutch, Spanish and Portuguese: We used the Universal Dependency Treebank 1.2 (Van der Beek et al., 2002; McDonald et al., 2013; Afonso et al., 2002) with gold POS tags and the official split. We removed sentences of length greater than 100 in the test sets.

Those datasets contain different structure distributions as shown in Table 1. Fortunately, our method allows us to easily change the bounded degree constraint or toggle the well-nestedness one. For each language, we decided to use the most constrained combination of bounded block degree constraints and well-nestedness which covers over 99% of the data. Therefore, we chose to enforce 2-BBD and well-nestedness for English and Spanish, 3-BBD and well-nestedness for Dutch and Portuguese and 3-BBD only for German.

7.2 Decoding

In order to compare our methods with previous approaches, we tested five decoding strategies. (`MSA`) computes the best unconstrained dependency tree. (`Eisner`) computes the best projective tree. (`LR`) and (`B&B`) are the heuristic and the exact method presented in Sections 5.3 and 6 respectively.[12] Finally (`MSA/Eisner`) consists in running the MSA algorithm and, if the solution is invalid, returning the (`Eisner`) solution instead.

Our attempt to run the dynamic programming algorithm of (Gómez-Rodríguez et al., 2009) was unsuccessful. Even with heavy pruning we were not able to run it on sentences above 20 words. We also tried to use CPLEX on a compact ILP formulation based on multi-commodity flows (see footnote 4). Parsing time was also prohibitive: a total of 3473 seconds on English data without the well-nestedness constraint, 7298 for German.

We discuss the efficiency of our methods on data for English and German. Other languages give similar results. Optimality rate after the subgradient descent are reported in Figure 2. We see that Lagrangian Relaxation often returns optimal solutions but fails to give a certificate of their optimality. Table 2 shows parsing times. We see that (`LR`) and (`B&B`), while slower than (`MSA`), are fast in the majority of cases, below the third quartile. Inevitably, there are some rare cases where a large portion of the search space is explored, and thus their parsing time is high. Let us remark that these algorithms are run only when (`MSA`) returns an invalid structure, and so total time is very acceptable compared to the baseline.

Finally, we stress the importance of problem reduction as a pre-processing step in `B&B`: after subgradient descent is performed, it removes an average of 83.97% (resp. 76.59%) of arc variables in the English test set (resp. German test set).

7.3 Training

Feature weights are trained using the averaged structured perceptron (Collins, 2002) with 10 iterations where the best iteration is selected on the development set. We used the same feature set as in TurboParser (Martins et al., 2010), including features for lemma. For German, we additionally use morpho-syntactic features.

The decoding algorithm used at training time is the MSA. We experimented with Branch-and-Bound and Lagrangian Relaxation decoding dur-

[8] `https://lemon.cs.elte.hu/trac/lemon`

[9] `http://nlp.cs.lth.se/software/treebank_converter/`

[10] `http://nlp.stanford.edu/software/tagger.shtml`

[11] Prediction precision: 97.40%

[12] In both methods, the subgradient descent is stopped after a fixed maximum number of iterations. We chose 100 for English and 200 for other languages after tuning on the development set.

	English		German		Dutch		Spanish		Portuguese	
	WN	IL	WN	IL	WN	IL	WN	IL	WN	IL
Block degree 1	92.26	-	67.60	-	69.13	-	93.95	-	81.56	0.05
Block degree 2	7.58	0.12	27.12	0.79	28.50	0.08	5.99	0.04	13.92	0.02
Block degree 3	0.12	0.01	3.86	0.30	2.24	0.01	0.02	-	3.76	-
Block degree 4	-	-	0.19	<0.01	0.04	-	-	-	0.54	-
Block degree > 4	-	-	0.11	<0.01	-	-	-	-	0.14	-

Table 1: Distribution of dependency tree characteristics in datasets.

	English (96 sentences)			German (59 sentences)		
	MSA	LR	B&B	MSA	LR	B&B
Mean	0.02	0.26	0.53	0.04	0.51	0.71
Std.	0.01	0.20	0.86	0.02	0.41	1.39
Med.	0.02	0.21	0.27	0.03	0.47	0.47
3rd	0.03	0.34	0.53	0.05	0.71	0.80
Total	1.81	25.09	50.52	2.18	30.19	42.20

Table 2: Timings for strategies (see Section 7.2) on test for solutions which do not satisfy constraints after running MSA. We give (in seconds) average time, standard deviation, median time, time to parse up to the 3rd quartile and total time.

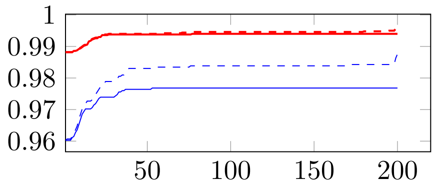

Figure 2: Optimality rate (y-axis) vs number of subgradient iterations (x-axis) for English (thin blue) and German (thick red). Solid line is the optimal rate with certificate, dashed is without.

ing training. It did not significantly improve accuracy and made training and decoding slower.

7.4 Parsing Results

Table 3 shows attachment score (UAS), percentage of valid dependency trees and relative time to (MSA) for different systems for our five decoding strategies. We can see (B&B) is on a par with (LR) on some corpora and more accurate on the others. The former takes more time, and the improvement is correlated with time difference for all corpora but the PTB. Dividing the five corpora in three cases, we can see that:

1. For English and Spanish, where projective dependency trees represent more than 90% of the data, (Eisner) outperforms (MSA). Our methods lie between the two. Here it is better to search for projective trees and (LR) and (B&B) are not interesting in terms of UAS. This is confirmed by the results of (MSA/Eisner).

2. For German and Dutch, where projective dependency trees represent less than 70% of the data, (MSA) outperforms (Eisner). For German, where well-nestedness is not required, our methods are as accurate as (MSA)[13], while for Dutch our methods seem to be useful, as (B&B) outperforms all systems. Moreover, our two methods guarantee validity.

3. For Portuguese, where projective dependency trees represent around 80% of the data, (MSA) is as accurate as (Eisner). In this case we see that, while our heuristic is below, the exact method is more accurate. This seems to be an edge case where neither unconstrained nor projective dependency trees seem to adequately capture the solution space. We also see that it is harder for our methods to give solutions (longer computation times, which tend to indicate that LR cannot guarantee optimality). Our methods are best fitted for this case.

In order to see how much well-nested and bounded block-degree structures are missed by a state-of-the-art parser, we compare our results with TurboParser.[14] We run the parser with three different feature sets: arc-factored, standard (second-order features), and full (third-order features). The results are shown in Table 4. Our model, by enforcing strict compliance to structural rules (100% valid dependency trees), is closer to the empirical distribution than TurboParser in arc-factored mode on all languages but German. Higher-order scoring functions manage to get more similar to the treebank data than our strict thresholds for all languages but Portuguese, at the expense of a significative computational burden.

[13]For German, we notice a small regression which we attribute to the representation of enumerations in the corpus: for enumerations of k elements, k-bounded block-degree subtrees must be generated.

[14]We used 2.1.0 and all defaults but the feature set.

		MSA	Eisner	LR	B&B	MSA/Eisner
English	UAS	89.45	**89.82**	89.54	89.53	89.60
	2-BBD/WN	96.02	–	–	–	–
	Relative Time	1	2.5	1.8	2.5	1.2
German	UAS	**87.79**	86.97	87.78	87.78	87.46
	3-BBD	98.81	–	–	–	–
	Relative Time	1	2.1	1.5	1.7	1.3
Dutch	UAS	77.30	76.62	76.96	**77.40**	76.79
	3-BBD/WN	94.82	–	–	–	–
	Relative Time	1	1.5	1.7	5	1.3
Spanish	UAS	83.37	**83.56**	83.37	83.44	83.48
	2-BBD/WN	92.62	–	–	–	–
	Relative Time	1	2.8	2.7	3	1.5
Portuguese	UAS	83.13	83.14	82.99	**83.21**	82.90
	3-BBD/WN	87.84	–	–	–	–
	Relative Time	1	2.7	5.7	19.7	1.7

Table 3: UAS, percentage of valid structure and decoding time for test data. Time is relative to MSA decoding. The percentage of valid structure is always 100% except for MSA decoding.

	English (99.84)			**German (99.27)**			**Dutch (99.87)**			**Spanish (99.94)**			**Portuguese (99.24)**		
Order	UAS	VDT	RT	UAS	VDT	RT	UAS	VDT	RT	UAS	VDT	RT	UAS	VDT	RT
1st	89.29	94.87	1	87.97	98.74	1	76.10	93.26	1	83.11	93.43	1	83.53	94.79	1
2nd	92.04	99.75	16	89.83	99.28	16	79.05	97.93	18	86.61	98.54	10	87.35	98.96	15
3rd	92.37	99.75	34	90.35	99.24	36	79.68	97.41	37	87.31	99.64	18	88.09	98.98	32

Table 4: UAS, percentage of valid dependency trees (VDT) and relative time (RT) obtained by Turboparser for different score functions on test sets. For each language we give the percentage of valid dependency structures in the data, according to the constraints postulated in Section 7.1.

We interpret this fact as an indication that adding higher order features into our system would make the relaxation method converge more often and faster.

8 Conclusion

We presented a novel characterization of dependency trees complying with two structural rules: bounded block degree and well-nestedness from which we derived two methods for arc-factored dependency parsing. The first one is a heuristic which relies on Lagrangian Relaxation and the Chu-Liu-Edmonds efficient maximum spanning arborescence algorithm. The second one is an exact Branch-and-Bound procedure where bounds are provided by Lagrangian Relaxation. We showed experimentally that these methods give results comparable with state-of-the-art arc-factored parsers, while enforcing constraints in all cases.

In this paper we focused on arc-factor models, but our method could be extended to higher order models, following the dual decomposition method presented in (Koo et al., 2010) in which the maximum-weight spanning arborescence component would be replaced by our constrained model.

Our method opens new perspectives for LTAG parsing, in particular using decomposition techniques where dependencies and templates are predicted separately. Moreover, while well-nested dependencies with 2-bounded block degree can represent LTAG derivations, toggling the well-nestedness or setting the block degree bound allows to express the whole range of derivations in lexicalized LCFRS, whether well-nested or with a bounded fan-out. Our algorithm can exactly represent these settings with a comparable complexity.

Acknowledgements

We thank the anonymous reviewers for their insightful comments which let us significantly improve the submitted paper. This work is supported by a public grant overseen by the French National Research Agency (ANR) as part of the Investissements d'Avenir program (ANR-10-LABX-0083).

References

[Afonso et al.2002] Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. 2002. Floresta sintá (c) tica: A treebank for portuguese. In *LREC*.

[Attardi2006] Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 166–170. Association for Computational Linguistics.

[Ballesteros and Carreras2015] Miguel Ballesteros and Xavier Carreras. 2015. Transition-based spinal parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 289–299, Beijing, China, July. Association for Computational Linguistics.

[Beasley1993] John E Beasley. 1993. Lagrangian relaxation. In *Modern heuristic techniques for combinatorial problems*, pages 243–303. John Wiley & Sons, Inc.

[Bertsekas1999] Dimitri P Bertsekas. 1999. *Nonlinear programming*. Athena scientific.

[Bodirsky et al.2009] Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2009. Well-nested drawings as models of syntactic structure. In *Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*, pages 195–203.

[Camerini et al.1975] Paolo M Camerini, Luigi Fratta, and Francesco Maffioli. 1975. On improving relaxation methods by modified gradient techniques. In *Nondifferentiable optimization*, pages 26–34. Springer.

[Carreras et al.2008] Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics.

[Collins2002] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

[Das et al.2012] Dipanjan Das, André FT Martins, and Noah A Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 209–217. Association for Computational Linguistics.

[Du et al.2015] Yantao Du, Weiwei Sun, and Xiaojun Wan. 2015. A data-driven, factorization parser for CCG dependency structures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1545–1555.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Eisner and Satta2000] Jason Eisner and Giorgio Satta. 2000. A faster parsing algorithm for lexicalized tree-adjoining grammars. In *Proceedings of the 5th Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+ 5)*, pages 14–19.

[Eisner2000] Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.

[Fernández-González and Martins2015] Daniel Fernández-González and André F. T. Martins. 2015. Parsing as Reduction. In *Annual Meeting of the Association for Computational Linguistics (ACL'15)*, Beijing, China, July.

[Fischetti and Toth1992] Matteo Fischetti and Paolo Toth. 1992. An additive bounding procedure for the asymmetric travelling salesman problem. *Mathematical Programming*.

[Fisher1981] Marshall L Fisher. 1981. The lagrangian relaxation method for solving integer programming problems. *Management science*, 27.

[Gabow and Tarjan1984] Harold N. Gabow and Robert E. Tarjan. 1984. Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms*, 5(1):80–131.

[Gómez-Rodríguez et al.2009] Carlos Gómez-Rodríguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 291–299. Association for Computational Linguistics.

[Gómez-Rodríguez et al.2011] Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational linguistics*, 37(3):541–586.

[Havelka2007] Jiří Havelka. 2007. Relationship between non-projective edges, their level types, and well-nestedness. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 61–64. Association for Computational Linguistics.

[Joshi and Schabes1997] Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

[Kallmeyer and Kuhlmann2012] Laura Kallmeyer and Marco Kuhlmann. 2012. A formal model for plausible dependencies in lexicalized tree adjoining grammar. In *Proceedings of TAG*, volume 11, pages 108–116.

[Koo and Collins2010] Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

[Koo et al.2010] Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics.

[Land and Doig1960] Ailsa H. Land and Alison G. Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, 28(3):497–520.

[Lemaréchal2001] Claude Lemaréchal. 2001. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer.

[Lucena2005] Abilio Lucena. 2005. Non delayed relax-and-cut algorithms. *Annals of Operations Research*, 140(1):375–410.

[Lucena2006] Abilio Lucena. 2006. Lagrangian relax-and-cut algorithms. In *Handbook of Optimization in Telecommunications*, pages 129–145. Springer.

[Martins et al.2009] André FT Martins, Noah A Smith, and Eric P Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 342–350. Association for Computational Linguistics.

[Martins et al.2010] André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 34–44.

[McDonald and Pereira2006] Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

[McDonald et al.2005] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

[McDonald et al.2013] Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97. Citeseer.

[Möhl2006] Mathias Möhl. 2006. *Drawings as models of syntactic structure: Theory and algorithms*. Ph.D. thesis, Saarland University.

[Nivre2003] Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*.

[Pitler and McDonald2015] Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 662—-671.

[Pitler et al.2012] Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 478–488. Association for Computational Linguistics.

[Qian and Liu2013] Xian Qian and Yang Liu. 2013. Branch and bound algorithm for dependency parsing with non-local features. *Transactions of the Association for Computational Linguistics*, 1:37–48.

[Riedel et al.2012] Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut: delayed column and row generation for graph based parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 732–743. Association for Computational Linguistics.

[Riedel2009] Sebastian Riedel. 2009. Cutting plane map inference for markov logic. In *SRL 2009*.

[Rush and Collins2012] Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*.

[Rush et al.2013] Alexander M. Rush, Yin-Wen Chang, and Michael Collins. 2013. Optimal beam search for machine translation. In *Proceedings of EMNLP*.

[Satta and Kuhlmann2014] Giorgio Satta and Marco Kuhlmann. 2014. Efficient parsing for head-split dependency trees. *Transactions of the Association for Computational Linguistics*, 1:267–278.

[Schrijver2003] A. Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer.

[Seddah et al.2014] Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109.

[Shen and Joshi2007] Libin Shen and Aravind K Joshi. 2007. Bidirectional ltag dependency parsing. Technical report, Technical Report 07-02, IRCS, University of Pennsylvania.

[Vadas and Curran2007] David Vadas and James R. Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 240–247.

[Van der Beek et al.2002] Leonoor Van der Beek, Gosse Bouma, Rob Malouf, and Gertjan Van Noord. 2002. The alpino dependency treebank. *Language and Computers*, 45(1):8–22.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Efficient Discontinuous Phrase-Structure Parsing via the Generalized Maximum Spanning Arborescence

Caio Corro **Joseph Le Roux** **Mathieu Lacroix**
Laboratoire d'Informatique de Paris Nord,
Université Paris 13 – SPC, CNRS UMR 7030,
F-93430, Villetaneuse, France
{corro,leroux,lacroix}@lipn.fr

Abstract

We present a new method for the joint task of tagging and non-projective dependency parsing. We demonstrate its usefulness with an application to discontinuous phrase-structure parsing where decoding lexicalized spines and syntactic derivations is performed jointly. The main contributions of this paper are (1) a reduction from joint tagging and non-projective dependency parsing to the Generalized Maximum Spanning Arborescence problem, and (2) a novel decoding algorithm for this problem through Lagrangian relaxation. We evaluate this model and obtain state-of-the-art results despite strong independence assumptions.

1 Introduction

Discontinuous phrase-structure parsing relies either on formal grammars such as LCFRS, which suffer from a high complexity, or on reductions to non-projective dependency parsing with complex labels to encode phrase combinations. We propose an alternative approach based on a variant of spinal TAGs, which allows parses with discontinuity while grounding this work on a lexicalized phrase-structure grammar. Contrarily to previous approaches, (Hall and Nivre, 2008; Versley, 2014; Fernández-González and Martins, 2015), we do not model supertagging nor spine interactions with a complex label scheme. We follow Carreras et al. (2008) but drop projectivity.

We first show that our discontinuous variant of spinal TAG reduces to the Generalized Maximum Spanning Arborescence (GMSA) problem (Myung et al., 1995). In a graph where vertices are partitioned into clusters, GMSA consists in finding the arborescence of maximum weight incident to exactly one vertex per cluster. This problem is NP-complete even for arc-factored models. In order to bypass complexity, we resort to Lagrangian relaxation and propose an efficient resolution based on dual decomposition which combines a simple non-projective dependency parser on a contracted graph and a local search on each cluster to find a global consensus.

We evaluated our model on the discontinuous PTB (Evang and Kallmeyer, 2011) and the Tiger (Brants et al., 2004) corpora. Moreover, we show that our algorithm is able to quickly parse the whole test sets.

Section 2 presents the parsing problem. Section 3 introduces GMSA from which we derive an effective resolution method in Section 4. In Section 5 we define a parameterization of the parser which uses neural networks to model local probabilities and present experimental results in Section 6. We discuss related work in Section 7.

2 Joint Supertagging and Spine Parsing

In this section we introduce our problem and set notation. The goal of phrase-structure parsing is to produce a derived tree by means of a sequence of operations called a derivation. For instance in context-free grammars the derived tree is built from a sequence of substitutions of a non-terminal symbol with a string of symbols, whereas in tree adjoining grammars (TAGs) a derivation is a sequence of substitutions and adjunctions over elementary trees. We are especially interested in building discontinuous phrase-structure trees which may contain constituents with gaps.[1]

We follow Shen (2006) and build derived trees from adjunctions performed on spines. Spines are lexicalized unary trees where each level represents

[1] Although we will borrow concepts from TAGs, we do not require derivations to be TAG compatible (i.e. well-nested dependencies with a bounded number of gaps).

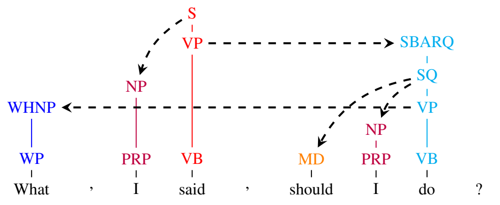

Figure 1: A derivation with spines and adjunctions (dashed arrows). The induced dependency tree is non-projective. Each color corresponds to a spine. We omit punctuation to simplify figures.

a lexical projection of the anchor. Carreras et al. (2008) showed how spine-based parsing could be reduced to dependency parsing: since spines are attached to words, equivalent derivations can be represented as a dependency tree where arcs are labeled by spine operations, an adjunction together with information about the adjunction site. However, we depart from previous approaches (Shen and Joshi, 2008; Carreras et al., 2008) by relaxing the projectivity constraint to represent all discontinuous phrase-structure trees (see Figure 1).

We assume a finite set of spines S. A spine s can be defined as a sequence of grammatical categories, beginning at root. For a sentence $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ where w_k is the word at position k and w_0 is a dummy root symbol, a derivation is a triplet $(\mathbf{d}, \mathbf{s}, \mathbf{l})$ defined as follows. Adjunctions are described by a dependency tree rooted at 0 written as a sequence of arcs $\mathbf{d}$. If $(h, m) \in \mathbf{d}$ with $h \in \{0, \ldots, n\}$ and $m \in \{1, \ldots, n\}$, then the derivation contains an adjunction of the root of the spine at position m to a node from the spine at position h. Supertagging, the assignment of a spine to each word, is represented by a sequence $\mathbf{s} = (s_0, s_1, \ldots, s_n)$ of $n+1$ spines, each spine s_k being assigned to word w_k. Finally, labeling $\mathbf{l} = (l_1, \ldots, l_n)$ is a sequence where l_k is the label of the k^{th} arc (h, m) of $\mathbf{d}$. The label consists of a couple (op, i) where *op* is the type of adjunction, here *sister* or *regular*[2], and i is the index of the adjunction node in s_h.

Each derivation is assigned an arc-factored score σ which is given by:

$$\sigma(\mathbf{d}, \mathbf{s}, \mathbf{l}; \mathbf{w}) = \sum_{(h,m) \in \mathbf{d}} \Omega(h, m, s_h, s_m, l_{hm}; \mathbf{w})$$

For instance, following score functions developed in (Carreras et al., 2008), this function could read $s_h[i], s_h[i+1]$ and $s_m[0]$, where $s[i]$ denotes the i-th grammatical category of the spine s. The score of the derivation in Figure 1 could then reflect that the spine `WHNP-WP` associated with *What* is adjoined on the spine `SBARQ-SQ-VP-VB` associated with *do* on a site with the grammatical triple `[VP WHNP VB]`.

We assume that Ω accounts for the contribution of arcs, spines and labels to the score. The details of the contribution depend on the model. We choose the following:

$$\begin{aligned}\sigma(\mathbf{d}, \mathbf{s}, \mathbf{l}; \mathbf{w}) = \sum_{(h,m) \in \mathbf{d}} & (\alpha(h, m; \mathbf{w}) \\ & + \nu(s_m; h, m, \mathbf{w}) \\ & + \gamma(l_{hm}; h, m, s_h, \mathbf{w}))\end{aligned}$$

where α is the score related to the dependency tree, ν is the supertagging score and γ the labeling score. Note that functions α, ν and γ have access to the entire input string $\mathbf{w}$. Score function σ can be parameterized in many ways and we discuss our implementation in Section 5. In this setting, parsing a sentence $\mathbf{w}$ amounts to finding the highest-scoring derivation $(\mathbf{d}^*, \mathbf{s}^*, \mathbf{l}^*) = \arg\max_{(\mathbf{d},\mathbf{s},\mathbf{l})} \sigma(\mathbf{d}, \mathbf{s}, \mathbf{l}; \mathbf{w})$.

Recovering the derived tree from a derivation is performed by recursively mapping each spine and its dependencies to a possibly gappy constituent. Given a spine s_h and site index i, we look for the leftmost s_l and rightmost s_r dependents attached with regular adjunction. If any, we insert a new node between $s_h[i]$ and $s_h[i+1]$ with the same grammatical category as the first one. This new node fills the role of the foot node in TAGs. Every dependent of $s_h[i]$ with anchor in interval $[l+1, r-1]$ is moved to the newly created node. Remaining sister and regular adjunctions are simply attached to $s_h[i]$.

The complexity of the parsing problem depends on the type of dependency trees. In the case of projective trees, it has been shown (Eisner, 2000; Carreras et al., 2008; Li et al., 2011) that this could be performed in cubic worst-case time complexity with dynamic programming, whether supertags are fixed beforehand or not. However, the modification of the original Eisner algorithm requires that chart cells must be indexed not only by spans, or pairs of positions, but also by pairs of supertags. In practice the problem is intractable unless heavy

[2]The distinction is not crucial for the exposition. We refer readers to (Shen and Joshi, 2008; Carreras et al., 2008).

pruning is performed first in order to select a subset of spines at each position.

In the case of non-projective dependency trees, the problem has quadratic worst-case time complexity when supertags are fixed, since the problem then amounts to non-projective parsing and reduces to the Maximum Spanning Arborescence problem (MSA) as in (McDonald et al., 2005). Unfortunately, the efficient algorithm for MSA is greedy and does not *store* potential substructure candidates. Hence, when supertags are not fixed beforehand, a new arborescence must be recomputed for each choice of supertags. This problem can be seen as instance of the Generalized Maximum Spanning Arborescence problem, an NP-complete problem, which we review in the next section. Note that arc labels do not impact the asymptotic complexity of an arc-factored model. Indeed, only the labeled arc with maximum weight between two vertices is considered when parsing.

3 The Generalized Maximum Spanning Arborescence

In this section, we first define GMSA introduced by Myung et al. (1995). We formulate this problem as an integer linear program. We then explain the reduction from the joint supertagging and spine parsing task to this problem.[3]

3.1 Problem definition

Let $D = (V, A)$ be a directed graph. Given a subset $T \subseteq A$ of arcs, $V[T]$ denotes the set of vertices of V which are the tail or the head of at least one arc of T. These vertices are said to *be covered* by T. A subset $T \subseteq A$ of arcs is called an *arborescence* if the graph $(V[T], T)$ is connected, acyclic and each vertex has at most one entering arc. The vertex with no entering arc is called the *root* of T. An arborescence covering all vertices is called a *spanning arborescence*.

Let $\pi = \{V_0, \dots, V_n\}$, $n \in \mathbb{N}$ be a partition of V. Each element of π is called a *cluster*. An arborescence T of D covering exactly one vertex per cluster of π is called a *generalized spanning arborescence* (GSA). Figure 2 gives an example of a GSA. The partition of V is composed of a cluster having one vertex and six clusters having four vertices. Each cluster is depicted by a hatched area. The GSA is depicted by the dashed arcs.

[3]A similar reduction can be obtained in the reverse direction, thus proving the NP-completeness of our problem.

Let W be a vertex subset of V. We denote $\delta^-(W)$ (resp. $\delta^+(W)$) the set of arcs entering (resp. leaving) W and $\delta(W) = \delta^-(W) \cup \delta^+(W)$.[4] Contracting W consists in replacing in D all vertices in W by a new vertex w, replacing each arc $uv \in \delta^-(W)$ by the arc uw and each arc $vu \in \delta^+(W)$ by wu. Let D^π be the graph obtained by contracting each cluster of π in D. Note that a GSA of D and π induces a spanning arborescence of D^π.[5] For instance, contracting each cluster in the graph given by Figure 2 leads to a graph D^π having 7 vertices and the set of dashed arcs corresponds to a spanning arborescence of D^π.

Given arc weights $\phi \in \mathbb{R}^A$, the weight of an arborescence T is $\sum_{a \in T} \phi_a$. Given (D, π, ϕ), the *Generalized Maximum Spanning Arborescence problem (GMSA)* consists in finding a GSA of D and π of maximum weight whose root is in V_0.

3.2 Integer linear program

Given a set S, $z \in \mathbb{R}^S$ is a vector indexed by elements in S. For $S' \subseteq S$, $z(S') = \sum_{s \in S'} z_s$.

A GSA $T \subseteq A$ is represented by variables $x \in \{0,1\}^V$ and $y \in \{0,1\}^A$ such that x_v (resp. y_a) is equal to 1 iff $v \in V[T]$ (resp. $a \in T$).

Since a GSA of D and π induces a spanning arborescence of D^π, the arc-incidence vector $y \in \{0,1\}^A$ of a GSA with root in V_0 satisfies the following, adapted from MSA (Schrijver, 2003):

$$y(\delta^-(V_0)) = 0 \tag{1}$$

$$y(\delta^-(V_k)) = 1 \qquad \forall 1 \leq k \leq n, \tag{2}$$

$$y(\delta^-(\bigcup_{V_k \in \pi'} V_k)) \geq 1 \quad \forall \pi' \subseteq \pi \setminus \{V_0\}. \tag{3}$$

Let $\mathcal{Y}$ denote all the arc-incidence vectors on D corresponding to a spanning arborescence in D^π whose root is the contraction of V_0. Then,

$$\mathcal{Y} = \{y \in \{0,1\}^A | y \text{ satisfies (1)-(3)}\}.$$

GMSA can be formulated with the following integer linear program:

$$\max_{x,y} \quad \phi \cdot y \tag{4}$$

$$\text{s.t.} \quad y \in \mathcal{Y} \tag{5}$$

$$x_v \geq y_a \qquad \forall v \in V, a \in \delta(v), \tag{6}$$

$$x_v(V_k) = 1 \qquad \forall 0 \leq k \leq n, \tag{7}$$

$$x_v \in \{0,1\} \qquad \forall v \in V. \tag{8}$$

[4]By an abuse of notation, we identify any singleton $\{v\}$ with its element v.

[5]The converse does not hold: an arc subset of A corresponding to a spanning arborescence of D^π may not be a GSA of D and π since it may not induce a connected graph.

Let W and T be the vertex and arc sets given by $x_v = 1$ and $y_a = 1$ respectively. Since T is a spanning arborescence of D^π by (5), $(V[T], T)$ is an acyclic directed graph with n arcs such that V_0 has no entering arc and V_i, $i \in \{1, \ldots, n\}$, has one entering arc. By constraints (7), W contains one vertex per cluster of π. Moreover, by inequalities (6), $V[T] \subseteq W$. Since $|W| = n + 1$ and $|T| = n$, $W = V[T]$ and $(V[T], T)$ is connected, so it is a GSA. Because its root is in V_0 by (5), it is an optimal solution for GMSA by (4).

3.3 Reduction from joint parsing to GMSA

Given an instance of the joint parsing problem, we construct an instance of GMSA as follows. With every spine s of every word w_k different from w_0, we associate a vertex v. For $k = 1, \ldots, n$, we denote by V_k the set of vertices associated with the spines of w_k. We associate with w_0 a set V_0 containing only one vertex and V_0 will now refer both the cluster and the vertex it contains depending on the context. Let $\pi = \{V_0, \ldots, V_n\}$ and $V = \cup_{k=0}^n V_k$. For every couple u, v of vertices such that $u \in V_h$ and $v \in V_m$, $h \neq m$ and $m \neq 0$, we associate an arc uv corresponding to the best adjunction of the root of spine s_m associated with v of V_m to spine s_h associated with vertex u of V_h. The weight of this arc is given by

$$\phi_{uv} = \alpha(h, m; \mathbf{w}) + \nu(s_m; h, m, \mathbf{w}) + \max_{l_{hm}} \gamma(l_{hm}; h, m, s_h, \mathbf{w})$$

which is the score of the best adjunction of s_m to s_h. This ends the construction of (D, π, ϕ).

There is a 1-to-1 correspondence between the solutions to GMSA and those to the joint supertagging and spine parsing task in which each adjunction is performed with the label maximizing the score of the adjunction. Indeed, the vertices covered by a GSA T with root V_0 correspond to the spines on which the derivation is performed. By definition of GSAs, one spine per word is chosen. Each arc of T corresponds to an adjunction. The score of the arborescence is the sum of the scores of the selected spines plus the sum of the scores of the best adjunctions with respect to T. Hence, one can solve GMSA to perform joint parsing.

As an illustration, the GSA depicted in Figure 2 represents the derivation tree of Figure 1: the vertices of $V \setminus V_0$ covered by the GSA are those associated with the spines of Figure 1 and the arcs represent the different adjunctions. For instance the arc from V_3 to V_2 represents the adjunction of spine `NP-PRP` to spine `S-VP-VB` at index 0.

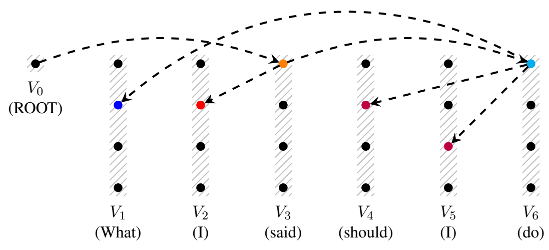

Figure 2: The generalized spanning arborescence inducing the derivation tree in Figure 1.

4 Efficient Decoding

Lagrangian relaxation has been successfully applied to various NLP tasks (Koo et al., 2010; Le Roux et al., 2013; Almeida and Martins, 2013; Das et al., 2012; Corro et al., 2016). Intuitively, given an integer linear program, it consists in relaxing some linear constraints which make the program difficult to solve and penalizing their violation in the objective function.

We propose a new decoding method for GMSA based on dual decomposition, a special flavor of Lagrangian relaxation where the problem is decomposed in several independent subproblems.

4.1 Dual decomposition

To perform the dual decomposition, we first reformulate the integer linear program (4)-(8) before relaxing linear constraints. For this purpose, we replace the variables y by three copies $\{y^i\} = \{y^0, y^1, y^2\}$, $y^i \in \{0, 1\}^A$. We also consider variables $z \in \mathbb{R}^A$. Let ϕ^0, ϕ^1 and ϕ^2 be arc weight vectors such that $\sum_i \phi^i = \phi$.[6] GMSA can then be reformulated as:

$$\max_{x,\{y^i\},z} \sum_i \phi^i \cdot y^i \tag{9}$$

$$\text{s.t.} \quad y^0 \in \mathcal{Y} \tag{10}$$

$$x_v \geq y^1_a \quad \forall v \in V, a \in \delta^-(v), \tag{11}$$

$$x_v \geq y^2_a \quad \forall v \in V, a \in \delta^+(v), \tag{12}$$

$$x_v(V_k) = 1 \quad \forall 0 \leq k \leq n, \tag{13}$$

$$x_v \in \{0, 1\} \quad \forall v \in V, \tag{14}$$

$$z = y^i \quad \forall i. \tag{15}$$

[6]In our implementation, we choose $\phi^0 = \phi^1 = \phi^2 = \frac{1}{3}\phi$.

Note that variables z only appear in equations (15). Their goal is to ensure equality between copies y^0, y^1 and y^2. Variables z are usually called *witness variables* (Komodakis et al., 2007). Equality between y^0, y^1 and y^2 implies that (10)-(12) are equivalent to (5) and (6).

We now relax constraints (15) and build the dual objective (Lemaréchal, 2001) $\mathcal{L}^*(\{\lambda^i\})$:

$$\max_{x,\{y^i\},z} \sum_i \phi^i \cdot y^i + \sum_{i\in\{0,1,2\}} \lambda^i \cdot (z - y^i)$$
$$\text{s.t.} \quad (10) - (14)$$

where $\{\lambda^i\} = \{\lambda^0, \lambda^1, \lambda^2\}$, $\lambda^i \in \mathbb{R}^A$ for $i = 0, 1, 2$, is the set of Lagrangian multipliers. The dual problem is then:

$$\min_{\{\lambda^i\}} \mathcal{L}^*(\{\lambda^i\})$$

Note that, as there is no constraint on z, if $\sum_i \lambda^i \neq \mathbf{0}$ then $\mathcal{L}^*(\{\lambda^i\}) = +\infty$. Therefore, we can restrict the domain of $\{\lambda^i\}$ in the dual problem to the set $\Lambda = \{\{\lambda^i\} | \sum_i \lambda^i = \mathbf{0}\}$. This implies that z may be removed in the dual objective. This latter can be rewritten as:

$$\mathcal{L}^*(\{\lambda^i\}) = \max_{x,\{y^i\}} \sum_i \bar{\phi}^i \cdot y^i$$
$$\text{s.t.} \quad (10) - (14)$$

where $\bar{\phi}^i = \phi^i - \lambda^i$ for $i = 0, 1, 2$.

4.2 Computing the dual objective

Given $\{\lambda^i\} \in \Lambda$, computing the dual objective $\mathcal{L}^*(\{\lambda^i\})$ can be done by solving the two following distinct subproblems:

$$\begin{aligned} P_1(\bar{\phi}^0) &= \max_{y^0} \bar{\phi}^0 \cdot y^0 \\ &\text{s.t.} \quad y^0 \in \mathcal{Y} \\ P_2(\bar{\phi}^1, \bar{\phi}^2) &= \max_{x,y^1,y^2} \bar{\phi}^1 \cdot y^1 + \bar{\phi}^2 \cdot y^2 \\ &\text{s.t.} \quad (11) - (14) \\ &\quad\quad y^i_a \in \{0,1\} \qquad \forall a \in A, i = 1, 2. \end{aligned}$$

Subproblem P_1 can be solved by simply running the MSA algorithm on the contracted graph D^π.

Subproblem P_2 can be solved in a combinatorial way. Indeed, observe that each value of y^1 and y^2 is only constrained by a single value of x. The problem amounts to selecting for each cluster a vertex as well as all the arcs with positive weight covering it. More precisely, for each vertex $v \in V$, compute the *local weight* c_v defined by:

$$\sum_{a\in\delta^-(v)} \max\{0, \bar{\phi}^1\} + \sum_{a\in\delta^+(v)} \max\{0, \bar{\phi}^2\}.$$

Let $V^{\max}$ be the set of vertices defined as follows. For $k = 0, \ldots, n$, add in $V^{\max}$ the vertex $v \in V_k$ with the maximum weight c_v. Let A^1 and A^2 be the sets of arcs such that A^1 (resp. A^2) contains all the arcs with positive weights entering (resp. leaving) a vertex of $V^{\max}$. The vectors x, y^1 and y^2 corresponding respectively to the incidence vectors of $V^{\max}$, A^1 and A^2 form an optimal solution to P_2.

Hence, both supbroblems can be be solved with a $O(|\pi|^2)$ time complexity, that is quadratic w.r.t. the length of the input sentence.[7]

4.3 Decoding algorithm

Our algorithm seeks for a solution to GMSA by solving the dual problem since its solution is optimal to GMSA whenever it is a GSA. If not, a solution is constructed by returning the highest GSA on the spines computed during the resolution of the dual problem.

We solve the dual problem using a projected subgradient descent which consists in iteratively updating $\{\lambda^i\}$ in order to reduce the distance to the optimal assignment. Let $\{\lambda^{i,t}\}$ denotes the value of $\{\lambda^i\}$ at iteration t. $\{\lambda^{i,0}\}$ is initially set to $\mathbf{0}$. At each iteration, the value of $\{\lambda^{i,t+1}\}$ is computed from the value of $\{\lambda^{i,t}\}$ thanks to a subgradient of the dual objective. More precisely, we have

$$\{\lambda^{i,t+1}\} = \{\lambda^{i,t}\} - \eta^t \times \nabla\mathcal{L}^*(\{\lambda^{i,t}\})\}$$

where $\nabla\mathcal{L}^*(\{\lambda^{i,t}\})$ is a subgradient of $\mathcal{L}^*(\{\lambda^{i,t}\})$ and $\eta^t \in \mathbb{R}$ is the stepsize at iteration t. We use the projected subgradient from Komodakis et al. (2007). Hence, at iteration t, we must solve reparameterized subproblems P_1 and P_2 to obtain the current solution $(\bar{x}^t, \bar{y}^{0,t}, \bar{y}^{1,t}, \bar{y}^{2,t})$ of the dual objective. Then each multiplier is updated following

$$\lambda^{i,t+1} = \lambda^{i,t} - \eta^t \times \left(\bar{y}^{i,t} - \sum_{j=0}^{2} \frac{\bar{y}^{j,t}}{3} \right).$$

Note that for any value of $\{\lambda^i\}$, $\mathcal{L}^*(\{\lambda^i\})$ gives an upper bound for GMSA. So, whenever the

[7]In the general case, the time complexity is $O(|V|^2)$. But in our problem, the number of vertices per cluster is bounded by the grammar size: $O(|V|^2) = O(|S\pi|^2) = O(|\pi|^2)$.

optimal solution $\bar{x}^t, \{\bar{y}^{i,t}\}$ to the dual objective $\mathcal{L}^*(\{\lambda^{i,t}\})$ at iteration t is a primal feasible solution, that is $\bar{y}^{0,t} = \bar{y}^{1,t} = \bar{y}^{2,t}$, it is an optimal solution to GMSA and the algorithm ends. Otherwise, we construct a *pipeline solution* by performing a MSA on the vertices given by $\bar{x}^t$.

If after a fixed number of iterations we have not found an optimal solution to GMSA, we return the pipeline solution with maximum weight.

4.4 Lagrangian enhancement

The previsouly defined Lagrangian dual is valid but may lead to slow convergence. Thus, we propose three additional techniques which empirically improve the decoding time and the convergence rate: constraint tightening, arc reweighing and problem reduction.

Constraint tightening: In subproblem P_2, we consider a vertex and all of its adjacent arcs of positive weight. However, we know that our optimal solution must satisfy tree-shape constraints (5). Thus, every cluster except the root must have exactly one incoming arc and there is at most one arc between two clusters. Both constraints are added to P_2 without hurting its time complexity.

Reweighing: By modifying weights such that less incoming arcs have a positive weight, the solution of P_2 tends to be an arborescence. For each cluster $V_k \in \pi \setminus V_0$, let $\hat{A}_k$ be the set of incoming arcs with the highest weight $\hat{\phi}_k$. Then, let γ_k be a value such that $\phi_a - \gamma_k$ is positive only for arcs in $\hat{A}_k$. Subtracting γ_k from the weight ϕ_a of each arc of $\delta^-(V_k)$ and adding γ_k to the objective score does not modify the weight of the solution because only one entering arc per cluster is selected.

Problem reduction: We use the pipeline solutions computed at each iteration to set the value of some variables. Let $\bar{x}, \{\bar{y}^i\}$ be the optimal solution of $\mathcal{L}^*(\{\lambda^i\})$ computed at any iteration of the subgradient algorithm. For $k = 1, \ldots, n$, let $\bar{v}$ be the vertex of V_k such that $\bar{x}_{\bar{v}} = 1$. Using the local weights (Section 4.2), for all $v \in V_k \setminus \{\bar{v}\}$, $\mathcal{L}^*(\{\lambda^i\}) + c_v - c_{\bar{v}}$ is an upper bound on the weight of any solution (x, y) to GMSA with $x_v = 1$. Hence, if it is lower than the weight of the best pipeline solution found so far, we can guarantee that $x_v = 0$ in any optimal solution. We can check the whole graph in linear time if we keep local weights c in memory.

5 Neural Parameterization

We present a probabilistic model for our framework. We implement our probability distributions with neural networks, more specifically we build a neural architecture on top of bidirectional recurrent networks that compute context sensitive representations of words. At each step, the recurrent architecture is given as input a concatenation of word and part-of-speech embeddings. We refer the reader to (Kiperwasser and Goldberg, 2016; Dozat and Manning) for further explanations about bidirectional LSTMs (Hochreiter and Schmidhuber, 1997). In the rest of this section, b_m denotes the context sensitive representation of word w_m.

We now describe the neural network models used to learn and assign weight functions α, ν and γ under a probabilistic model. Given a sentence $\mathbf{w}$ of length n, we assume a derivation $(\mathbf{d}, \mathbf{s}, \mathbf{l})$ is generated by three distinct tasks. By chain rule, $P(\mathbf{d}, \mathbf{s}, \mathbf{l}|\mathbf{w}) = P_\alpha(\mathbf{d}|\mathbf{w}) \times P_\nu(\mathbf{s}|\mathbf{d}, \mathbf{w}) \times P_\gamma(\mathbf{l}|\mathbf{d}, \mathbf{s}, \mathbf{w})$. We follow a common approach in dependency parsing and assign labels $\mathbf{l}$ in a post-processing step, although our model is able to incorporate label scores directly. Thus, we are left with jointly decoding a dependency structure and assigning a sequence of spines. We note s_i the i^{th} spine:[8]

$$\begin{aligned}
&P_\alpha(\mathbf{d}|\mathbf{w}) \times P_\nu(\mathbf{s}|\mathbf{d}, \mathbf{w}) \\
&= \prod_{(h,m) \in \mathbf{d}} P_\alpha(h|m, \mathbf{w}) \times P_\nu(s_m|m, \mathbf{d}, \mathbf{w}) \\
&= \prod_{(h,m) \in \mathbf{d}} P_\alpha(h|m, \mathbf{w}) \times P_\nu(s_m|m, h, \mathbf{w})
\end{aligned}$$

We suppose that adjunctions are generated by an arc-factored model, and that a spine prediction depends on both current position and head position.

Then parsing amounts to finding the most probable derivation and can be realized in the log space, which gives following weight functions:

$$\begin{aligned}
\alpha(h, m; \mathbf{w}) &= \log P_\alpha(h|m, \mathbf{w}) \\
\nu(s_m; h, m, \mathbf{w}) &= \log P_\nu(s_m|m, h, \mathbf{w})
\end{aligned}$$

where α represents the arc contribution and ν the spine contribution (cf. Section 2).

Word embeddings b_k are first passed through specific feed-forward networks depending on the

[8] We assume that the spine for the root w_0 is unique.

distribution and role. The result of the feed-forward transformation parameterized by set of parameters ρ of a word embedding b_s is a vector denoted $b_s^{(\rho)}$. We first define a biaffine attention networks weighting dependency relations (Dozat and Manning):

$$o^{(\alpha)}_{h,m} = b^{(\alpha_1)\top}_m W^{(\alpha)} b^{(\alpha_2)}_h + V^{(\alpha)} b^{(\alpha_2)}_h$$

where $W^{(\alpha)}$ and $V^{(\alpha)}$ are trainable parameters. Moreover, we define a biaffine attention classifier networks for class c as:

$$\begin{aligned} o^{(\tau)}_{c,h,m} = {} & b^{(\tau_1)\top}_m W^{(\tau_c)} b^{(\tau_2)}_h \\ & + V^{(\tau_c)} \left(b^{(\tau_1)}_m \oplus b^{(\tau_2)}_h \right) \\ & + u^{(\tau_c)} \end{aligned}$$

where $\oplus$ is the concatenation. $W^{(\tau_c)}$, $V^{(\tau_c)}$ and $u^{(\tau_c)}$ are trainable parameters. Then, we define the weight of assigning spine s to word at position m with head h as $o^{(\nu)}_{s,h,m}$.

Distributions P_α and P_ν are parameterized by these biaffine attention networks followed by a softmax layer:

$$P_\alpha(h|m, \mathbf{w}) = \frac{\exp o^{(\alpha)}_{h,m}}{\sum_{h'} \exp o^{(\alpha)}_{h',m}}$$

$$P_\nu(s|h, m, \mathbf{w}) = \frac{\exp o^{(\nu)}_{s,h,m}}{\sum_{s'} \exp o^{(\nu)}_{s',h,m}}$$

Now we move on to the post-processing step predicting arc labels. For each adjunction of spine s at position m to spine t at position h, instead of predicting a site index i, we predict the non-terminal nt at $t[i]$ with a biaffine attention classifier.[9] The probability of the adjunction of spine s at position m to a site labeled with nt on spine t at position h with type $a \in \{\text{regular}, \text{sister}\}$ is:

$$\begin{aligned} P_\gamma(nt, a|h, m) = {} & P_{\gamma'}(nt|h, m, \mathbf{w}) \\ & \times P_{\gamma''}(a|h, m\mathbf{w}) \end{aligned}$$

P_γ and $P_{\gamma''}$ are again defined as distributions from the exponential family using biaffine attention classifiers:

$$P_{\gamma'}(nt|h, m, t) = \frac{\exp o^{(\gamma')}_{nt,h,m}}{\sum_{nt'} \exp o^{(\gamma')}_{nt,h,m}}$$

$$P_{\gamma''}(a|h, m, t) = \frac{\exp o^{(\gamma'')}_{t,h,m}}{\sum_{a'} \exp o^{(\gamma'')}_{a',h,m}}$$

[9]If a spine contains repeated non-terminal sequences, we select the lowest match.

We use embeddings of size 100 for words and size 50 for parts-of-speech tags. We stack two bidirectional LSTMs with a hidden layer of size 300, resulting in a context sensitive embedding of size 600. Embeddings are shared across distributions. All feed-forward networks have a unique elu-activated hidden layer of size 100 (Clevert et al., 2016). We regularize parameters with a dropout ratio of 0.5 on LSTM input. We estimate parameters by maximizing the likelihood of the training data through stochastic subgradient descent using Adam (Kingma and Ba, 2015). Our implementation uses the Dynet library (Neubig et al., 2017) with default parameters.

6 Experiments

We ran a series of experiments on two corpora annotated with discontinuous constituents.

English We used an updated version of the Wall Street Journal part of the Penn Treebank corpus (Marcus et al., 1994) which introduces discontinuity (Evang and Kallmeyer, 2011). Sections 2-21 are used for training, 22 for developpement and 23 for testing. We used gold and predicted POS tags by the Stanford tagger,[10] trained with 10-jackknifing. Dependencies are extracted following the head-percolation table of Collins (1997).

German We used the Tiger corpus (Brants et al., 2004) with the split defined for the SPMRL 2014 shared task (Maier, 2015; Seddah et al., 2013). Following Maier (2015) and Coavoux and Crabbé (2017), we removed sentences number 46234 and 50224 as they contain annotation errors. We only used the given gold POS tags. Dependencies are extracted following the head-percolation table distributed with Tulipa (Kallmeyer et al., 2008).

We emphasize that long sentences are not filtered out. Our derivation extraction algorithm is similar to the one proposed in Carreras et al. (2008). Regarding decoding, we use a beam of size 10 for spines w.r.t. $P_\nu(s_m|m, \mathbf{w}) = \sum_h P_\nu(s_m|h, m, \mathbf{w}) \times P_\alpha(h|m, \mathbf{w})$ but allow every possible adjunction. The maximum number of iterations of the subgradient descent is set to 500 and the stepsize η^t is fixed following the rule of Polyak (1987).

Parsing results and timing on short sentences only ($\leq$ 40 words) and full test set using the de-

[10]http://nlp.stanford.edu/software/tagger.shtml

fault discodop[11] eval script are reported on Table 1 and Table 2.[12] We report labeled recall (LR), precision (LP), F-measure (LF) and time measured in minutes. We also report results published by van Cranenburgh et al. (2016) for the discontinuous PTB and Coavoux and Crabbé (2017) for Tiger. Moreover, dependency unlabeled attachment scores (UAS) and tagging accuracies (Spine acc.) are given on Table 3. We achieve significantly better results on the discontinuous PTB, while being roughly 36 times faster together with a low memory footprint.[13] On the Tiger corpus, we achieve on par results. Note however that Coavoux and Crabbé (2017) rely on a greedy parser combined with beam search.

Fast and efficient parsing of discontinuous constituent is a challenging task. Our method can quickly parse the whole test set, without any parallelization or GPU, obtaining an optimality certificate for more than 99% of the sentences in less than 500 iterations of the subgradient descent. When using a non exact decoding algorithm, such as a greedy transition based method, we may not be able to deduce the best opportunity for improving scores on benchmarks, such as the parameterization method or the decoding algorithm. Here the behavior may be easier to interpret and directions for future improvement easier to see. We stress that our method is able to produce an optimality certificate on more than 99% of the test examples thanks to the enhancement presented in Section 4.4.

	LR	LP	LF	Time
	Short sentences only			
This work	90.63	91.01	90.82	≈ 4
This work†	89.57	90.13	89.85	≈ 4
VC2016†			87.00	≈ 180
	Full test set			
This work	89.89	90.29	90.09	≈ 6.5
This work†	88.90	89.45	89.17	≈ 5.5

Table 1: Parsing results and processing time on the english discontinuous PTB corpus. Results marked with † use predicted part-of-speech tags. VC2016 indicates results of van Cranenburgh et al. (2016).

	LR	LP	LF	Time
	Short sentences only			
This work	82.69	84.68	83.67	≈ 7.5
	Full test set			
This work	80.66	82.63	81.63	≈ 11
C2017			81.60	≈ 2.5

Table 2: Parsing results and processing time on the german Tiger corpus. C2017 indicates results of Coavoux and Crabbé (2017).

7 Related Work

Spine-based parsing has been investigated in (Shen and Joshi, 2005) for Lexicalized TAGs with a left-to-right shift-reduce parser which was subsequently extended to a bidirectional version in (Shen and Joshi, 2008). A graph-based algorithm was proposed in (Carreras et al., 2008) for second-order projective dependencies, and for a form of non-projectivity occurring in machine translation (i.e. projective parses of permutated input sentences) in (Carreras and Collins, 2009).

Discontinuous phrase-structure parsing through dependencies in contexts other that TAGs have been explored in (Hall and Nivre, 2008; Versley, 2014; Fernández-González and Martins, 2015). The first two encode spine information as arc labels while the third one relaxes spine information by keeping only the root and height of the adjunction, thus avoiding combinatorial explosion. Labeling is performed as a post-processing step in these approaches, since the number of labels can be very high. Our model also performs labeling after structure construction, but it could be performed jointly without major issue. This is one way our model could be improved.

GMSA has been studied mostly as a way to solve the non directed version (i.e. with symetric arc weights) (Myung et al., 1995), see (Pop, 2009; Feremans et al., 1999) for surveys on resolution methods. Myung et al. (1995) proposed an exact decoding algorithm through branch-and-bound using a dual ascent algorithm to compute bounds. Pop (2002) also used Lagrangian relaxation – in the non directed case – where a single subproblem is solved in polynomial time. However, the relaxed constraints are inequalities: if the dual objective returns a valid primal solution, it is not a sufficient condition in order to guarantee that

[11]https://github.com/andreasvc/disco-dop/

[12]C2017 processing time is 137.338 seconds plus approximatively 30 seconds for model and corpus loading (personnal communication).

[13]Execution times are not directly comparable because we report our experimental conditions and published results.

	UAS	Spine acc.
English	93.70	97.32
English†	93.04	96.81
German	92.25	96.49

Table 3: Dependency parsing and tagging results. Results marked with † use predicted part-of-speech tags.

it is the optimal solution (Beasley, 1993), and thus the stopping criterion for the subgradient descent is usually slow to obtain. To our knowledge, our system is the first time that GMSA is used to solve a NLP problem.

Dual decomposition has been used to derive efficient practical resolution methods in NLP, mostly for machine translation and parsing, see (Rush et al., 2010) for an overview and (Koo et al., 2010) for an application to dependency parsing.

To accelerate the resolution, our method relies heavily on problem reduction (Beasley, 1993), which uses the primal/dual bounds to filter out suboptimal assignments. Exact pruning based on duality has already been studied in parsing, with branch and bound (Corro et al., 2016) or column generation (Riedel et al., 2012) and in machine translation with beam search (Rush et al., 2013).

8 Conclusion

We presented a novel framework for the joint task of supertagging and parsing by a reduction to GMSA. Within this framework we developed a model able to produce discontinuous constituents. The scoring model can be decomposed into tagging and dependency parsing and thus may rely on advances in those active fields.

This work could benefit from several extensions. Bigram scores on spines could be added at the expense of a third subproblem in the dual objective. High-order scores on arcs like grandparent or siblings can be handled in subproblem P_2 with the algorithms described in (Koo et al., 2010). In this work, the parameters are learned as separate models. Joint learning in the max-margin framework (Komodakis, 2011; Komodakis et al., 2015) may model interactions between vertex and arc weights better and lead to improved accuracy. Finally, we restricted our grammar to spinal trees but it could be possible to allow full lexicalized TAG-like trees, with substitution nodes and even obligatory adjunction sites. Derivations compatible with the TAG formalism (or more generally LCFRS) could be recovered by the use of a constrained version of MSA (Corro et al., 2016).

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We thank Laura Kallmeyer and Kilian Evang for providing us with the script for the discontinuous PTB. First author is supported by a public grant overseen by the French National Research Agency (ANR) as part of the Investissements d'Avenir program (ANR-10-LABX-0083). Second author, supported by a public grant overseen by the French ANR (ANR-16-CE33-0021), completed this work during a CNRS research leave at LIMSI, CNRS / Université Paris Saclay.

References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 196–206, Sofia, Bulgaria. Association for Computational Linguistics.

John Beasley. 1993. *Modern heuristic techniques for combinatorial problems*, chapter Lagrangian relaxation. John Wiley & Sons, Inc.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Xavier Carreras and Michael Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209, Singapore. Association for Computational Linguistics.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16, Manchester, England. Coling 2008 Organizing Committee.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the 2016 International Conference on Learning Representations*.

Maximin Coavoux and Benoit Crabbé. 2017. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, Mathieu Lacroix, Antoine Rozenknop, and Roberto Wolfler Calvo. 2016. Dependency parsing with bounded block degree and well-nestedness via lagrangian relaxation and branch-and-bound. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 355–366, Berlin, Germany. Association for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *The First Joint Conference on Lexical and Computational Semantics*, pages 209–217, Montréal, Canada. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *Proceedings of the 2017 International Conference on Learning Representations*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Corinne Feremans, Martine Labbé, and Gilbert Laporte. 1999. The generalized minimum spanning tree: Polyhedra and branch-and-cut. *Electronic Notes in Discrete Mathematics*, 3:45–50.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In *Advances in Natural Language Processing: 6th International Conference, GoTAL 2008 Gothenburg, Sweden, August 25-27, 2008 Proceedings*, pages 169–180, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Laura Kallmeyer, Timm Lichte, Wolfgang Maier, Yannick Parmentier, Johannes Dellert, and Kilian Evang. 2008. Tulipa: Towards a multi-formalism parsing environment for grammar engineering. In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 1–8, Manchester, England. Coling 2008 Organizing Committee.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of The International Conference on Learning Representations (ICLR)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Nikos Komodakis. 2011. Efficient training for pairwise or higher order crfs via dual decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1841–1848. IEEE.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

Nikos Komodakis, Bo Xiang, and Nikos Paragios. 2015. A framework for efficient structured max-margin learning of high-order mrf models. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1425–1441.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. Association for Computational Linguistics.

Joseph Le Roux, Antoine Rozenknop, and Jennifer Foster. 2013. Combining PCFG-LA models with dual decomposition: A case study with function labels and binarization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1158–1169, Seattle, Washington, USA. Association for Computational Linguistics.

Claude Lemaréchal. 2001. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HLT'94: Proceedings of the workshop on Human Language Technology*, pages 114–119, Morristown, NJ, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. 1995. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Boris T Polyak. 1987. Introduction to optimization. *Optimization Software*.

Petrica Claudiu Pop. 2002. *The generalized minimum spanning tree problem*. Twente University Press.

Petrica Claudiu Pop. 2009. A survey of different integer programming formulations of the generalized minimum spanning tree problem. *Carpathian Journal of Mathematics*, 25(1):104–118.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut—delayed column and row generation for graph based parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 732–743, Jeju Island, Korea. Association for Computational Linguistics.

Alexander Rush, Yin-Wen Chang, and Michael Collins. 2013. Optimal beam search for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 210–221, Seattle, Washington, USA. Association for Computational Linguistics.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Cambridge, MA. Association for Computational Linguistics.

A. Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, D. Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Villemonte Eric de la Clergerie. 2013. *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, chapter Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. Association for Computational Linguistics.

Libin Shen. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania.

Libin Shen and Aravind Joshi. 2005. Incremental ltag parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 811–818, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Libin Shen and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu, Hawaii. Association for Computational Linguistics.

Yannick Versley. 2014. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland. Dublin City University.

Predicting Lagrangian Multipliers for Mixed Integer Linear Programs

Francesco Demelas [1] **Joseph Le Roux** [1] **Mathieu Lacroix** [1] **Axel Parmentier** [2]

Abstract

Lagrangian Relaxation stands among the most efficient approaches for solving Mixed Integer Linear Programs (MILPs) with difficult constraints. Given any duals for these constraints, called Lagrangian Multipliers (LMs), it returns a bound on the optimal value of the MILP, and Lagrangian methods seek the LMs giving the best such bound. But these methods generally rely on iterative algorithms resembling gradient descent to maximize the concave piecewise linear dual function: the computational burden grows quickly with the number of relaxed constraints. We introduce a deep learning approach that bypasses the descent, effectively amortizing *per instance* optimization. A probabilistic encoder based on a graph neural network computes, given a MILP instance and its Continuous Relaxation (CR) solution, high-dimensional representations of relaxed constraints, which are turned into LMs by a decoder. We train the encoder and the decoder jointly by directly optimizing the bound obtained from the predicted multipliers. Our method is applicable to any problem with a compact MILP formulation, and to any Lagrangian Relaxation providing a tighter bound than CR. Experiments on two widely known problems, Multi-Commodity Network Design and Generalized Assignment, show that our approach closes up to 85 % of the gap between the continuous relaxation and the best Lagrangian bound, and provides a high-quality warm-start for descent-based Lagrangian methods.

[1]Laboratoire d'Informatique de Paris-Nord, Université Sorbonne Paris Nord — CNRS, France [2]CERMICS, École des Ponts, France. Correspondence to: Francesco Demelas <demelas@lipn.fr>, Joseph Le Roux <leroux@lipn.fr>, Mathieu Lacroix <lacroix@lipn.fr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Mixed Integer Linear Programs (MILPs) (Wolsey, 2021) have two main strengths that make them ubiquitous in combinatorial optimization (Korte & Vygen, 2012). First, they can model many combinatorial optimization problems. Second, extremely efficient solvers can now handle MILPs with millions of constraints and variables. They therefore have a wide variety of applications in logistics, telecommunications and beyond. MILP algorithms are exact: they return an optimal solution, or an optimality gap between the returned solution and an optimal one.

MILPs are sometimes hard to solve due to a collection of difficult constraints. Typically, a small number of constraints link together otherwise independent subproblems. For instance, in vehicle routing problems (Golden et al., 2008), there is one independent problem for each vehicle, except for the linking constraints that ensure that exactly one vehicle operates each task of interest. Lagrangian relaxation approaches are popular in such settings as they allow to unlink the different subproblems.

More formally (Conforti et al., 2014, Chap. 8), let P be a MILP of the form:

$$(P) \qquad \min_{\boldsymbol{x}} \ \boldsymbol{w}^\top \boldsymbol{x} \tag{1a}$$

$$\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} \tag{1b}$$

$$\boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{d} \tag{1c}$$

$$\boldsymbol{x} \in \mathbb{R}^m_+ \times \mathbb{N}^p \tag{1d}$$

While CR amounts to simply removing the integrity constraints (*i.e.* (1d) becomes $\boldsymbol{x} \in \mathbb{R}^{m+p}_+$), the relaxed Lagrangian problem is obtained by dualizing difficult constraints (1b) and penalizing their violation with Lagrangian multipliers (LMs) $\boldsymbol{\pi} \geq \boldsymbol{0}$:

$$(LR(\boldsymbol{\pi})) \qquad \min_{\boldsymbol{x}} \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{\pi}^\top(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})$$

$$\boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{d}$$

$$\boldsymbol{x} \in \mathbb{R}^m_+ \times \mathbb{N}^p$$

Standard weak Lagrangian duality ensures that $LR(\boldsymbol{\pi})$ is a lower bound on P. The Lagrangian dual problem aims at finding the best such bound:

$$(LD) \qquad \max_{\boldsymbol{\pi} \geq \mathbf{0}} LR(\boldsymbol{\pi}).$$

Geoffrion's theorem (1974) ensures that LD is a lower bound at least as tight as the continuous relaxation. It is strictly better on most applications. Beyond this bound, Lagrangian approaches are also useful to find good primal solutions. Indeed, Lagrangian heuristics (Beasley, 1990) exploit the dual solution $\boldsymbol{\pi}$ and the variable assignment for $\boldsymbol{x}$ of $LR(\boldsymbol{\pi})$ to compute good quality solutions of (1a)-(1d). Note that both the bound and the heuristic hold even in the case of non-optimal duals $\boldsymbol{\pi}$. We define *good* Lagrangian duals $\boldsymbol{\pi}$ as those that lead to a bound $LR(\boldsymbol{\pi})$ better than the CR solution, and thus closer to LD.

Since $\boldsymbol{\pi} \mapsto LR(\boldsymbol{\pi})$ is piecewise linear and concave, it is generally optimized using a subgradient algorithm. Unfortunately, the number of iterations required to obtain good duals quickly increases with the dimension of $\boldsymbol{\pi}$, which makes the approach extremely intensive computationally.

In this work[1] we introduce a state-of-the-art encoder-decoder neural network that computes *good* duals $\boldsymbol{\pi}$ from the CR solution. The probabilistic encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\iota)$, based on a graph neural network (GNN), takes as input a MILP instance ι as well as the primal and dual CR solutions, and returns an embedding of the instance, where each dualized constraint is mapped to a high-dimensional dense vector. The deterministic decoder $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ reconstructs single dimensional duals from constraint vectors. The learning objective is unsupervised since the Lagrangian dual function $LR(\boldsymbol{\pi})$ leads to a natural loss function that does not require gold references. Experiments on two standard and widely used problems from the Combinatorial Optimization literature, Multi-Commodity Network Design and General Assignment, show that the predicted duals close up to 85% of the gap between the CR and LD solutions. Finally, when optimal duals are the target, we show that predicted duals provide an excellent warm-start for state-of-the-art descent-based algorithms for objective (1). Our approach is restricted to compact MILPs and Lagrangian Relaxations admitting a tighter bound than CR since primal and dual CR solutions are part of the GNN input.

2. Learning Framework

2.1. Overall Architecture

Iterative algorithms for setting LMs to optimality such as the subgradient method (SM) (Polyak, 1987, Chap 5.3) or the Bundle method (BM) (Hiriart-Urruty & Lemaréchal, 1996; Le et al., 2007) start by initializing LMs. They can be set to zero but a solution considered as better in practice by the Combinatorial Optimization community is to take advantage of the bound given by CR and its dual solution, often computationally cheap for compact MILPs. Specifically, optimal values of the CR dual variables identified with the constraints dualized in the Lagrangian relaxation can be understood as LMs. In many problems of interest these LMs are not optimal and can be improved by SM or BM. We leverage this observation by trying to predict a deviation from the LMs corresponding to the CR dual solution.

The architecture is depicted in Figure 1. We start from an input instance ι of MILP P with a set of constraints for which the Lagrangian relaxed problem is easy to compute, then solve CR and obtain the corresponding primal and dual solutions. The input enriched with CR solutions is then passed through a probabilistic encoder, composed of three parts: *(i)* the input is encoded as a bipartite graph in a way similar to (Gasse et al., 2019), also known as a *factor graph* in probabilistic modelling, and initial graph node feature extraction is performed, *(ii)* this graph is fed to a GNN in charge of refining the node features by taking into account the structure of the MILP, *(iii)* the last layer of the GNN is used to parameterize a distribution from which vectors $\boldsymbol{z}_c$ can be sampled for each dualized constraint c.

The decoder then translates $\boldsymbol{z}_c$ to a positive LM $\pi_c = \lambda_c + \delta_c$ by predicting a deviation δ_c from the CR dual solution variable λ_c. Finally, the predicted LMs can be used in several ways, in particular to compute a Lagrangian bound or to warm-start an iterative solver.

2.2. Objective

We train the network's parameters in an end-to-end fashion by maximizing the average Lagrangian bound $LR(\boldsymbol{\pi})$ obtained from the predicted LMs $\boldsymbol{\pi}$ over a training set. This can be cast as an empirical risk optimization, or an Energy-Based Model (Le Cun et al., 2006) with latent variables, where the Lagrangian bound is the (negative) energy corresponding to the coupling of the instance with the subproblem solutions, and the LMs — or more precisely their high-dimensional representations — the latent variables. For our problem, a natural measure of the quality of the prediction is provided by the value LR that we want to maximize to tighten the duality gap. Given an instance ι we want to learn to predict the latent representations $\boldsymbol{z}$ of the LMs for which the Lagrangian bound is the highest:

$$\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\cdot|\iota)} \left[LR([\boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})]_+; \iota) \right]$$

where $q_{\boldsymbol{\phi}}$ is the probabilistic encoder, mapping each dualized constraint c in ι to a latent vector $\boldsymbol{z}_c$ computed by independent Gaussian distributions, $f_{\boldsymbol{\theta}}$ is the decoder map-

[1]Code in JULIA at https://github.com/FDemelas/Learning_Lagrangian_Multipliers.jl

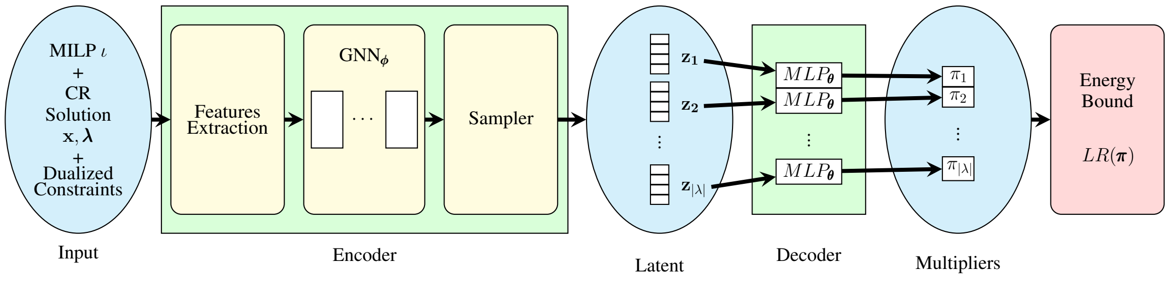

Figure 1. Overall Architecture. From the bipartite graph representation of a MILP and its CR solution, the model computes a Lagrangian dual solution. First the MILP is encoded by a GNN, from which we parameterize a sampler for constraint representations. These representations are then passed through a decoder to compute Lagrangian Multipliers.

ping each[2] $\boldsymbol{z}_c$ to its corresponding LM deviation δ_c from the CR dual value λ_c, and $[\cdot]_+$ is the component-wise soft-plus function. We can observe that this objective has the following properties amenable to gradient-based learning:

1. $LR(\boldsymbol{\pi})$ is bounded from above: optimal LMs $\boldsymbol{\pi}^*$ maximize $LR(\boldsymbol{\pi})$ over all possible LMs, that is $LR(\boldsymbol{\pi}^*) \geq LR(\boldsymbol{\pi})$ for any $\boldsymbol{\pi} = \boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})$. Moreover, $LR(\boldsymbol{\pi})$ is a concave piece-wise linear function, in other words all optimal solutions will give the same bound.

2. It is straightforward to compute a subgradient w.r.t. to parameters $\boldsymbol{\theta}$: $\nabla_{\boldsymbol{\theta}} LR([\boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})]_+; \iota)$ is equal to:

$$\left(\frac{\partial[\boldsymbol{\lambda} + f_{\boldsymbol{\theta}}(\boldsymbol{z})]_+}{\partial \boldsymbol{\theta}}\right)^\top \nabla_{\boldsymbol{\pi}} LR(\boldsymbol{\pi}; \iota)$$

   The Jacobian on the left is computed via backpropagation, while $LR(\boldsymbol{\pi}; \iota)$ is simple enough for a subgradient to be given analytically. Provided that $\bar{\boldsymbol{x}}$ is an optimal solution of the relaxed Lagrangian problem of ι associated with $\boldsymbol{\pi}$, we derive:

$$\nabla_{\boldsymbol{\pi}} LR(\boldsymbol{\pi}; \iota) = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}$$

   This means that in order to compute a subgradient for $\boldsymbol{\theta}$, we first need to solve each subproblem. Since subproblems are independent, this can be done in parallel.

3. For parameters $\boldsymbol{\phi}$, we again leverage function composition and the fact that $q_{\boldsymbol{\phi}}$ is a Gaussian distribution, so we can approximate the expectation by sampling and use the reparameterization trick (Kingma & Welling, 2014; Schulman et al., 2015) to perform standard backpropagation. We implement $q_{\boldsymbol{\phi}}$ as a neural network, described in details in the following section, returning a mean vector and a variance vector for each dualized constraint c, from which a sampler returns a representation vector $\boldsymbol{z}_c$. For numerical stability, the variance is clipped to a safe interval (Rybkin et al., 2021).

[2]With a slight abuse of notation, we use function $f : \mathbb{R}^m \to \mathbb{R}^n$ on *batches* of size p to become $\mathbb{R}^{m \times p} \to \mathbb{R}^{n \times p}$.

2.3. Encoding and Decoding Instances

Encoder One of the challenges in Machine Learning applications to Combinatorial Optimization is that instances have different input sizes, and so the encoder must be able to cope with these variations to produce high-quality features. Of course this is also the case in many other applications, for instance NLP where texts may differ in size, but there is no general consensus as to what a good feature extractor for MILP instances looks like, contrarily to other domains where variants of RNNs or Transformers have become the de facto standard encoders.

We depart from previous approaches to Lagrangian prediction (Sugishita et al., 2024) restricted to instances of the same size, and follow more generic approaches to MILP encoding such as (Gasse et al., 2019; Nair et al., 2020; Khalil et al., 2017) where each instance is converted into a bipartite graph and further encoded by GNNs to compute meaningful feature vectors associated with dualized constraints. Each MILP is converted to a bipartite graph composed of one node for each variable and one node for each constraint. There is an edge between a variable node n_v and a constraint node n_c if and only if v appears in c. Each node (variable or constraint) is represented by an initial feature vector $\boldsymbol{e}_n$. We use features similar to ones given in (Gasse et al., 2019).[3] Following Nair et al. (2020), variables and constraints are encoded as the concatenation of variable features followed by constraint features, of which only one is non-zero, depending on the type of nodes.

To design our stack of GNNs, we take inspiration from structured prediction models for images and texts, where Transformers (Vaswani et al., 2017) are ubiquitous. However, since our input has a bipartite graph structure, we replace the multihead self-attention layers with simple linear graph convolutions[4] (Kipf & Welling, 2017). Closer to our work, we follow Nair et al. (2020) which showed

[3]See Appendix A for more details.

[4]Alternatively, this can be seen as a masked attention, where the mask is derived from the input graph adjacency matrix.

that residual connections (He et al., 2016), dropout (Srivastava et al., 2014) and layer normalization (Ba et al., 2016) are important for the successful implementation of feature extractors for MILP bipartite graphs.

Before the actual GNNs, initial feature vectors $\{\boldsymbol{e}_n\}_n$ are passed through a MLP F to find feature combinations and extend node representations to high-dimensional spaces: $\boldsymbol{h}_n^0 = F(\boldsymbol{e}_n), \forall n$. Then interactions between nodes are taken into account by passing vectors through blocks, represented in Figure 2, consisting of two sublayers.

- The first sublayer connects its input via a residual connection to a layer normalization LN followed by a linear graph convolution $CONV$ of length 1, followed by a dropout regularization DO:

  $$\boldsymbol{h}'_n = \boldsymbol{h}_n + DO(CONV(LN(\boldsymbol{h}_n)))$$

  The graph convolution passes messages between nodes. In our context, it passes information from variables to constraints, and conversely.

- The second sublayer takes as input the result of first one, and connects it with a residual connection to a sequence made of a layer normalization LN, a MLP transformation and a dropout regularization DO:

  $$\boldsymbol{h}_n = \boldsymbol{h}'_n + DO(MLP(LN(\boldsymbol{h}'_n)))$$

  The MLP is in charge of finding non-linear interactions in the information collected in the previous sublayer.

This block structure, depicted in Figure 2, is repeated several times, typically 5 times in our experiments, in order to extend the domain of locality. The learnable parameters of a block are the parameters of the convolution in the first sublayer and the parameters of the MLP in the second one. Remark that we start each sublayer with normalization, as it has become the standard approach in Transformer recently (Chen et al., 2018). We note in passing that this has also been experimented with by Gasse et al. (2019) in the context of MILP, although only once before the GNN input, whereas we normalize twice per block, at each block.

Finally, the GNN returns the vectors associated with dualized constraints $\{\boldsymbol{h}_c\}_c$. Each vector $\boldsymbol{h}_c$ is interpreted as the concatenation of two vectors $[\boldsymbol{z}_\mu; \boldsymbol{z}_\sigma]$ from which we compute $\boldsymbol{z}_c = \boldsymbol{z}_\mu + \exp(\boldsymbol{z}_\sigma) \cdot \boldsymbol{\epsilon}$ where elements of $\boldsymbol{\epsilon}$ are sampled from the normal distribution. This concludes the implementation of the probabilistic encoder $q_{\boldsymbol{\phi}}$.

Decoder Recall that, in our architecture, from each latent vector representation $\boldsymbol{z}_c$ of dualized constraint c we want to compute the scalar deviation δ_c to the CR dual value λ_c so that the sum of the two improves the Lagrangian bound given by the CR dual solution. In other words, we want to compute $\boldsymbol{\delta}$ such as $\boldsymbol{\pi} = [\boldsymbol{\lambda} + \boldsymbol{\delta}]_+$ gives a *good* Lagrangian bound $LR(\boldsymbol{\pi})$. Its exact computation is of combinatorial nature and problem specific.[5]

The probabilistic nature of the encoder-decoder can be exploited further: during evaluation, when computing a Lagrangian Relaxation, we sample constraint representations 5 times from the probabilistic encoder and return the best $LR(\boldsymbol{\pi})$ value from the decoder.

Link with Energy Based Models in Structured Prediction The relaxed Lagrangian problem usually decomposes into independent subproblems due to the dualization of the linking constraints. In this case, for each independent Lagrangian subproblem we want to find its optimal variable assignment, usually with local combinatory constraints, for its objective reparameterized with $\boldsymbol{\pi}$. This approach is typical of structured prediction: we leverage neural networks to extract features in order to compute local energies (scalars), which are used by a combinatorial algorithm outputting a structure whose objective value can be interpreted as a global energy. For instance, this is reminiscent of how graph-based syntactic parsing models in NLP compute parse scores (global energies) as sums of arc scores (local energies) computed by RNNs followed by MLPs, where the choice of arcs is guided by well-formedness constraints enforced by a maximum spanning tree solver, see for instance (Kiperwasser & Goldberg, 2016). Thus, the decoder is local to each dualized constraint, and we leverage subproblems to interconnect predictions:

1. We compute LMs (local energies) $\pi_c = [\lambda_c + f_{\boldsymbol{\theta}}(\boldsymbol{z}_c)]_+$ for all dualized constraints c, where $f_{\boldsymbol{\theta}}$ is implemented as a feed-forward network computing the deviation.
2. For parameter learning or if the subproblem solutions or the Lagrangian bound are the desired output, vector $\boldsymbol{\pi}$ is then passed to the Lagrangian subproblems which compute independently and in parallel their local solutions $\boldsymbol{x}$ and the corresponding values are summed to give (global energy) $LR(\boldsymbol{\pi})$.

3. Related Work

There is growing interest in leveraging Machine Learning (ML) alongside optimization algorithms (Bengio et al., 2021), in particular with the goal of improving MILP solvers's efficiency (Zhang et al., 2023). Indeed, even though MILP solvers solve problems in an exact way, they make many heuristic decisions which can be based on data-driven ML systems. For instance, classifiers have been

[5] $LR(\boldsymbol{\pi})$ is described in Appendices B and C for the two problems on which we evaluate our method.

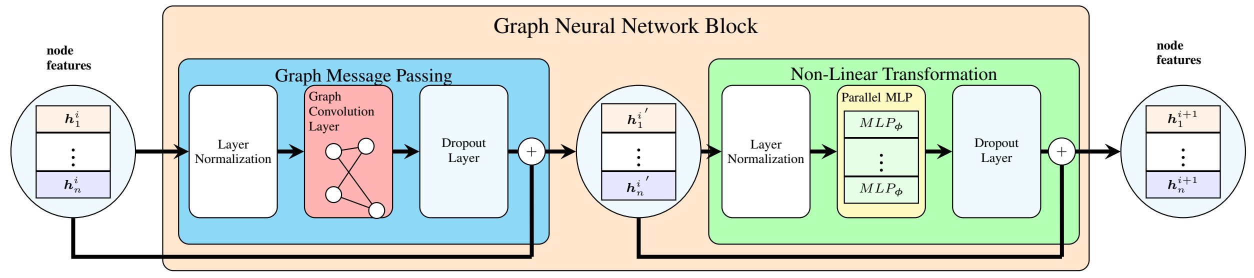

Figure 2. The Graph Neural Network block. The first part is graph message-passing: we apply layer normalization to node features, then convolution over the instance's bipartite graph representation and finally dropout. The second phase consists of normalization, a Multi-Layer perceptron in parallel over all the nodes of the bipartite graph, then dropout. Both sublayers use residual connection between input and output. We apply this block several times to improve feature representations.

designed for Branch and Bound (B&B) algorithms (Lodi & Zarpellon, 2017) in order to choose which variables to branch on (Alvarez et al., 2017; Khalil et al., 2016; He et al., 2014; Etheve et al., 2020), which B&B node to process (Yilmaz & Yorke-Smith, 2021; Labassi et al., 2022), to decide when to perform heuristics (Hottung et al., 2020; Khalil et al., 2017) or how to schedule them (Chmiela et al., 2021).

In this work, we depart from this main trend and predict a dual bound for MILP instances sharing common features, which can in turn be used to improve solvers. Several propositions have tackled the prediction of high quality primal and dual bounds. For instance, Nair et al. (2020) predict partial variable assignments, resulting in small MILPs which can be solved to optimality. Another way to provide primal solutions is to transform a MILP into an easier one, solve it and apply a procedure to recover primal feasibility (Dalle et al., 2022; Parmentier, 2022). Many works use Reinforcement Learning and guided greedy decoding to find high-quality approximate solutions for NP-hard problems, *e.g.* (Kool et al., 2019). For dual bounds, ML has been employed for cut selection in cutting planes algorithms (Baltean-Lugojan et al., 2018; Wang et al., 2023; Balcan et al., 2021; Berthold et al., 2022; Tang et al., 2020; Huang et al., 2022; Afia & Kabbaj, 2017, Tetouan Morocco; Morabit et al., 2021), an essential feature of MILP solvers which must balance strengthened linear relaxations with increased computations due to added cuts (Dey & Molinaro, 2018).

Regarding specifically prediction for Lagrangian dual solutions, Nair et al. (2018) consider 2-stage stochastic MILPs, approached by a Lagrangian decomposition for which they learn to predict LMs compliant with any second-stage scenario to give a good bound on average. Lange & Swoboda (2021) propose a heuristic to solve binary ILPs based on a specific LD where the relaxed LR problem is decomposed into many subproblems, one per constraint, solved using binary decision diagrams. This method is modified by Abbas & Swoboda (2022) to be run on GPU. The block coordinate method used to heuristically solve LD is improved by learning parameters used for initializing and updating Lagrangian multipliers (Abbas & Swoboda, 2024). In contrast to our generic method, other previous attempts at Lagrangian dual solution prediction for deterministic MILPs focus on a specific combinatorial optimization problem, such as the cutting stock problem (Kraul et al., 2023), where a MLP predicts the dual Lagrangian value for each constraint (*i.e.* stock) separately, or the unit commitment problem (Sugishita et al., 2024), where the same problem is solved daily but with different demand forecasts with either a MLP or a random forest which predicts dual solutions used to warm-start BM.

In our work, we assume that the set of dualized constraints is given, but predicting such a set is also an active avenue of research where solutions must find a good compromise between the quality of the Lagrangian dual bound and the running time to compute this bound (Kruber et al., 2017; Basso et al., 2020).

Regarding our use of GNNs, this has become a common MILP feature extractor in recent works, either on the factor bipartite graph (Gasse et al., 2019; Nair et al., 2020) or directly on the underlying graph in routing problems (Sun & Yang, 2023). While these works use GNNs to extract features for variable predictions, we use GNNs to extract features for constraints, which are then decoded to Lagrangian Multipliers. Our specific GNN architecture is based on the block structure of Transformers (Vaswani et al., 2017) where attention is replaced by a linear graph convolution.

Our method predicts a deviation from an initial solution, and can also be understood as predicting a gradient or subgradient step. We can thus relate our approach to works on gradient descent (Andrychowicz et al., 2016; Ba et al., 2022) and unrolling of iterative methods for structured prediction (Yang et al., 2016; Belanger et al., 2017). This is also related to amortization especially in relation with subgradient methods already studied in the ML community (Komodakis et al., 2014; Meshi et al., 2010). However, in previous works amortization was performed only during training, and iterative methods were used at testing time.

In the case of semi-amortization, our method can be used as an informed starting point for a descent algorithm applied to quadratic optimization (Sambharya et al., 2022), or probabilistic inference (Kim et al., 2018).

4. Evaluation

We evaluate our approach[6] on two standard problems of Operations Research, namely Multi-Commodity Fixed-Charge Network Design and Generalized Assignment.

4.1. Problems and Datasets

We review briefly the two problems and the data generation process. More details on MILP formulations and Lagrangian relaxations can be found in Appendices B and C and a thorough description of dataset generation is given in Appendix D.

Multi-Commodity Fixed-Charge Network Design (MC) Given a network with arc capacities and a set of commodities, MC consists in activating a subset of arcs and routing each commodity from its origin to its destination, possibly fractioned on several paths, using only the activated arcs. The objective is to minimize the total cost induced by the activation of arcs and the routing of commodities. This problem has been used in many real-world applications for a long time, see for instance (Magnanti & Wong, 1984) for telecommunications. It is NP-hard and its continuous relaxation provides poor bounds when arc capacities are high. Hence, it is usually tackled with Lagrangian relaxation-based methods (Akhavan Kazemzadeh et al., 2022).

While the Canad dataset is the standard and well-established dataset of instances for evaluating MC solvers (Crainic et al., 2001), it is too small to be used as a training set for Machine Learning where large collections of instances sharing common features are required. Thus, we generate new instances from a subset of instances of the Canad dataset (Crainic et al., 2001), that we divide into four datasets of increasing difficulty. The first two datasets, MC-SML-40 and MC-SML-VAR, contain instances that all share the same network (20 nodes and 230 edges) and the same arc capacities and fixed costs, but with different values for origins, destinations, volumes, and routing costs. Instances of the former all involve the same number of commodities (40), while for the latter the number of commodities varies from 40 to 200. Dataset MC-BIG-40 is generated similarly to MC-SML-40 but upon a bigger graph containing 30 nodes and 520 arcs. Finally, MC-BIG-VAR contains examples generated using either the network of MC-SML-40 or the one of MC-BIG-40, with the number of commodities varying between 40 and 200.

[6]See Appendix E for hyperparameter values used in our experiments.

Generalized Assignment (GA) GA consists, given a set of items and a set of capacitated bins, in assigning items to bins without exceeding their capacity in order to maximize the profit of the assignment. GA is a well-known problem in Operations Research and has numerous applications such as job-scheduling in Computer Science (Balachandran, 1976), distributed caching (Fleischer et al., 2011) or even parking allocation (Mladenović et al., 2020).

We generated two datasets, namely GA-10-100 and GA-20-400, containing respectively instances with 10 bins and 100 items, and with 20 bins and 400 items. Weights, profits and bin capacities are sampled using a distribution determined from values of standard instances (Yagiura et al., 1999).

4.2. Numerical Results

We want to evaluate how our Lagrangian bound prediction compares to an iterative model based on subgradient, and how useful the former is as an initial point to warm-start the latter. For that purpose, we choose a state-of-the-art proximal bundle solver provided by SMS++ (Frangioni et al., 2023) which allows writing a MILP in a block structure fashion and using decomposition techniques to solve subproblems efficiently. We also compare our approach with CR computed using the CPLEX[7] optimizer.

All MILP instances for which we want to evaluate our model are first solved by SMS++. For an instance ι we denote $\boldsymbol{\pi}^*_\iota$ the LMs returned by SMS++.

Metrics We use the percentage gap as metrics to evaluate the quality of the bounds computed by the different systems, averaged over a dataset of instances $\mathcal{I}$. For a system returning a bound B_ι for an instance ι the percentage GAP is:

$$100 \times \frac{1}{|\mathcal{I}|} \sum_{\iota \in \mathcal{I}} \frac{LR(\boldsymbol{\pi}^*_\iota) - B_\iota}{LR(\boldsymbol{\pi}^*_\iota)}$$

GAP measures the quality of the bound B_ι, and is zero when B_ι equals the optimal Lagrangian bound.

Data for Evaluation We divide each dataset of 2000 instances in train (80%), validation (10%) and test (10%). Parameters are learned on the train set, model selection is performed on validation set, and test proxies for unseen data. Results are averaged over 3 random initializations.

Bound Accuracy Table 1 reports the performance of different systems on our 6 datasets. We compare the bound returned by CR, and the bound of the Lagrangian relaxed

[7]`https://www.ibm.com/products/ilog-cplex-optimization-studio`

problem obtained with LMs computed by different methods:

- LR(0) is the LR value computed with LMs set to zero.
- LR(CR) is the LR value computed with LMs set to CR dual solution.
- LR(k-NN) is the LR value computed with LMs set to the average value of LMs from the train set returned by a k-NN regressor.[8]
- LR(MLP) is the LR value computed with LMs returned by a MLP[9] instead of the GNN-based encoder-decoder.
- Ours, that is the LR value computed with LMs set the output of our encoder-decoder. As written in the previous section we sample 5 LM assignments per instance and return the best LR value.

For all datasets, our method outperforms other approaches. Our model can reach 2% difference with BM on MC-SML-40, the easiest corpus with a small fixed network and a fixed number of commodities. This means that one pass through our network can save numerous iterations if we can accept about 2% bound error on average. The margin with other methods is quite large for MC datasets where the CR bound is far from the optimum, with a gap reduction ranging from 77% (MC-BIG-VAR) to 84% (MC-SML-40) depending on the dataset. For GA, where CR is closer to the optimum, our model still manages to find better solutions. Even though the gap absolute difference may seem small, the gap reduction from the second-best model LR(CR) ranges from 30% (GA-10-100) to 44% (GA-20-400), a significant error reduction.

Compared to simpler ML approaches, we see *(i)* that retrieving LM values from k-NN clustering is not a viable solution, even when the validation instances are close to the training instances (MC-SML-40), clustering cannot find meaningful neighbors, and *(ii)* the graph feature extractor (GNN) is paramount: the LR(MLP) architecture seems unable to deviate LMs consistently from CR solutions and can even perform worse than CR or LR(CR) (GA-20-400).

Regarding speed, LR(0) is the fastest since it simply amounts to solving the relaxed Lagrangian problem with the original costs. Then CR and LR(CR) are second, the difference being that for the latter after solving CR, the dual solution $\boldsymbol{\lambda}$ is used to compute the $LR(\boldsymbol{\lambda})$. Slowness for LR(MLP) and LR(k-NN) is mainly caused by feature extraction (*cf*. Appendix G).

[8]See Appendices F and G for more information about the implemented k-NN method.

[9]Additional initial features that the ones used in our model are used, see Appendix G for more details.

Table 1. Bound accuracies of different methods on test sets averaged by instance.

Dataset	Methods	GAP %	time (ms)
MC-SML-40	CR	12.99	90.63
	LR(0)	100.00	**0.35**
	LR(CR)	12.97	90.98
	LR(k-NN)	38.80	219.42
	LR(MLP)	10.70	142.48
	ours	**2.09**	120.96
MC-SML-VAR	CR	22.29	283.63
	LR(0)	100.00	**1.32**
	LR(CR)	22.29	285.03
	LR(k-NN)	44.12	371.51
	LR(MLP)	16.71	369.61
	ours	**4.42**	374.20
MC-BIG-40	CR	15.94	220.91
	LR(0)	100.00	**0.75**
	LR(CR)	15.85	229.57
	LR(k-NN)	54.57	334.99
	LR(MLP)	13.67	556.89
	ours	**4.20**	283.40
MC-BIG-VAR	CR	20.66	287.20
	LR(0)	100.00	**1.37**
	LR(CR)	20.63	288.55
	LR(k-NN)	49.74	886.91
	LR(MLP)	16.14	515.60
	ours	**4.77**	374.78
GA-10-100	CR	1.91	9.59
	LR(0)	3.13	**0.44**
	LR(CR)	0.79	10.15
	LR(k-NN)	1.07	11.70
	LR(MLP)	0.78	51.71
	ours	**0.55**	16.19
GA-20-400	CR	0.44	71.40
	LR(0)	2.70	**7.51**
	LR(CR)	0.27	78.80
	LR(k-NN)	0.43	89.68
	LR(MLP)	0.28	114.41
	ours	**0.15**	124.96

Warm-starting Iterative Solvers We want to test whether the Lagrangian Multipliers predicted by our model can be used as an informed starting point for an iterative solver for the Lagrangian Dual LD, namely the bundle method as implemented by SMS++ and the subgradient method. While the latter is simple to implement and only requires solving $LR(\boldsymbol{\pi})$, it has a non-smooth objective and the subgradient does not always give a descent direction, resulting in unstable updates. In contrast, the bundle method is stabilized with a quadratic penalty assuring a smooth objective, at the expense of longer computation times. We hope that our model can produce good starting points for both methods and thus avoid many early iterations.

In Table 2 we compare different initial LM vectors on the validation set of MC-BIG-VAR for the bundle method. We run our bundle solver until the difference between $LR(\boldsymbol{\pi}^*)$ and the current bound is smaller than the threshold ϵ. We average resolution times and numbers of iterations over instances, and compute standard deviation. We compare three initialization methods: zero, using CR dual solutions, and our model's predictions.

Table 2. Impact of initialization for a Bundle solver on MC-BIG-VAR. We consider initializations from the null vector (zero), the continuous relaxation duals (CR), and our model (Ours).

ϵ	zero		CR		Ours	
	time (s)	# iter.	time (s)	# iter.	time (s)	# iter.
1e-1	34.34 (±81.22)	90.12 (±52.41)	31.67 (±75.12)	83.00 (±50.73)	**16.09** (± 42.79)	**60.39** (± 41.37)
1e-2	68.80 (±188.21)	141.43 (±112.72)	62.09 (±171.71)	133.26 (±109.60)	**36.10** (± 106.22)	**105.93**(± 97.04)
1e-3	100.71 (±288.16)	188.14 (±167.33)	89.26 (±251.15)	179.40 (±170.15)	**57.23** (± 177.24)	**143.36** (± 142.58)
1e-4	105.03 (±298.53)	207.90 (±198.92)	101.14 (±283.52)	200.42 (±197.60)	**63.25** (± 190.47)	**159.42** (± 162.32)

We can see that CR is not competitive with the null initialization, since the small gain in the number of iterations is absorbed by the supplementary computation. However, our model's predictions give a significant improvement over the other two initialization methods, despite the additional prediction time. Resolution time is roughly halved for the coarsest threshold, and above one-third faster for the finest one. This is expected, as gradient-based methods naturally slow down as they approach convergence. In appendix I we perform the same experiments as in Table 2 for the subgradient method.

Ablation Study In Table 3 we compare three variants of our original model, denoted `ours`, on MC-SML-40 and MC-BIG-VAR. Results are averaged over 3 runs.

In the first variant `-max`, instead of sampling multiple LMs for each dualized constraint and keeping the best, we take one sample only per constraint. We can see that this has a minor incidence on the quality of the returned solution. In `-sum`, the dual solution values are passed as constraint node features but are not added to the output of the decoder to produce LMs, *i.e.* the network must transport these values from its input layer to its output. This has a sensible negative impact of GAP scores. In the third variant, `-cr` the CR solution is not given as input features to the network (nor added to the network's output). This is challenging because the network does not have access to a good starting point, this is equivalent to initializing LMs to zero. The last variant, `-sample`, uses CR as `ours` but does not sample representations $\boldsymbol{z}_c$ in the latent domain. We interpret the vector $\boldsymbol{h}_c$ associated with dualized constraint c after the GNN stack directly as vector $\boldsymbol{z}_c$, making the encoder deterministic.

We can see that the performance of `-sum` just below `ours`, while `-cr` cannot return competitive bounds. This indicates that the CR solution passed as input features is essential for our architecture to get good performance, whereas the computation of the deviation instead of the full LM directly is not an important trait. Still, we note that performances of `-cr` should be compared with $LR(0)$ in Table 1 rather than $LR(CR)$. In that case, we see that the GAP reduction is around 80%, making it clear that our model is not simply repeating CR solutions. This means that our model could be used without the CR solution information as input, opening our methods to a wider range of problems, and paving the way for faster models.

Table 3. Ablation studies comparing the prediction of our model with, predicting LMs on rather than deviation from CR (`-sum`), not using CR features at all (`-cr`), or replacing the probabilistic encoder by a deterministic one (`-sample`).

	GAP %	
model	MC-SML-40	MC-BIG-VAR
`ours`	2.09	4.77
`-max`	2.10	4.79
`-sum`	2.63	6.77
`-cr`	20.26	23.78
`-sample`	2.18	5.86

Table 4. Generalization results over bigger instances.

	GAP %		time (s)	
# commodities	Ours	LR(CR)	Ours	LR(CR)
160	6.51	27.85	1.533	0.8915
200	7.62	30.18	1.4328	1.0889

Finally, `-sample` is a system trained without sampling at training time, *i.e.* the encoder-decoder is deterministic. We see that sampling gives a slight performance increase on small and bigger instances.

In Appendix H we compare the architecture we introduce in this work with the architecture proposed by Nair et al. in (Nair et al., 2018) and the one presented in (Gasse et al., 2019), for several layers from 1 to 10.

Generalization Properties We test the model trained on MC-BIG-VAR on a dataset composed of 1000 bigger instances. They are created using the biggest graph used to generate the MC-BIG-VAR dataset but contain 160 or 200 commodities whereas the instances of MC-BIG-VAR with the same graphs only contain up to 120 commodities. In Table 4 we can see that our model still performs well in these instances dividing by 4 the gap provided by LR(CR).

5. Conclusion

We have presented a novel method to compute good Lagrangian dual solutions for MILPs sharing common attributes, by predicting Lagrangian multipliers. We cast this problem as an encoder-decoder prediction, where the probabilistic encoder outputs one distribution per dualized constraint from which we sample constraint vector representation. Then a decoder transforms these representations into Lagrangian multipliers.

We experimentally showed that this method gives bounds significantly better than the commonly used heuristics on two standard combinatorial problems: it reduces the continuous relaxation gap to the optimal bound up to 85%, and when used to warm-start an iterative solver, the points predicted by our models reduce solving times by a large margin.

Our predictions could be exploited in primal heuristics, possibly with auxiliary losses predicting values from variable nodes, or to efficiently guide a Branch-and-Bound exact search. Predictions could be stacked to act as an unrolled iterative solver. Finally, we can see our model as performing denoising from a previous solution and could be adapted to fit in a diffusion model.

Acknowledgments

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-23-CE23-0005 (project SEMIAMOR).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abbas, A. and Swoboda, P. Fastdog: Fast discrete optimization on gpu. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 439–449, June 2022.

Abbas, A. and Swoboda, P. Doge-train: Discrete optimization on GPU with end-to-end training. In Wooldridge, M. J., Dy, J. G., and Natarajan, S. (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 20623–20631. AAAI Press, 2024.

Afia, A. E. and Kabbaj, M. M. Supervised learning in branch-and-cut strategies. In *International Conference on Big Data Cloud and Applications*, 2017, Tetouan Morocco.

Akhavan Kazemzadeh, M. R., Bektaş, T., Crainic, T. G., Frangioni, A., Gendron, B., and Gorgone, E. Node-based lagrangian relaxations for multicommodity capacitated fixed-charge network design. *Discrete Applied Mathematics*, 308:pp. 255–275, 2022. Combinatorial Optimization ISCO 2018.

Alvarez, A. M., Louveaux, Q., and Wehenkel, L. A machine learning-based approximation of strong branching. *INFORMS J. Comput.*, 29(1):185–195, 2017.

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Ba, J., Erdogdu, M. A., Suzuki, T., Wang, Z., Wu, D., and Yang, G. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 37932–37946. Curran Associates, Inc., 2022.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.

Balachandran, V. An Integer Generalized Transportation Model for Optimal Job Assignment in Computer Networks. *Operations Research*, 24(4):742–759, August 1976.

Balcan, M.-F. F., Prasad, S., Sandholm, T., and Vitercik, E. Sample complexity of tree search configuration: Cutting planes and beyond. *Advances in Neural Information Processing Systems*, 34:pp. 4015–4027, 2021.

Baltean-Lugojan, R., Bonami, P., Misener, R., and Tramontani, A. Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks. In *optimization-online*, 2018.

Basso, S., Ceselli, A., and Tettamanzi, A. Random sampling and machine learning to understand good decompositions. *Annals of Operations Research*, 284(2):pp. 501–526, January 2020.

Beasley, J. E. A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):pp. 151–164, 1990.

Belanger, D., Yang, B., and McCallum, A. End-to-end learning for structured prediction energy networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 429–439. PMLR, 2017.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):pp. 405–421, 2021.

Berthold, T., Francobaldi, M., and Hendel, G. Learning to use local cuts. *arXiv preprint arXiv:2206.11618*, 2022.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics.

Chmiela, A., Khalil, E. B., Gleixner, A., Lodi, A., and Pokutta, S. Learning to schedule heuristics in branch and bound. *Advances in Neural Information Processing Systems*, 34:pp. 24235–24246, 2021.

Conforti, M., Cornuéjols, G., and Zambelli, G. *Integer Programming*. Springer, New York, 2014.

Crainic, T. G., Frangioni, A., and Gendron, B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):pp. 73–99, 2001.

Dalle, G., Baty, L., Bouvier, L., and Parmentier, A. Learning with combinatorial optimization layers: a probabilistic approach. *arXiv preprint arXiv:2207.13513*, 2022.

Dey, S. S. and Molinaro, M. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170 (1):pp. 237–266, July 2018. ISSN 0025-5610, 1436-4646.

Etheve, M., Alès, Z., Bissuel, C., Juan, O., and Kedad-Sidhoum, S. Reinforcement learning for variable selection in a branch and bound algorithm. In Hebrard, E. and Musliu, N. (eds.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21-24, 2020, Proceedings*, volume 12296 of *Lecture Notes in Computer Science*, pp. 176–185. Springer, 2020.

Fleischer, L., Goemans, M. X., Mirrokni, V. S., and Sviridenko, M. Tight approximation algorithms for maximum separable assignment problems. *Mathematics of Operations Research*, 36(3):pp. 416–431, 2011. ISSN 0364765X, 15265471.

Frangioni, A., Iardella, N., and Durbano Lobato, R. SMS++, 2023. URL `https://gitlab.com/smspp/smspp-project`.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Gendron, B., Crainic, T. G., and Frangioni, A. *Multicommodity Capacitated Network Design*, pp. 1–19. Springer US, Boston, MA, 1999.

Geoffrion, A. M. Lagrangean relaxation for integer programming. In Balinski, M. L. (ed.), *Approaches to Integer Programming*, pp. 82–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 1974. ISBN 978-3-642-00740-8.

Golden, B., Raghavan, S., and Wasil, E. (eds.). *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*. Springer US, Boston, MA, 2008. ISBN 978-0-387-77777-1 978-0-387-77778-8.

He, H., Daume III, H., and Eisner, J. M. Learning to search in branch and bound algorithms. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE, June 2016.

Hiriart-Urruty, J.-B. and Lemaréchal, C. *Convex analysis and minimization algorithms II: Advance Theory and Bundle Methods*, volume 305. Springer science & business media, 1996.

Hottung, A., Tanaka, S., and Tierney, K. Deep learning assisted heuristic tree search for the container premarshalling problem. *Computers & Operations Research*, 113:104781, 2020.

Huang, Z., Wang, K., Liu, F., Zhen, H.-L., Zhang, W., Yuan, M., Hao, J., Yu, Y., and Wang, J. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, 2022. ISSN 0031-3203.

Khalil, E., Le Bodic, P., Song, L., Nemhauser, G., and Dilkina, B. Learning to branch in mixed integer programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016.

Khalil, E. B., Dilkina, B., Nemhauser, G. L., Ahmed, S., and Shao, Y. Learning to run heuristics in tree search. In *Ijcai*, pp. 659–666, 2017.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-amortized variational autoencoders. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2678–2687. PMLR, 10–15 Jul 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kiperwasser, E. and Goldberg, Y. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:pp. 313–327, 2016.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Komodakis, N., Xiang, B., and Paragios, N. A Framework for Efficient Structured Max-Margin Learning of High-Order MRF Models. Technical Report 7, 2014.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Korte, B. H. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms*. Number v. 21 in Algorithms and Combinatorics. Springer, Heidelberg ; New York, 5th ed edition, 2012.

Kraul, S., Seizinger, M., and Brunner, J. O. Machine learning–supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation. *INFORMS Journal on Computing*, 35(3):pp. 692–709, 2023.

Kruber, M., Lübbecke, M. E., and Parmentier, A. Learning when to use a decomposition. In Salvagnin, D. and Lombardi, M. (eds.), *Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings*, volume 10335 of *Lecture Notes in Computer Science*, pp. 202–210. Springer, 2017.

Labassi, A. G., Chételat, D., and Lodi, A. Learning to compare nodes in branch and bound with graph neural networks. *Advances in Neural Information Processing Systems*, 35:pp. 32000–32010, 2022.

Lange, J.-H. and Swoboda, P. Efficient message passing for 0–1 ilps with binary decision diagrams. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. pp. 6000–6010. PMLR, 18–24 Jul 2021.

Le, Q., Smola, A., and Vishwanathan, S. Bundle methods for machine learning. *Advances in neural information processing systems*, 20, 2007.

Le Cun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Lodi, A. and Zarpellon, G. On learning and branching: a survey. *Top*, 25:pp. 207–236, 2017.

Magnanti, T. L. and Wong, R. T. Network design and transportation planning: Models and algorithms. *Transp. Sci.*, 18(1):pp. 1–55, 1984.

Meshi, O., Sontag, D., Jaakkola, T., and Globerson, A. Learning efficiently with approximate inference via dual losses. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 783–790, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Mladenović, M., Delot, T., Laporte, G., and Wilbaut, C. The parking allocation problem for connected vehicles. *Journal of Heuristics*, 26(3):377–399, June 2020. ISSN 1381-1231, 1572-9397.

Morabit, M., Desaulniers, G., and Lodi, A. Machine-Learning–Based Column Selection for Column Generation. *Transportation Science*, 55(4):815–831, July 2021. ISSN 0041-1655, 1526-5447.

Nair, V., Dvijotham, D., Dunning, I., and Vinyals, O. Learning fast optimizers for contextual stochastic integer programs. In *UAI*, pp. 591–600, 2018.

Nair, V., Bartunov, S., Gimeno, F., von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., Addanki, R., Hapuarachchi, T., Keck, T., Keeling, J., Kohli, P., Ktena, I., Li, Y., Vinyals, O., and Zwols, Y. Solving mixed integer programs using neural networks. *CoRR*, abs/2012.13349, 2020.

Parmentier, A. Learning to approximate industrial problems by operations research classic problems. *Oper. Res.*, 70 (1):606–623, 2022.

Polyak, B. *Introduction to Optimization*. Optimization Software, New York, 1987.

Rybkin, O., Daniilidis, K., and Levine, S. Simple and effective vae training with calibrated decoders. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9179–9189. PMLR, 18–24 Jul 2021.

Sambharya, R., Hall, G., Amos, B., and Stellato, B. End-to-end learning to warm-start for real-time quadratic optimization. In *Conference on Learning for Dynamics & Control*, 2022.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):pp. 1929–1958, 2014.

Sugishita, N., Grothey, A., and McKinnon, K. Use of Machine Learning Models to Warmstart Column Generation for Unit Commitment. *INFORMS Journal on Computing*, January 2024. ISSN 1091-9856, 1526-5528.

Sun, Z. and Yang, Y. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Tang, Y., Agrawal, S., and Faenza, Y. Reinforcement learning for integer programming: Learning to cut. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9367–9376. PMLR, 13–18 Jul 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Z., Li, X., Wang, J., Kuang, Y., Yuan, M., Zeng, J., Zhang, Y., and Wu, F. Learning cut selection for mixed-integer linear programming via hierarchical sequence model. In *The Eleventh International Conference on Learning Representations*, 2023.

Wolsey, L. A. *Integer Programming*. Wiley, Hoboken, NJ, second edition edition, 2021.

Yagiura, M., Yamaguchi, T., and Ibaraki, T. *A Variable Depth Search Algorithm for the Generalized Assignment Problem*, pp. 459–471. Springer US, Boston, MA, 1999. ISBN 978-1-4615-5775-3.

Yang, Y., Sun, J., Li, H., and Xu, Z. Deep admm-net for compressive sensing mri. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Yilmaz, K. and Yorke-Smith, N. A study of learning search approximation in mixed integer branch and bound: Node selection in scip. *AI*, 2(2):pp. 150–178, 2021. ISSN 2673-2688.

Zhang, J., Liu, C., Li, X., Zhen, H.-L., Yuan, M., Li, Y., and Yan, J. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519:pp. 205–217, 2023.

A. Initial Features

To extract useful features, we define a network based on graph convolutions presented in Figure 1 in the line of the work of (Gasse et al., 2019) on MILP encoding. We detail the initial node features $\{\boldsymbol{e}_n\}_n$ of the MILP-encoding bipartite graph presented in Section 2.3.

Given an instance of the form:

$$(P) \qquad \min_{\boldsymbol{x}} \boldsymbol{w}^\top \boldsymbol{x} \tag{2a}$$

$$\boldsymbol{A}\boldsymbol{x} \begin{pmatrix} \geq \\ = \end{pmatrix} \boldsymbol{b} \tag{2b}$$

$$\boldsymbol{x} \in \mathbb{R}^m_+ \times \mathbb{N}^p \tag{2c}$$

we consider the following initial features for a variable x_j:

- its coefficient w_j in the objective function;
- its value in the primal solution of CR;
- its reduced cost $\bar{c}_j = w_j - \boldsymbol{\lambda}^T \boldsymbol{A}_j$ in CR where $\boldsymbol{A}_j$ is the j^{th} column of $\boldsymbol{A}$ and $\boldsymbol{\lambda}$ is the dual solution of CR;
- a binary value indicating whether x_j is integral or continuous.

For constraint $\boldsymbol{a}^\top \boldsymbol{x} \begin{pmatrix} \geq \\ = \end{pmatrix} b$ of (2b), we consider:

- the right-hand side b of the constraint;
- the value of the associated dual solution in CR;
- one binary value indicating whether the constraint is an equality or an inequality;
- one binary value stating whether c is dualized in the relaxed Lagrangian problem.

We use for each node n of the bipartite graph a feature vector $\boldsymbol{e}_n \in \mathbb{R}^8$. The first four components are used to encode the initial features if n corresponds to a variable and are set to 0 otherwise, whereas the next four components are used only if n is associated with a constraint and are set to 0 otherwise.

B. Multi Commodity Capacitated Network Design Problem

A MC instance is given by a directed simple graph $D = (N, A)$, a set of commodities K, an arc-capacity vector c, and two cost vectors r and f. Each commodity $k \in K$ corresponds to a triplet (o^k, d^k, q^k) where $o^k \in N$ and $d^k \in N$ are the nodes corresponding to the origin and the destination of commodity k, and $q^k \in \mathbb{N}^*$ is its volume. For each arc, $(i,j) \in A$, $c_{ij} > 0$ corresponds to the maximum amount of flow that can be routed through (i,j) and $f_{ij} > 0$ corresponds to the fixed cost of using arc (i,j) to route commodities. For each arc $(i,j) \in A$ and each commodity $k \in K$, $r^k_{ij} > 0$ corresponds to the cost of routing one unit of commodity k through arc (i,j).

A MC solution consists of an arc subset $A' \subseteq A$ and, for each commodity $k \in K$, in a flow of value q^k from its origin o^k to its destination d^k with the following requirements: all commodities are only routed through arcs of A', and the total amount of flow routed through each arc $(i,j) \in A'$ does not exceed its capacity c_{ij}. The solution cost is the sum of the fixed costs over the arcs of A' plus the routing cost, the latter being the sum over all arcs $(i,j) \in A$ and all commodities $k \in K$ of the unitary routing cost r^k_{ij} multiplied by the amount of flow of k routed through (i,j).

B.1. MILP formulation

A standard model for the MC problem (Gendron et al., 1999) introduces two sets of variables: the continuous flow variables x^k_{ij} representing the amount of commodity k that is routed through arc (i,j) and the binary design variables y_{ij} representing whether or not arc (i,j) is used to route commodities. Denoting respectively by $N^+_i = \{j \in N \mid (i,j) \in A\}$ and $N^-_i = \{j \in N \mid (j,i) \in A\}$ the sets of forward and backward neighbors of a vertex $i \in N$, the MC problem can be modeled as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{(i,j)\in A} \left(f_{ij} y_{ij} + \sum_{k\in K} r^k_{ij} x^k_{ij} \right) \tag{3a}$$

$$\sum_{j\in N^+_i} x^k_{ij} - \sum_{j\in N^-_i} x^k_{ji} = b^k_i \quad \forall i \in N, \forall k \in K \tag{3b}$$

$$\sum_{k\in K} x^k_{ij} \leq c_{ij} y_{ij}, \qquad \forall (i,j) \in A \tag{3c}$$

$$x^k_{ij} = 0 \qquad \forall k \in K, \forall (i,j) \in A \text{ s.t. } i = d^k \text{ or } j = o^k \tag{3d}$$

$$0 \leq x^k_{ij} \leq q^k \qquad \forall (i,j) \in A, \forall k \in K \tag{3e}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{3f}$$

where

$$b^k_i = \begin{cases} q^k & \text{if } i = o^k, \\ -q^k & \text{if } i = d^k, \\ 0 & \text{otherwise.} \end{cases}$$

The objective function (3a) minimizes the sum of the routing and fixed costs. Equations (3b) are the flow conservation constraints that properly define the flow of each commodity through the graph. Constraints (3c) are the capacity constraints ensuring that the total amount of flow routed through each arc does not exceed its capacity or is zero if the arc is not used to route commodities. Equations (3d) ensure that a commodity is not routed on an arc entering its origin or leaving its destination. Finally inequalities (3e) are the bounds for the x variables and inequalities (3f) are the integer constraints for the design variables.

B.2. Lagrangian Knapsack Relaxation

A standard way to obtain good bounds for the MC problem is to solve the Lagrangian relaxation obtained by dualizing the flow conservation constraints (3b) in formulation (3a)-(3f). Let π_i^k be the Lagrangian multiplier associated with node $i \in N$ and commodity $k \in K$. Dualizing the flow conservation constraints gives the following relaxed Lagrangian problem $LR(\boldsymbol{\pi})$[10]:

$$\min_{(\boldsymbol{x},\boldsymbol{y}) \text{ satisfies (3c)}-\text{(3f)}} \sum_{(i,j)\in A} \left(f_{ij} y_{ij} + \sum_{k\in K} r_{ij}^k x_{ij}^k \right) + \sum_{k\in K}\sum_{i\in N} \pi_i^k \left(b_i^k - \sum_{j\in N_i^+} x_{ij}^k + \sum_{j\in N_i^-} x_{ji}^k \right)$$

Rearranging the terms in the objective function and observing that the relaxed Lagrangian problem is decomposed by arcs, we obtain a subproblem for each arc $(i,j) \in A$ of the form:

$$(LR_{ij}(\boldsymbol{\pi})) \quad \min_{\boldsymbol{x},\boldsymbol{y}} f_{ij} y_{ij} + \sum_{k\in K_{ij}} w_{ij}^k x_{ij}^k \tag{4a}$$

$$\sum_{k\in K_{ij}} x_{ij}^k \le c_{ij} y_{ij} \tag{4b}$$

$$0 \le x_{ij}^k \le q^k \qquad \forall k \in K_{ij} \tag{4c}$$

$$y_{ij} \in \{0,1\} \tag{4d}$$

where $w_{ij}^k = r_{ij}^k - \pi_i^k + \pi_j^k$ and $K_{ij} = \{k \in K \mid j \neq o^k \text{ and } i \neq d^k\}$ is the set of commodities that may be routed through arc (i,j).

For each $(i,j) \in A$, $LR_{ij}(\boldsymbol{\pi})$ is a MILP with only one binary variable. If $y_{ij} = 0$, then, by (4b) and (4c), $x_{ij}^k = 0$ for all $k \in K_{ij}$. If $y_{ij} = 1$, the problem reduces to a continuous knapsack problem. An optimal solution is obtained by ordering the commodities of K_{ij} with respect to decreasing values w_{ij}^k and setting for each variable x_{ij}^k the value $\max\{\min\{q^k, c_{ij} - \sum_{k\in K(k)} q^k\}, 0\}$ where $K(k)$ denotes the set of commodities that preceded k in the order. This step can be done in $O(|K_{ij}|)$ if one computes x_{ij}^k following the computed order. Hence, the complexity of the continuous knapsack problem is $O(|K_{ij}|\log(|K_{ij}|))$. The solution of $LR_{ij}(\boldsymbol{\pi})$ is the minimum between the cost of the continuous knapsack problem and $\mathbf{0}$.

[10]Since the dualized constraints are equations, $\boldsymbol{\pi}$ have no sign constraints.

Lagrangian duality implies that

$$LR(\boldsymbol{\pi}) = \sum_{(i,j)\in A} LR_{ij}(\boldsymbol{\pi}) + \sum_{i\in N}\sum_{k\in K} \pi_i^k b_i^k$$

is a lower bound for the MC problem and the best one is obtained by solving the following Lagrangian dual problem:

$$(LD) \qquad \max_{\boldsymbol{\pi}\in\mathbb{R}^{N\times K}} LR(\boldsymbol{\pi})$$

C. Generalized Assignment Problem

A GA instance is defined by a set I of items and a set J of bins. Each bin j is associated with a certain capacity c_j. For each item $i \in I$ and each bin $j \in J$, p_{ij} is the profit of assigning item i to bin j, and w_{ij} is the weight of item i inside bin j.

Considering a binary variable x_{ij} for each item and each bin that is equal to one if and only if item i is assigned to bin j, the GA problem can be formulated as:

$$\max_{\boldsymbol{x}} \sum_{i\in I}\sum_{j\in J} p_{ij} x_{ij} \tag{5a}$$

$$\sum_{j\in J} x_{ij} \le 1 \qquad \forall i \in I \tag{5b}$$

$$\sum_{i\in I} w_{ij} x_{ij} \le c_j \qquad \forall j \in J \tag{5c}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J. \tag{5d}$$

The objective function (5a) maximizes the total profit. Inequalities (5b) assert that each item is contained in no more than one bin. Inequalities (5c) ensure that the sum of the weights of the items assigned to a bin does not exceed its capacity. Finally, constraints (5d) assure the integrality of the variables.

C.1. Lagrangian Relaxation

A Lagrangian relaxation of the GA problem is obtained by dualizing (5b). For $i \in I$, let $\pi_i \ge 0$ be the Lagrangian multiplier of inequality (5b) associated with item i. For each bin j the subproblem becomes:

$$(LR_j(\boldsymbol{\pi})) \quad \max_{\boldsymbol{x}} \sum_{i\in I}\sum_{j\in J} (p_{ij} - \pi_i) x_{ij}$$

$$\sum_{i\in I} w_{ij} x_{ij} \le c_j$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I$$

It corresponds to an integer knapsack with $|I|$ binary variables. For $\boldsymbol{\pi} \ge \mathbf{0}$, the Lagrangian bound $LR(\boldsymbol{\pi})$ is:

$$LR(\boldsymbol{\pi}) = \sum_{j\in J} LR_j(\boldsymbol{\pi}) + \sum_{i\in I} \pi_i.$$

The Lagrangian dual can be then written as:

$$\min_{\boldsymbol{\pi}\in\mathbb{R}^{|I|}_{\geq 0}} LR(\boldsymbol{\pi})$$

D. Dataset collection details

In this appendix, we provide further details on the dataset construction.

Multi-Commodity Fixed-Charge Network Design We generate four datasets of 2000 instances each (1600 for training, 200 for validation and 200 for test) based on Canad instances (Crainic et al., 2001). These canad instances have been chosen such that the Lagrangian dual bound can be solved in nearly one second for the easiest instances and in approximately one hour for the hardest ones.

The first two datasets MC-SML-40 and MC-SML-VAR consider the same graph with 20 nodes and 230 edges, and the same capacity and fixed cost vectors. The first dataset has only instances with 40 commodities whereas the second one has instances with 40, 80, 120, 160 or 200 commodities.

Origins and destinations are randomly chosen using a uniform distribution. Volumes and routing costs are randomly sampled using a Gaussian distribution. Sampling uses four different means μ and variances σ^2 which are determined from the four canad instances p33, p34, p35 and p36 (having the same graph and fixed costs as the datasets) in order to generate four different types of instances: whether the fixed costs are high with respect to routing costs, and whether capacities are high with respect to commodity volumes.

The third dataset MC-BIG-40 is generated similarly as the first one except that it is based on a graph with 30 nodes and 520 edges. The means and variances used to sample the fixed costs and the volumes are determined from the four canad instances p49, p50, p51 and p52. The number of commodities is equal to 40 in each instance.

Finally, the last dataset MC-BIG-VAR contains instances with either the graph, capacities and fixed costs of the first two datasets or the ones of the third dataset. Sampling uses either the canad instances p33, p34, p35 and p36 or the canad instances p49, p50, p51 and p52 for determining the mean and variance, depending on the size of the graph. The number of commodities varies from 40 to 200 if the graph is the one of the first two datasets, and from 40 to 120 otherwise.

Generalized Assignment We create two datasets of GA instances containing 2000 instances each (1600 for training, 200 for validation and 200 for test). The first one contains instances with 10 bins and 100 items whereas the second one contains instances with 20 bins and 400 items. For each dataset, all instances are generated by randomly sampling capacities, weights and profits using a Gaussian distribution of mean μ and variance σ^2 and the values are clipped to an interval $[a, b]$. The values μ, σ^2, a and b are determined from the instance e10100 for the first dataset, and from the instance e20400 for the second one [11]. More specifically, for each type of data (capacities, weights and profits), μ and σ^2 are given by the average and variance of the values of the instance, and a and b are fixed to 0.8 times the minimum value and 1.2 times the maximum value, respectively.

E. Hyperparameters

Model Architecture For all datasets, the MLP F from initial features to high-dimensional is implemented as a linear transformation (8 to 250) followed by a non-linear activation. Then, we consider a linear transformation to the size of the internal representation of nodes for the GNN.

For MC we use 5 blocks, while for GA we use only 3. The fact that for GA are sufficient fewer layers can be explained by looking at the bipartite-graph representation of the instance that is denser for GA than for MC. For instance, in MC, a variable x_{ij}^k appears in three constraints involving several variables while in GA, each variable x_{ij} appears in $|I| + |J|$ constraints so the propagation needs fewer convolutions for the information to be propagated.

The hidden layer of the MLP in the second sub-layer of each block has a size of 1000.

The decoder is an MLP with one hidden layer of 250 nodes.

All non-linear activations are implemented as ReLU. Only the one for the output of the GA is a softplus.

The dropout rate is set to 0.25.

Optimiser Specifications We use as optimizer RAdam, with learning rate 0.0001 for MC and 0.00001 for GA, a Clip Norm (to 5) and exponential decay 0.9, step size 100000 and minimum learning rate 10^{-10}.

GPU specifics For the training on the datasets MC-SML-40, MC-BIG-40, GA-10-100 and GA-20-400 we use GPUs Nvidia Quadro RTX 5000 with 16 GB of RAM. To train the datasets MC-SML-VAR and MC-BIG-VAR we use Nvidia A40 GPUs accelerators with 48Gb of RAM. To test the performance we use Nvidia A40 GPUs accelerators with 48Gb of RAM for all models and all the datasets on validation and test.

CPU specifics The warm starting of the proximal Bundle in SMS++ needs only CPU, the experiments are done on

[11]Instances e10100 and e20400 are GA instances generated by (Yagiura et al., 1999) and available at `http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/gap/`.

Intel Core i7-8565U CPU @ 1.80GHz × 8.

F. k-NN

We consider the same features as for MLP (see Appendix G) and independently select for each dualized constraint c the 20 nearest neighbors with respect to the Euclidean distance. The LM predicted for c is the mean of the LMs associated with its neighbors. It is important to note that it is a supervised learning method while ours is an unsupervised one. We tried different values of k from 1 to 20 and we find that the best choice is 20. For the implementation we use the julia package NearestNeighbors.jl [12].

G. Features used for MLP and k-NN

Since MLP and k-NN do not use a mechanism such as convolution to propagate the information between the representations of the dualized constraints, we consider for initial features of each dualized constraint all the information provided to our model (see Appendix A for details), as well as a weighted linear combination of variable feature vectors. The weights are the variable coefficients in that constraint and each feature vector contains the initial features provided to our model for the variable and the following additional information:

- the mean values and deviations of the coefficients of that variable on the dualized constraints, and on the non dualized ones,
- its lower and upper bounds.

H. Ablation Study - Number of Layers

In Tables 5 and 6, we present the gaps of three different architectures with varying numbers of layers. The columns represent three different architectures: "Ours," the architecture we introduce in this work; "Nair," the architecture proposed by Nair et al. in (Nair et al., 2020); and "Gasse," the one presented in (Gasse et al., 2019). The rows indicate an incremental number of layers from one to ten.

Table 5. Test set 1 sample - MC-SML-40 - GAP

# Layers	Ours	Nair	Gasse
1	7.56	7.63	9.59
2	5.27	5.23	10.11
3	3.18	3.30	9.47
4	2.62	3.06	2.72
5	2.29	2.47	2.59
6	1.90	2.23	2.76
7	1.80	1.91	2.80
8	1.69	1.76	2.68
9	1.64	1.70	2.84
10	1.56	1.56	3.16

Table 6. Test set 1 sample - GA-10-100 - GAP (s)

# Layers	Ours	Nair	Gasse
1	0.553	0.557	0.70
2	0.543	0.546	0.699
3	0.533	0.524	0.695
4	0.509	0.524	0.690
5	0.512	0.517	0.682
6	0.513	0.518	0.720
7	0.510	0.511	0.785
8	0.512	0.517	0.752
9	0.511	0.515	0.785
10	0.512	0.516	0.722

From Table 6, we observe that for GA, using more than four layers seems to be counterproductive. This can be explained by examining the bipartite graph representation of the instance. In the Generalized Assignment problem, the shortest path between two different nodes associated with the relaxed constraints always consists of four edges. For the Multi-commodity problem, there is no similar bound, as the shortest path between two relaxed nodes depends on the specific structure of the instance. From Table 5, we see that adding more than six layers leads to diminishing improvements, though we can still enhance solution quality by increasing the number of layers.

Gasse's architecture is also more unstable, which could result in significantly higher gaps with some layers compared to fewer layers. This instability may be due to the absence of Layer Normalization, leading to very high gradient values.

Notice that the results in Table 5 and Table 6 show small differences compared to the ones on the main part for 5 layers, as they correspond to other runs of the training.

I. Subgradient Method Initialization

In Table 7 we perform the same experiments as Table 2 with a solver implementing the subgradient method. We see that subgradient method requires much more time than the bundle method, even for low precision levels. The initialization yields more or less the same results. This is likely due to the step size scheduler, which always starts with a step size of one. Then, at a given iteration i, the learning rate is $\frac{1}{1+m}$, where m is the total number of iterations where the predicted value is worse than the previous iteration. An accurate choice of step size can lead to better results, par-

[12] https://juliapackages.com/p/nearestneighbors

Table 7. Impact of initialization for a Sub-Gradient solver on MC-BIG-VAR. We consider initialization from the null vector (zero), the continuous relaxation duals (CR), and our model (Ours). We set the maximum iterations to 100000.

ϵ	zero		CR		Ours	
	time (s)	# iter.	time (s)	# iter.	time (s)	# iter.
1e-1	275.09 ($\pm$ 166.59)	86755.53 ($\pm$ 28222.26)	284.74 ($\pm$ 184.58)	85899.10 ($\pm$ 29126.65)	**271.73** ($\pm$ 168.97)	**84882.55** ($\pm$ 29704.22)
1e-2	281.00 ($\pm$ 168.63)	88175.07 ($\pm$ 27438.57)	291.40 ($\pm$ 186.65)	87513.27 ($\pm$ 28093.75)	**278.55** ($\pm$ 171.83)	**86526.59** ($\pm$ 29003.66)
1e-3	281.00 ($\pm$ 168.64)	88176.12 ($\pm$ 27439.02)	291.66 ($\pm$ 186.67)	87605.24 ($\pm$ 28110.87)	**278.58** ($\pm$ 171.82)	**86540.76** ($\pm$ 29003.68)
1e-4	281.00 ($\pm$ 168.64)	88176.12 ($\pm$ 27439.02)	291.66 ($\pm$ 186.67)	87605.24 ($\pm$ 28110.87)	**278.58** ($\pm$ 171.82)	**86540.76** ($\pm$ 29003.68)

ticularly for non-zero initialization. However, this should be done specifically for each initialization (and possibly for each instance), which is beyond the scope of this work.