INTEGER PROGRAMMING APPROACHES FOR SEMICONTINUOUS AND STOCHASTIC OPTIMIZATION

A Thesis
Presented to
The Academic Faculty

by

Gustavo I. Angulo Olivares

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2014

INTEGER PROGRAMMING APPROACHES FOR SEMICONTINUOUS AND STOCHASTIC OPTIMIZATION

Approved by:

Professor Shabbir Ahmed,
Co-advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Santanu S. Dey,
Co-advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor George Nemhauser
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Volker Kaibel
Institut für Mathematische Optimierung
Otto-von-Guericke-Universität Magdeburg

Dr. Myun-Seok Cheon
Corporate Strategic Research
ExxonMobil Research and Engineering

Date Approved: 26 March 2014

To my wife, Tamara.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors, Shabbir Ahmed and Santanu Dey, for their guidance and support throughout these years at Georgia Tech. I feel very fortunate of having had the opportunity of working with them and learning from them.

I would like to thank the members of the thesis committee for their willingness to be part of this process. I am in debt to George Nemhauser for his classes on integer programming and his support during my first year. I also owe quite a lot to Volker Kaibel for his collaboration in the forbidden-vertices problem. I also thank Myun-Seok Cheon for his support and collaboration with ExxonMobil.

Also, many thanks to Pam Morrison and Yvonne Smith for the many many times they helped me with paperwork and information. They are efficiency at its best!

Of course, I am extremely grateful for having had the chance of meeting such a great group of students. I knew some before coming to Tech, but I met most of them just here. Some are from Chile too, but many others are not. I met their families and friends here, and they became my friends and part of my family too. I just prefer not to give specific names, but you know who you are and I just wanted to say thank you so much!

And above everything, all my gratitude and recognition to my loved and exceptional wife, Tamara, for her love, support, patience, and encouragement throughout our years here. We made it possible together!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis concerns the application of mixed-integer programming techniques to solve special classes of network flow problems and stochastic integer programs. We draw tools from complexity and polyhedral theory to analyze these problems and propose improved solution methods.

In the first part, we consider semi-continuous network flow problems, that is, a class of network flow problems where some of the variables are required to take values above a prespecified minimum threshold whenever they are not zero. These problems find applications in management and supply chain models where orders in small quantities are undesirable. We introduce the semi-continuous inflow set with variable upper bounds as a relaxation of general semi-continuous network flow problems. Two particular cases of this set are considered, for which we present complete descriptions of the convex hull in terms of linear inequalities and extended formulations. We also consider a class of semi-continuous transportation problems where inflow systems arise as substructures, for which we investigate complexity questions. Finally, we study the computational efficacy of the developed polyhedral results in solving randomly generated instances of semi-continuous transportation problems.

In the second part, we introduce and study the forbidden-vertices problem. Given a polytope P and a subset X of its vertices, we study the complexity of optimizing a linear function on the subset of vertices of P that are not contained in X. This problem is closely related to finding the k-best basic solutions to a linear problem and finds applications in stochastic integer programming. We observe that the complexity of the problem depends on how P and X are specified. For instance, P can be explicitly given by its linear description, or implicitly by an oracle. Similarly, X can be explicitly given as a list of vectors, or implicitly as a face of P. While removing vertices turns to be hard in general, it is tractable for tractable 0-1 polytopes, and compact extended formulations can be obtained. Some extensions to integral polytopes are also presented.

The third part is devoted to the integer L-shaped method for two-stage stochastic integer programs. A widely used model assumes that decisions are made in a two-step fashion, where first-stage decisions are followed by second-stage recourse actions after the uncertain parameters are observed, and we seek to minimize the expected overall cost. In the case of finitely many possible outcomes or scenarios, the integer L-shaped method proposes a decomposition scheme akin to Benders' decomposition for linear problems, but where a series of mixed-integer subproblems have to be solved at each iteration. To improve the performance of the method, we devise a simple modification that alternates between linear and mixed-integer subproblems, yielding significant time savings in instances from the literature. We also present a general framework to generate optimality cuts via a cut-generating problem. Using an extended formulation of the forbidden-vertices problem, we recast our cut-generating problem as a linear problem and embed it within the integer L-shaped method. Our numerical experiments suggest that this approach can prove beneficial when the first-stage set is relatively complicated.

CHAPTER I

INTRODUCTION

A mixed-integer program is a mathematical optimization problem of the form

$$\begin{aligned} \text{(MIP)} \quad \min \quad & cx \\ \text{s.t.} \quad & Ax \geq b \\ & x \in \mathbb{R}^n \times \mathbb{Z}^p. \end{aligned}$$

Here the system of linear inequalities $Ax \geq b$ is coupled with the mixed-integer restrictions $x \in \mathbb{R}^n \times \mathbb{Z}^p$, leading to a usually complicated nonconvex set S. Among all vectors x in S, we seek the one with the smallest value with respect to the objective function c.

For more than fifty years, mixed-integer programming has drawn the attention of researchers and practitioners because of its elegant theory and modeling power. Following the seminal works of Dantzig [13] [12] and Gomory [24] [25], our understanding of linear and mixed-integer programming had led to the development of algorithms and software that have become core components in many industrial applications [2] [32] [19]. However, depending on the characteristics of the problem at hand, mixed-integer programs might still prove very challenging to state-of-the-art solvers. As challenge and curiosity are a irresistible combination for researchers, substantial progress has been done in the theory, computation, and practice of mixed-integer programming in the last decades. In this thesis, motivated by an application to the oil industry, we make further contributions to the understanding of mixed-integer programming that can have practical impact on solving methods.

1.1 *A motivating example*

The pooling problem arising in the oil industry has been extensively studied in the literature [3] [45] [57] [29] . In a simple setting, it can be described as follows. There are a number of inputs, outputs, and pools. Inputs are connected to outputs either directly or via a pool, thus defining a directed network. At each input we have some type of crude, which is characterized by its concentration of a given set of components and its purchase price per unit. Crudes leaving the inputs are mixed at pools and outputs, thus yielding different blends with varying compositions. The final blend at an output has lower and upper bounds on the concentration of each component and has a given selling price per unit. The objective is to find a flow in the network subject to capacity constraints so that the final blends meet the specifications and the profit is maximized. A graphical representation is given in Figure 1.

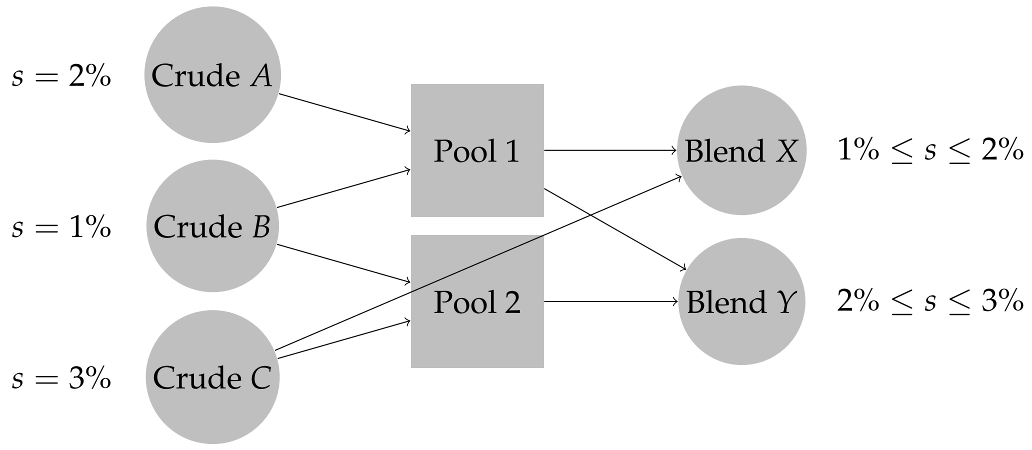


Figure 1: The pooling problem with a single specification s.

As a consequence of blending, the pooling problem yields an optimization problem with bilinear constraints for which different exact nonlinear formulations exist [27] [6] [57] [3]. Recent work [18] shows that reasonably good approximations of the pooling problem can be achieved with mixed-integer formulations. Since the technology for solving mixed-integer programs is at a rather mature stage, it is plausible to tackle the pooling problem, and perhaps other nonlinear problems, with this machinery.

There are also other considerations that prompt the use of mixed-integer programming techniques. The presence of discrete or binary decisions is one of them. In the case of the pooling problem, or network flow problems in general, we often find situations where the flows must take values above a given minimum threshold whenever they are nonzero. A natural way of modeling this restriction is with the use of binary variables, leading to mixed-integer formulations. However, it is also sensible to investigate formulations that depart from the usual binary formulation and study how they can be exploited to improve solution methods.

Another reason to consider mixed-integer formulations is having parameters that are not completely known within the data defining the problem. For instance, in the pooling problem, the future selling price of the final blends might not be known at the time the crudes are acquired. If we have to make purchase decisions now, we can wait until the true selling prices are revealed, assuming the lead times allow us to do this, and then decide the quantities of each blend to be produced. This dynamic is commonly found in problems where planning decisions such as investments are followed by operational decisions such as weekly production and leads to the notion of a two-stage stochastic program. Given the existence of a number of methods to solve stochastic integer programs, it is reasonable to consider casting the uncertain problem within this framework and look for efficient approaches to deal with the resulting model. At this point, the question of whether existent computational methods can be improved with the aid of polyhedral theory and efficient implementations arises.

1.2 *Some important tools in mixed-integer programming*

In order to succeed in solving (MIP), having tight formulations for S is of great importance. By a formulation, we mean a linear description of a polyhedron P such that $P \cap (\mathbb{R}^n \times \mathbb{Z}^p) = S$. In other words, P contains all feasible solutions to (MIP), but no other point in $\mathbb{R}^n \times \mathbb{Z}^p$ lies in P. Letting $\operatorname{conv}(S)$ denote the convex hull of S, by definition we have $\operatorname{conv}(S) \subseteq P$. A formulation is ideal if $\operatorname{conv}(S) = P$. In many cases though, $\operatorname{conv}(S)$ can prove extremely difficult to describe, in particular because of the large number of linear inequalities required. Thus we have to rely on techniques that allows us to obtain formulations that are as close as possible to $\operatorname{conv}(S)$, but still manageable from a practical perspective.

1.2.1 Cutting planes

A valid inequality, or cutting plane, is a linear inequality $\pi x \leq \pi_0$ that is satisfied by all x in S, and thus, by all $x \in \operatorname{conv}(S)$. The idea is to start with some formulation for S and then to add valid inequalities to strengthen it and, in some sense, get closer to $\operatorname{conv}(S)$. Together with branch-and-bound, this technique constitutes one of the main backbones of current solvers. Figure 2 illustrates an approximation of $\operatorname{conv}(S)$ using cutting planes $\pi^1 x \leq \pi_0^1$ and $\pi^2 x \leq \pi_0^2$.

While solvers can derive valid inequalities from the linear system defining S, this is usually done based on generic templates of substructures or relaxations of S that are commonly found in mixed-integer programs. We can exploit the specific knowledge that we might have about S to obtain stronger cutting planes for $\operatorname{conv}(S)$, i.e., valid inequalities that remove a greater portion of P that lies outside $\operatorname{conv}(S)$. For that, we usually drop some constraints and merge variables to obtain a relaxation of S that is simpler to analyze, yet retains structural information about S. With this simplification, we can derive separation routines in the form of combinatorial algorithms or cut-generating linear programs. In the latter case,

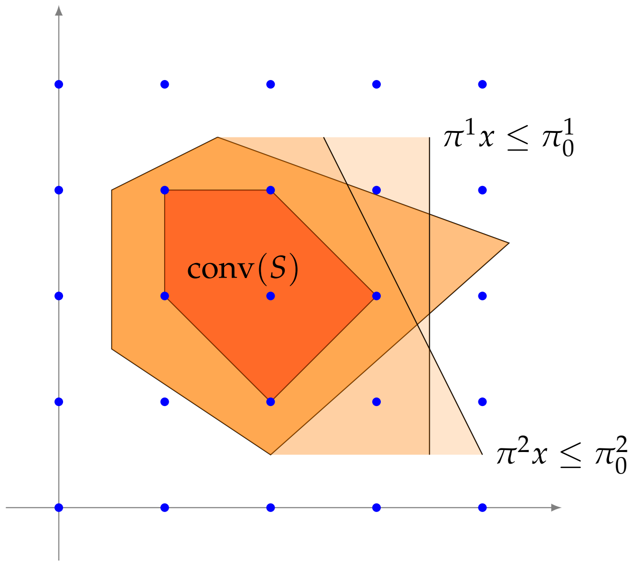


Figure 2: Cutting planes.

sometimes it is possible to embed the cut-generating linear program within the original formulation, yielding a new formulation where all cuts that could be derived with the separation routine are already implied from the beginning. This is an example of an extended formulation which is described next.

1.2.2 Extended formulations

There are many examples of integer programs for which the convex hull of its feasible region is completely known, but its description involves exponentially many linear inequalities. For example, consider the parity polytope PAR_n defined as the convex hull of all n-dimensional binary vectors having an even number of ones, i.e.,

$$\mathrm{PAR}_n := \mathrm{conv}\left\{ x \in \{0,1\}^n : \sum_{i=1}^{n} x_i \text{ is even} \right\}.$$

In [28], it is shown that PAR_n is completely described by the system

$$\begin{aligned} \sum_{i \in S} x_i + \sum_{i \notin S} (1 - x_i) \geq 1 \quad & \forall S \subseteq \{1, \dots, n\},\ |S| \text{ odd} \\ 0 \leq x_i \leq 1 \quad & \forall i \in \{1, \dots, n\}. \end{aligned}$$

Although the system has 2^{n-1} nontrivial inequalities, it is also the projection onto the x-space of the simpler polyhedron given by

$$\begin{aligned}
-x+y+z &= 0 \\
q_1+y_1 &= 1 \\
p_1+z_1 &= 0 \\
p_{i+1}+z_{i+1}-p_i-y_i &= 0 \quad \forall i=1,\dots,n-1 \\
q_{i+1}+y_{i+1}-q_i-z_i &= 0 \quad \forall i=1,\dots,n-1 \\
q_n+z_n &= 1 \\
p_n+y_n &= 0 \\
p,q,y,z \geq 0.
\end{aligned}$$

We have included $4n$ additional variables, but the number of nontrivial constraints has dramatically decreased from 2^{n-1} to just $3n$. The above system can be regarded as the description of the network flow depicted in Figure 3 below, where one unit of flow travels from left to right, forcing the vector $y+z$ to have an even number of ones.

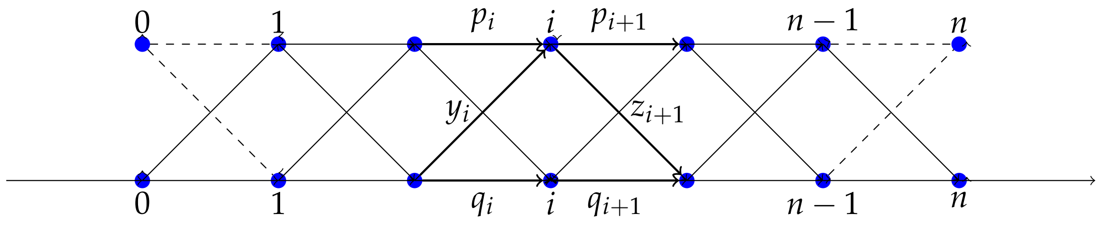


Figure 3: Network flow representation of the parity polytope.

More formally, an extended formulation of a polyhedron P is a linear description of another higher-dimensional polyhedron that can be linearly mapped onto P. A natural question that arises is that of the size of extended formulations. For our purposes, by size we mean the number of inequalities in the extended formulation. (It can be shown [21] that counting also the number of variables and equalities leads to notions of size that are basically equivalent.) We say that an extended formulation is compact if its size is polynomially bounded with respect to the size of some natural encoding of P, which in many cases reduces to the dimension of this polyhedron. Having extended formulations of small size allows us to prove tractability of some combinatorial problems and to derive separation routines for the convex hull of the feasible set of (MIP).

1.2.3 Decomposition

Another important tool in mixed-integer programming is the application of decomposition methods to tackle large-scale well-structured problems. The key idea to reformulate the original problem so that it can be divided into smaller pieces that are solved separately. A prime example are decomposition methods for two-stage stochastic linear and mixed-integer programs.

A two-stage stochastic integer program (SIP) has the form

$$\begin{aligned} \text{(SIP)} \quad \min \quad & cx + Q(x) \\ \text{s.t.} \quad & Ax \geq b \\ & x \in \mathbb{R}^n \times \mathbb{Z}^p, \end{aligned}$$

where $Q(x)$ is the expected second-stage cost associated to first-stage decision variables x. We usually have $Q(x) = \mathbb{E}_\xi[Q_\xi(x)]$, where

$$\begin{aligned} Q_\xi(x) := \quad \min \quad & qy \\ \text{s.t.} \quad & Wy \geq h - Tx \\ & y \in \mathbb{R}^s \times \mathbb{Z}^t \end{aligned}$$

and the expectation is taken over the random data $\xi = (q, W, T, h)$. Variables $y = y_\xi$ are referred to as recourse actions as they allow us to react given first-stage decision x and outcome ξ. When ξ follows a discrete distribution with a finite number of outcomes or scenarios, (SIP) can be formulated as

$$\begin{aligned} \min \quad & cx + \sum_{\xi} p_\xi q_\xi y_\xi \\ \text{s.t.} \quad & Ax \geq b \\ & T_\xi x + W_\xi y_\xi \geq h_\xi \ \forall \xi \\ & x \in \mathbb{R}^n \times \mathbb{Z}^p \\ & y_\xi \in \mathbb{R}^s \times \mathbb{Z}^t \ \forall \xi, \end{aligned}$$

where p_ξ denotes the probability of occurrence of outcome or scenario ξ. Thus, (SIP) reduces to a large-scale mixed-integer program. However, this formulation can quickly become challenging as the number of scenarios increases and decomposition approaches must be sought. In the absence of integrality requirements on first- and second-stage variables, (SIP) reduces to a large-scale linear program for which Benders' decomposition is applicable.

In Benders' decomposition, we introduce an additional variable θ, yielding the equivalent formulation of (SIP)

$$\begin{aligned} \min \quad & cx + \theta \\ \text{s.t.} \quad & Ax \geq b \\ & \theta \geq Q(x) \\ & x \in \mathbb{R}^n \\ & \theta \in \mathbb{R}. \end{aligned}$$

For simplicity, let us assume that $Q(x)$ is finite for any x satisfying $Ax \geq b$. By strong duality of linear programming, we have

$$\begin{aligned} Q(x) = \quad \max \quad & (h - Tx)\pi \\ \text{s.t.} \quad & \pi W = q \\ & \pi \in \mathbb{R}^m_+. \end{aligned}$$

Moreover, we have $Q(x) = \max\{(h - Tx)\pi : \pi \in \Pi\}$ with Π being the set of extreme points of the polyhedron $\{\pi \in \mathbb{R}^m_+ : \pi W = q\}$. Thus, we arrive at the equivalent formulation of (SIP)

$$\begin{array}{lll} \text{(BD)} & \min & cx + \theta \\ & \text{s.t.} & Ax \geq b \\ & & \theta \geq (h - Tx)\pi \quad \forall \pi \in \Pi \\ & & x \in \mathbb{R}^n \\ & & \theta \in \mathbb{R}. \end{array}$$

Since Π might have a large number of elements, we proceed as follows. Let $L \in \mathbb{R}$ be a lower bound for $Q(x)$. Let $\Pi_k \subseteq \Pi$ with $\Pi_0 = \varnothing$. Starting with $k = 0$, we solve the relaxation (BD_k) below

$$\begin{array}{lll} (\text{BD}_k) & \min & cx + \theta \\ & \text{s.t.} & Ax \geq b \\ & & \theta \geq (h - Tx)\pi \quad \forall \pi \in \Pi_k \\ & & \theta \geq L \\ & & x \in \mathbb{R}^n \\ & & \theta \in \mathbb{R}. \end{array}$$

Given an optimal solution (x^*, θ^*) to (BD_k), we compute $Q(x^*)$. If $\theta^* \geq Q(x^*)$, then the solution is feasible to (BD), and therefore it is optimal to the true problem. Otherwise, let π^* be an optimal solution to $\max\{(h - Tx^*)\pi : \pi \in \Pi\}$. We have $Q(x^*) = (h - Tx^*)\pi^* > \theta^*$, i.e., (x^*, θ^*) violates the inequality $\theta \geq (h - Tx)\pi^*$. Thus, we set $\Pi_{k+1} = \Pi_k \cup \{\pi^*\}$ and we solve (BD_{k+1}) knowing that (x^*, θ^*) is not feasible. We repeat this procedure until an optimal solution to (BD) is identified. Finite convergence is guaranteed by finiteness of Π.

1.3 *Overview*

This thesis concerns the application of mixed-integer programming techniques to solve special classes of network flow problems and stochastic integer programs. We draw tools from complexity and polyhedral theory to analyze these problems and propose improved solution methods.

First, in Chapter 2, we consider semi-continuous network flow problems, that is, a class of network flow problems where some of the variables are required to belong to a set of the form $\{0\} \cup [l, u]$ for some $0 \leq l \leq u$. These problems find applications in management and supply chain models where orders in small quantities are undesirable. We introduce the semi-continuous inflow set with variable upper bounds as a relaxation of general semi-continuous network flow problems. Two particular cases of this set are considered, for which we present complete descriptions of their convex hulls in terms of linear inequalities and extended formulations. We also consider a class of semi-continuous transportation problems where inflow systems arise as substructures, for which we investigate complexity questions. Finally, we study the computational efficacy of the developed polyhedral results in solving randomly generated instances of semi-continuous transportation problems.

In Chapter 3, motivated by an application to stochastic integer programming, we introduce and study the forbidden-vertices problem. Given a polytope P and a subset X of its vertices, we want to understand the complexity of optimizing a linear function on the subset of vertices of P that are not contained in X. We observe that the complexity of the problem depends on how P and X are specified. For instance, P can explicitly given by its linear description, or implicitly by an oracle. Similarly, X can be explicitly given as a list of vectors, or implicitly as a face of P, for example. While removing vertices turns to be hard in general, it is tractable for tractable 0-1 polytopes, and compact extended formulations can be obtained. Some extensions to integral polytopes are also presented.

Finally, in Chapter 4, we address computational improvements for the integer L-shaped method for two-stage stochastic programs with integer recourse. This method proposes a decomposition scheme akin to Benders' decomposition for linear problems, but where a series of mixed-integer subproblems have to be solved at each iteration. To improve the performance of the method, we devise a simple modification that alternates between linear and mixed-integer subproblems, yielding significant time savings in solving instances from the literature. We also present a general framework to generate optimality cuts via a cut-generating problem. Using an extended formulation of the forbidden-vertices problem, we recast our cut-generating problem as a linear problem and embed it within the integer L-shaped method. Our numerical experiments suggest that this approach can prove beneficial when the first-stage set is relatively complicated.

CHAPTER II

SEMI-CONTINUOUS NETWORK FLOW PROBLEMS

2.1 *Introduction*

A variable x is said to be semi-continuous if x is required to belong to a set of the form $\{0\} \cup [l, u]$ for some $0 \leq l \leq u$. We call l and u lower and upper bounds of x, respectively. Note that in this definition we consider continuous variables as a special case where $l = 0$. A semi-continuous variable can be regarded as a generalization of a binary variable. In fact, by setting $l = u = 1$ in the above definition, we have that x is binary. As such, the presence of these variables may lead to hard optimization problems.

Semi-continuous variables appear in models for inventory management where shippings from a given supplier are required to be between prestablished minimum and maximum quantities whenever an order is placed [58]. In portfolio optimization, semi-continuous constraints are known as minimum transaction levels, and are studied in [8] and [50]. Semi-continuous variables are also common when modeling petrochemical processes as described in [29] and [30]. Furthermore, as [30] and [58] suggest, supply chain models may involve network flow structures with semi-continuity constraints on flow variables whenever production, purchases, and shipping in low quantities are undesirable from the operational point of view.

Although semi-continuity can be modeled by means of introducing additional binary variables and constraints, this approach may have some drawbacks. We increase the size of the problem at hand, which can already be large-scale. Additionally, the presence of binary variables may lead to unnecessary branching decisions and large LP relaxations in a branch-and-bound procedure. On the other hand, models that incorporate auxiliary binary variables may benefit from presolve and bound tightening procedures available in state-of-the-art MIP solvers such as CPLEX and may be solved efficiently. To overcome difficulties with auxiliary binary variables, branching rules and cuts without the use of binary variables for some combinatorial problems have been studied in [15] and [16]. In particular, in [14] and [17] the semi-continuous knapsack problem is introduced and cutting-planes are presented.

In this chapter, we study some particular semi-continuous sets. Specifically, given their wide applicability, we focus on network flow problems having semi-continuous flow variables. Our main contributions are complete descriptions of the convex hull of two particular cases of a semi-continuous inflow set with variable upper bounds and a computational study of the effectiveness of the derived inequalities on a class of semi-continuous transportation problems. We observe that the polyhedral results derived from the semi-continuous sets can significantly improve the performance of both the semi-continuous and the standard mixed integer formulation involving auxiliary binary variables. The rest of the chapter is organized as follows. In Section 2.2 we introduce the semi-continuous inflow set along with some basic properties. In Sections 2.3 and 2.4 we present polyhedral studies of two particular cases of this set. Then, in Section 2.5 we introduce a class of semi-continuous transportation problems for which we give complexity results. We devote Section 2.6 to computational results regarding the performance of the polyhedral results when solving semi-continuous transportation problems. Finally, in Section 2.7 we conclude with some remarks.

2.2 *The semi-continuous inflow set*

Consider the network substructure shown in Figure 4. Let $N := \{1, \dots, n\}$ be a set of nodes, where $n \geq 2$, and let $d > 0$ be the required total flow from nodes in N to another node 0. For $i \in N$, let y_i be the flow from node i to node 0, and x_i be the flow into node i. Let l_i and h_i be the lower bounds on flows x_i and y_i whenever these variables are positive. Let t_i be the exogenous supply into node i. The semi-continuous inflow set with variable upper bounds is the set $S(t, h) \subseteq \mathbb{R}^n \times \mathbb{R}^n$ defined as

$$S(t,h) := \left\{ (x,y) \in \mathbb{R}^n \times \mathbb{R}^n : \begin{array}{rlr} \sum_{i \in N} y_i \geq d & & (1) \\ y_i \leq t_i + x_i & \forall i \in N & (2) \\ x_i \in \{0\} \cup [l_i, \infty) & \forall i \in N & (3) \\ y_i \in \{0\} \cup [h_i, \infty) & \forall i \in N & (4) \end{array} \right\}.$$

Constraint (1) ensures that the minimum total inflow into the node 0 is met. Constraints (2) bound y_i by the total available inflow $t_i + x_i$ for node $i \in N$. Finally, constraints (3) and (4) are semi-continuity requirements on x and y, respectively. As we shall see, the structure and tractability of the above set are essentially determined by t and h only, and therefore the dependence on l is not made explicit in the notation.

Next we discuss how the set $S(t, h)$ arises as a substructure in general semi-continuous network flow problems. Consider a network represented by a directed graph $G = (V, E)$, where each node $v \in V$ satisfies a constraint of the form

$$\sum_{u \in V^+(v)} f_{vu} - \sum_{u \in V^-(v)} f_{uv} = d_v, \tag{5}$$

where variable $f_{vu} \geq 0$ is the flow through the arc $(v, u) \in E$, $V^+(v) := \{u \in V : (v, u) \in E\}$, $V^-(v) := \{u \in V : (u, v) \in E\}$, and d_v is a given real parameter. Suppose that $f_{uv} \in \{0\} \cup [l_{uv}, u_{uv}]$, that is, f_{uv} is semi-continuous. We refer to

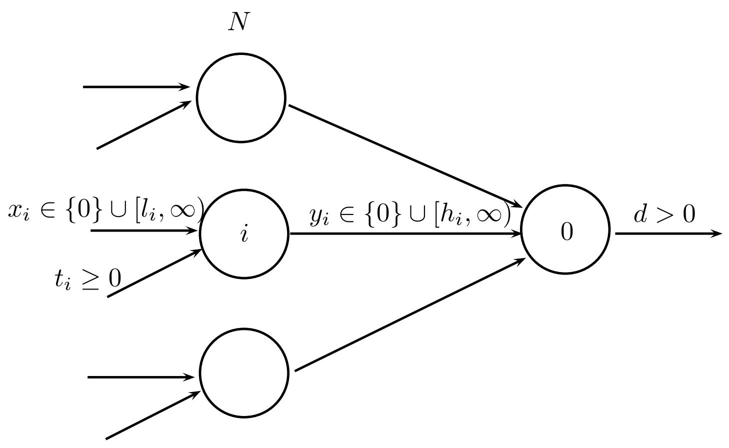

N
$x_i \in \{0\} \cup [l_i, \infty)$
$t_i \geq 0$
i
$y_i \in \{0\} \cup [h_i, \infty)$
0
$d > 0$

Figure 4: Inflow relaxation.

such problems as semi-continuous network flow problems. We obtain $S(t, h)$ as a relaxation as follows.

Consider a node $v \in V$ with $d_v < 0$ as depicted in Figure 5. Since the first sum in (5) is nonnegative, we have

$$\sum_{u \in V^-(v)} f_{uv} \geq -d_v = |d_v|,$$

which has the form of the semi-continuous knapsack set introduced in [14]. However, since we are dealing with a network flow problem, there is more structure to be exploited when looking for tighter relaxations. Indeed, consider $(u, v) \in E$. Then $v \in V^+(u)$ and (5) applied to u can be written as

$$\sum_{w \in V^+(u) \setminus \{v\}} f_{uw} + f_{uv} - \sum_{w \in V^-(u)} f_{wu} = d_u. \tag{6}$$

As before, since the first sum in (6) is nonnegative, we arrive at

$$f_{uv} \leq \sum_{w \in V^-(u)} f_{wu} + d_u \leq \sum_{w \in V^-(u)} f_{wu} + \max\{d_u, 0\}. \tag{7}$$

Note that $f_u := \sum_{w \in V^-(u)} f_{wu}$ is a semi-continuous variable taking values in $\{0\} \cup [l_u, u_u]$, where $l_u := \min_{w \in V^-(u)}\{l_{wu}\}$ and $u_u := \sum_{w \in V^-(u)} u_{wu}$. We obtain the system

$$
\begin{aligned}
\sum_{u \in V^-(v)} f_{uv} &\geq |d_v| \\
f_{uv} &\leq f_u + \max\{d_u, 0\} \quad \forall u \in V^-(v) \\
f_u &\in \{0\} \cup [l_u, u_u] \quad \forall u \in V^-(v) \qquad (8) \\
f_{uv} &\in \{0\} \cup [l_{uv}, u_{uv}] \quad \forall u \in V^-(v), \qquad (9)
\end{aligned}
$$

which is a relaxation for the original network flow set. Finally, removing the upper bounds from (8) and (9) we arrive at a relaxation having the form of $S(t, h)$.

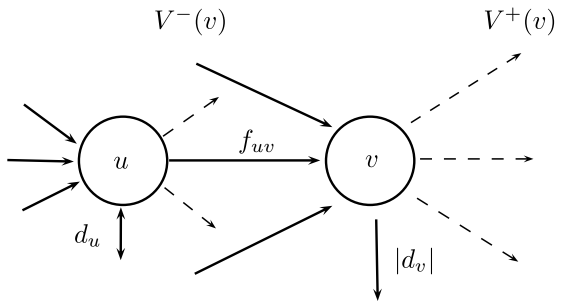


Figure 5: Nodes v and u, where $u \in V^-(v)$ and $d_v < 0$.

A similar approach can be followed when $d_v > 0$, in which case we drop the second sum in (5) and relax the balance equation for nodes in $V^+(v)$. In either case, by appropriately manipulating (5) applied to $v \in V$ and $u \in V^+(v) \cup V^-(v)$, we obtain the set $S(t, h)$ as a relaxation.

We omit the case $d = 0$ since (1) becomes redundant and then $S(t, h)$ is the product of n simple 2-dimensional sets.

2.2.1 Complexity of optimization

It is not difficult to verify that having finite upper bounds as in (8) and (9) would yield a set that is already hard to deal with. We show that in our setting optimization over $S(t, h)$ is intractable.

Proposition 1. *Optimizing a linear function over $S(t, h)$ is $\mathcal{NP}$-hard, even if $l = 0$.*

Proof. We will show that the Binary Knapsack problem, which is $\mathcal{NP}$-hard, can be reduced to optimization of a linear function over $S(t, h)$.

We start with a feasible instance of the Binary Knapsack problem of the form

$$\begin{aligned} \min \quad & \sum_{i \in N} f_i z_i \\ \text{s.t.} \quad & \sum_{i \in N} w_i z_i \geq d \\ & z_i \in \{0, 1\} \quad \forall i \in N, \end{aligned}$$

where $d \in \mathbb{Z}_+$, $w \in \mathbb{Z}_+^n$, and $f \in \mathbb{Z}_+^n$. Consider the change of variables $y_i = w_i z_i$ for all $i \in N$. Given that $z_i \in \{0, 1\}$, we have that $y_i \in \{0, w_i\}$. Furthermore, this is equivalent to requiring $y_i \in \{0\} \cup [w_i, \infty)$ and $y_i \leq w_i$. Thus, the optimal value of the instance is the same as that of

$$\begin{aligned} \min \quad & \alpha y \\ \text{s.t.} \quad & \sum_{i \in N} y_i \geq d \\ & y_i \leq w_i && \forall i \in N \\ & y_i \in \{0\} \cup [w_i, \infty) && \forall i \in N, \end{aligned}$$

where $\alpha_i = \frac{f_i}{w_i}$ for each $i \in N$. Now, consider the problem

$$\begin{aligned} \min \quad & cx + \alpha y \\ \text{s.t.} \quad & \sum_{i \in N} y_i \geq d \\ & y_i \leq w_i + x_i && \forall i \in N \\ & x_i \geq 0 && \forall i \in N \\ & y_i \in \{0\} \cup [w_i, \infty) && \forall i \in N, \end{aligned}$$

where $c_i = M > 0$ for all $i \in N$. Let (x^*, y^*) be an optimal solution and let $N^* := \{i \in N : y_i^* > 0\}$. Given that $c > 0$, we must have $y_i^* = w_i + x_i^*$ for all $i \in N^*$. If $0 < \sum_{i \in N} x_i^* < 1$, then we have

$$d \leq \sum_{i \in N} y_i^* = \sum_{i \in N^*} y_i^* = \sum_{i \in N^*} (w_i + x_i^*)$$

$$\implies d \le \left\lfloor \sum_{i \in N} y_i^* \right\rfloor = \sum_{i \in N^*} w_i = \sum_{i \in N} \lfloor y_i^* \rfloor.$$

Thus, given that $\alpha > 0$, rounding down each component of y^* improves the solution. Hence, either $x^* = 0$ or $\sum_{i \in N} x_i^* \ge 1$. However, if M is sufficiently large, say $M = \sum_{i \in N} \alpha_i w_i = \sum_{i \in N} f_i$, then we must have $x^* = 0$. Therefore, the optimal values of this problem, which is an instance of linear optimization over $S(t,h)$, and the instance of the Binary Knapsack problem we started with are the same. Given that the transformation is polynomial in the original input size, the proof is complete. □

Despite the general complexity result in Proposition 1, there are at least two situations where $S(t,h)$ is tractable, namely when $t_i = 0$ for all $i \in N$ and when $h_i = 0$ for all $i \in N$. Note that the first case is a restriction. The second one is a relaxation as y becomes continuous. These cases will be discussed in Sections 2.3 and 2.4, respectively.

2.2.2 Basic polyhedral results

In [14], the semi-continuous knapsack is introduced. This set is of the form

$$K = \left\{ x \in \mathbb{R}^n : \begin{array}{ll} \sum_{i \in N} w_i x_i \le r & \\ x_i \in [0, p_i] \cup [l_i, u_i] & \forall i \in N^+ \\ x_i \in [0, p_i] \cup [l_i, \infty) & \forall i \in N^+_\infty \cup N^- \end{array} \right\},$$

where N^+, N^+_∞, N^- constitute a partition of N, $w_i > 0$ for all $i \in N^+ \cup N^+_\infty$, and $w_i < 0$ for all $i \in N^-$. Several classes of valid inequalities are presented along with lifting procedures. Note that when $r < 0$ and $N^- = N$, this set is a relaxation of $S(t,h)$ as we can aggregate constraints and arrive at a system having the above form. Thus, valid inequalities for K give rise to valid inequalities for $S(t,h)$. In some cases, a complete description of $\operatorname{conv}(K)$ can be found. In particular, if $N =$

N^-, $p_i = 0$ for each $i \in N$, and $r < 0$, then

$$\operatorname{conv}(K) = \left\{ x \in \mathbb{R}^n : \begin{array}{ll} \sum_{i \in N} \frac{w_i}{\min\{r, w_i l_i\}} x_i \geq 1 & \\ 0 \leq x_i & \forall i \in N \end{array} \right\}.$$

As we shall see, an exponential family of inequalities similar to the one above will suffice to describe $\operatorname{conv}(S(t,h))$ when $t = 0$ or $h = 0$. We first establish some fundamental results regarding $S(t,h)$.

Proposition 2. *$S(t,h)$ is full-dimensional.*

Proof. Consider the point $(\bar{x}, \bar{y}) \in \mathbb{R}^n \times \mathbb{R}^n$ given by $\bar{x}_i = \max\{d, l_i, h_i\} + 1$ and $\bar{y}_i = \max\{d, h_i\}$ for all $i \in N$. We have that $(\bar{x}, \bar{y})$ belongs to $S(t,h)$, and adding any standard unit vector from $\mathbb{R}^n \times \mathbb{R}^n$ to $(\bar{x}, \bar{y})$ yields another point that is also feasible to $S(t,h)$. The collection of such $2n$ points along with $(\bar{x}, \bar{y})$ is an affinely independent set, and therefore $S(t,h)$ is of full dimension. $\square$

Proposition 3. $\operatorname{conv}(S(t,h))$ *is a polyhedron.*

Proof. We prove a more general result of which Proposition 3 is a particular case.

Given an integer $t \geq 1$, let $T := \{1, \ldots, t\}$. For each $r \in T$, consider $\pi^r \in \mathbb{R}^n$ and $\pi_0^r, \pi_1^r \in \mathbb{R}$. We are mainly interested in the case $\pi_0^r < \pi_1^r$, although this is not required in what follows. Given a closed convex set $C \subseteq \mathbb{R}^n$, for each $Q \in \mathcal{T} := 2^T$, consider the set

$$C^Q := \{x \in C : \; \pi^r x \leq \pi_0^r \; \forall r \in Q, \; \pi^r x \geq \pi_1^r \; \forall r \notin Q\}.$$

We call the set $\cup_{Q \in \mathcal{T}} C^Q$ a t-branch split disjunction as defined in [43]. Let

$$C^{\pi, \pi_0, \pi_1} := \operatorname{conv}\left(\cup_{Q \in \mathcal{T}} C^Q\right).$$

We will prove that C^{π, π_0, π_1} in closed for any $t \geq 1$, extending the result in [11] for $t = 1$.

Let C_∞ be the recession cone of C, and for each $Q \in \mathcal{T}$, let $C^Q_\infty := C^Q + C_\infty$. Also, let $\mathcal{T}^* := \{Q \in \mathcal{T} : \ C^Q \neq \varnothing\}$. If $\mathcal{T}^*$ is empty, then the result holds. Thus, assume $\mathcal{T}^*$ is nonempty.

Claim: $C^{\pi,\pi_0,\pi_1} = \text{conv}\left(\cup_{Q\in\mathcal{T}^*} C^Q_\infty\right)$.

The forward inclusion is easy as $\cup_{Q\in\mathcal{T}} C^Q \subseteq \cup_{Q\in\mathcal{T}^*} C^Q_\infty$.

For the reverse inclusion, consider $x \in \text{conv}\left(\cup_{Q\in\mathcal{T}^*} C^Q_\infty\right)$. We can write $x = \sum_{Q\in\mathcal{T}^*} \lambda^Q (x^Q + y^Q)$, where $x^Q \in C^Q$, $y^Q \in C_\infty$, and $\lambda^Q \geq 0$ for each $Q \in \mathcal{T}^*$, and $\sum_{Q\in\mathcal{T}^*} \lambda^Q = 1$. If we show that for any $Q \in \mathcal{T}^*$, $x^Q + y^Q$ belongs to C^{π,π_0,π_1}, then the result follows. To that end, fix $Q \in \mathcal{T}^*$ and let

$$R^- := \{r \in T : \ \pi^r y^Q < 0\},$$

$$R^+ := \{r \in T : \ \pi^r y^Q > 0\},$$

$$R^= := \{r \in T : \ \pi^r y^Q = 0\}.$$

Note that there exists finite $\lambda \geq 1$ such that $\pi^r(x^Q + \lambda y^Q) \leq \pi^r_0$ for all $r \in R^-$ and $\pi^r(x^Q + \lambda y^Q) \geq \pi^r_1$ for all $r \in R^+$. Also, recall that x^Q satisfies $\pi^r x^Q \leq \pi^r_0$ for all $r \in Q$ and $\pi^r x^Q \geq \pi^r_1$ for all $r \notin Q$. Thus $x^Q + \lambda y^Q$ belongs to $C^{Q'}$, where $Q' := R^- \cup (R^= \cap Q)$. Finally, note that $x^Q + y^Q \in \text{conv}(\{x^Q, x^Q + \lambda y^Q\})$, which implies $x^Q + y^Q \in C^{\pi,\pi_0,\pi_1}$ as desired. $\diamond$

By the claim, C^{π,π_0,π_1} is the convex hull of the union of nonempty closed convex sets having the same recession cone. By Corollary 9.8.1 of [53], C^{π,π_0,π_1} is a closed convex set. Moreover, if C is a polyhedron, then C^{π,π_0,π_1} is the convex hull of the union of nonempty polyhedra having the same recession cone, which is a polyhedron [5]. $\square$

We now proceed to identify the trivial facets of $\text{conv}(S(t,h))$.

Proposition 4. *For each each $i \in N$, $y_i \geq 0$ and $y_i \leq t_i + x_i$ are facet-defining for* $\text{conv}(S(t,h))$. *In addition, $x_i \geq 0$ is facet-defining if and only if $t_i > 0$.*

Proof. Let $i \in N$. Choose a point $\bar{x} \in \mathbb{R}^n$ satisfying $\bar{x}_j > \max\{d, l_j, h_j\}$ for all $j \in N$. Set $\bar{y}_i = 0$ and $\bar{y}_j = \bar{x}_j$ for all $j \in N$, $j \neq i$. We have that $(\bar{x}, \bar{y})$ belongs to $S(t, h)$. Now for each $j \in N$, $j \neq i$, consider the points $\left(x^j, y^j\right)$ and $\left(x^{n+j}, y^{n+j}\right)$ given by

$$\left(x_k^j, y_k^j\right) = \begin{cases} (\bar{x}_j + \epsilon, \bar{y}_j) & k = j \\ (\bar{x}_k, \bar{y}_k) & k \neq j, \end{cases}$$

$$\left(x_k^{n+j}, y_k^{n+j}\right) = \begin{cases} (\bar{x}_j, \bar{y}_j - \epsilon) & k = j \\ (\bar{x}_k, \bar{y}_k) & k \neq j. \end{cases}$$

Finally, let $\left(x^i, y^i\right) = (\bar{x}, \bar{y})$ and let $\left(x^{n+i}, y^{n+i}\right)$ be given by

$$\left(x_k^{n+i}, y_k^{n+i}\right) = \begin{cases} (\bar{x}_i + \epsilon, \bar{y}_i) & k = i \\ (\bar{x}_k, \bar{y}_k) & k \neq i. \end{cases}$$

For $\epsilon > 0$ sufficiently small, the collection $\left\{\left(x^j, y^j\right), \left(x^{n+j}, y^{n+j}\right) : j \in N\right\}$ is contained in $S(t, h)$. Moreover, it is an affinely independent set, and since these $2n$ points satisfy $y_i \geq 0$ at equality, this constraint defines a facet of $\text{conv}(S(t, h))$. The proof for $y_i \leq t_i + x_i$ is analogous by setting $\bar{y}_i = t_i + \bar{x}_i$ and defining $\left(x^{n+i}, y^{n+i}\right)$ as

$$\left(x_k^{n+i}, y_k^{n+i}\right) = \begin{cases} (\bar{x}_i + \epsilon, \bar{y}_i + \epsilon) & k = i \\ (\bar{x}_k, \bar{y}_k) & k \neq i. \end{cases}$$

For the last part, if $t_i > 0$, set $\bar{x}_i = 0$ and $\bar{y}_i = t_i$. Again, the proof is similar by defining $\left(x^{n+i}, y^{n+i}\right)$ as

$$\left(x_k^{n+i}, y_k^{n+i}\right) = \begin{cases} (\bar{x}_i, \bar{y}_i - \epsilon) & k = i \\ (\bar{x}_k, \bar{y}_k) & k \neq i. \end{cases}$$

Finally, note that if $t_i = 0$, then $x_i \geq 0$ is dominated by $y_i \geq 0$. $\square$

In the following two sections we turn our attention to polyhedral results for $S(0, h)$ and $S(t, 0)$, respectively.

2.3 *The case* $t = 0$

In this section we assume that $t = 0$, and therefore $S(0,h) \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is the set of vectors (x, y) satisfying

$$\sum_{i \in N} y_i \geq d \tag{10}$$

$$y_i \leq x_i \quad \forall i \in N \tag{11}$$

$$x_i \in \{0\} \cup [l_i, \infty) \quad \forall i \in N \tag{12}$$

$$y_i \in \{0\} \cup [h_i, \infty) \quad \forall i \in N. \tag{13}$$

2.3.1 Inequality description of conv$(S(0,h))$

Define the sets

$$L := \{i \in N : \max\{d, h_i\} < l_i\},$$

$$H := \{i \in N : h_i \geq d\},$$

and consider the family of inequalities given by

$$\sum_{i \in T} \frac{x_i}{l_i} + \sum_{i \in N \setminus T} \frac{y_i}{\max\{d, h_i\}} \geq 1 \; \forall T \subseteq L. \tag{14}$$

Recalling that $d > 0$, we have that $l_i = 0$ implies $i \in N \setminus L$. Thus, (14) is well-defined for all $T \subseteq L$. Furthermore, note that if $l = 0$, then $L = \varnothing$ and (14) reduces to the single inequality

$$\sum_{i \in N} \frac{y_i}{\max\{d, h_i\}} \geq 1,$$

which in Section 2.2.2 was seen to be the semi-continuous cut derived in [14].

Proposition 5. *For each $T \subseteq L$, (14) is valid and facet-defining for* conv$(S(0,h))$.

Proof. To show validity, consider $(x, y) \in S(0,h)$ and $T \subseteq L$. If for some $i \in T$ we have $x_i > 0$, then $\frac{x_i}{l_i} \geq 1$. If for some $i \in (N \setminus T) \cap H$ we have $y_i > 0$, then $\frac{y_i}{\max\{d,h_i\}} = \frac{y_i}{h_i} \geq 1$. In both cases (14) is satisfied. If none of them occur, then $y_i = 0$

for all $i \in T \cup [(N \setminus T) \cap H]$. Since $(x, y) \in S(0, h)$, we must have $\sum_{i \in N} y_i \geq d$, and therefore

$$\sum_{i \in (N \setminus T) \setminus H} \frac{y_i}{\max\{d, h_i\}} = \sum_{i \in (N \setminus T) \setminus H} \frac{y_i}{d} = \sum_{i \in N} \frac{y_i}{d} \geq 1.$$

Hence, (14) is satisfied in this case as well.

Now, given $T \subseteq L$, we will show that (14) is facet-defining by showing $2n$ affinely independent points in $S(0, h)$ that satisfy (14) at equality. Let $\left(x^i, y^i\right)$, $i = 1, \ldots, 2n$, be such points defined as follows:

If $i \in T$, then

$$\left(x_j^i, y_j^i\right) = \begin{cases} (l_i, \max\{d, h_i\}) & j = i \\ (0, 0) & j \neq i, \end{cases}$$

$$\left(x_j^{n+i}, y_j^{n+i}\right) = \begin{cases} (l_i, \max\{d, h_i\} + \epsilon) & j = i \\ (0, 0) & j \neq i. \end{cases}$$

If $i \in N \setminus T$, then

$$\left(x_j^i, y_j^i\right) = \begin{cases} (\max\{d, h_i, l_i\}, \max\{d, h_i\}) & j = i \\ (0, 0) & j \neq i, \end{cases}$$

$$\left(x_j^{n+i}, y_j^{n+i}\right) = \begin{cases} (\max\{d, h_i, l_i\} + \epsilon, \max\{d, h_i\}) & j = i \\ (0, 0) & j \neq i. \end{cases}$$

The points previously defined belong to $S(0, h)$ for $\epsilon > 0$ sufficiently small. Finally, $\left\{ \left(x^i, y^i\right), \left(x^{n+i}, y^{n+i}\right) : i \in N \right\}$ is a linearly independent set of points satisfying (14) at equality. Thus this constraint defines a facet of $\text{conv}(S(0, h))$. $\square$

Theorem 6 below shows that all the non-trivial facets of $\text{conv}(S(0, h))$ are given by (14).

Theorem 6. $\operatorname{conv}(S(0,h))$ *is given by the following facet-defining inequalities*

$$\sum_{i \in T} \frac{x_i}{l_i} + \sum_{i \in N \setminus T} \frac{y_i}{\max\{d, h_i\}} \geq 1 \quad \forall T \subseteq L$$

$$y_i \leq x_i \qquad \forall i \in N \tag{15}$$

$$y_i \geq 0 \qquad \forall i \in N. \tag{16}$$

Proof. We already showed that (14) is facet-defining for each $T \subseteq L$, and that (15) and (16) are also facet-defining for each $i \in N$. To show that (14)-(16) completely describe $\operatorname{conv}(S(0,h))$, we apply the technique presented in [44]: it suffices to show that if we optimize any non-zero linear function over $S(0,h)$, then there exists one inequality from (14)-(16) such that all optimal solutions, if one exists, belong to the facet defined by that inequality.

Let $(c, \alpha) \in \mathbb{R}^n \times \mathbb{R}^n$ be a non-zero vector and consider the problem

$$\min \{cx + \alpha y : (x, y) \in S(0,h)\}.$$

Assumption 1: $c \geq 0$ and $c + \alpha \geq 0$.

If for some $i \in N$ we have $c_i < 0$ or $c_i + \alpha_i < 0$, then the problem is unbounded. Thus, we may assume $c \geq 0$, $c + \alpha \geq 0$. $\diamond$

In particular, Assumption 1 implies that the optimal value is nonnegative and that an optimal solution exists. Let (x^*, y^*) be any such solution.

Assumption 2: $\alpha \geq 0$.

If for some $i \in N$, $\alpha_i < 0$, then $y_i^* = x_i^*$, that is, (15) is satisfied as an equality. To see this, suppose that $y_i^* < x_i^*$. If $y_i^* > 0$, then we can increase it and get a better solution. If $y_i^* = 0$, since $x_i^* > 0$ and $c_i > 0$ by Assumption 1 and $\alpha_i < 0$, we can decrease x_i^* to zero and get a better solution. Thus, we may assume $\alpha \geq 0$. $\diamond$

Assumption 3: $c + \alpha > 0$.

Suppose that $c_i = \alpha_i = 0$ for some $i \in N$. Then the optimal value is zero. Since $(c, \alpha) \neq (0, 0)$, by Assumptions 1 and 2, there must exist $j \in N$, $j \neq i$, such that either $\alpha_j > 0$ or $c_j > 0$. By optimality, in the former case we must have $y_j^* = 0$, while in the latter $x_j^* = 0$ must hold. Therefore, either (16) or (15) must be satisfied at equality by all optimal solutions. Thus, we may assume $c + \alpha > 0$. $\diamond$

Claim 1: $y_i^* > 0 \Rightarrow c_i x_i^* + \alpha_i y_i^* > 0$.

If $y_i^* > 0$, then $x_i^* > 0$, and by Assumption 3, $c_i x_i^* + \alpha_i y_i^* > 0$ holds. $\diamond$

Let $T = \{i \in L : \alpha_i = 0\}$. Then $c_i > 0$ for all $i \in T$ by Assumption 3, and $\alpha_i > 0$ for all $i \in L \setminus T$. We claim that

$$\sum_{i \in T} \frac{x_i^*}{l_i} + \sum_{i \in N \setminus T} \frac{y_i^*}{\max\{d, h_i\}} = 1.$$

We prove the claim by contradiction. Let $T^+ = \{i \in T : x_i^* > 0\}$ and $(N \setminus T)^+ = \{i \in N \setminus T : y_i^* > 0\}$. Then

$$\sum_{i \in T^+} \frac{x_i^*}{l_i} + \sum_{i \in (N \setminus T)^+} \frac{y_i^*}{\max\{d, h_i\}} > 1. \tag{17}$$

Claim 2: $T^+ = \varnothing$.

Suppose $i \in T^+$, that is, $x_i^* \geq l_i > \max\{d, h_i\}$. Since $\alpha_i = 0$ and $\alpha_j > 0$ for all $j \in L \setminus T$, by optimality we must have $y_j^* = 0$ for all $j \in L \setminus T$. In addition, by Claim 1, we must have $y_j^* = 0$ for all $j \in N \setminus L$ as well. Thus $(N \setminus T)^+ = \varnothing$. Moreover, given that $c_j > 0$ for any $j \in T^+$, we must have $T^+ = \{i\}$. Then (17) takes the form $x_i^* > l_i$, a contradiction with optimality as $c_i > 0$. $\diamond$

By Claim 2, we arrive at

$$\sum_{i \in (N \setminus T)^+} \frac{y_i^*}{\max\{d, h_i\}} > 1. \tag{18}$$

Claim 3: $(N \setminus T)^+ \cap H = \varnothing$.

Let $i \in (N \setminus T)^+$ be such that $h_i \geq d$. By Claim 1 and optimality, $(N \setminus T)^+ = \{i\}$. Then (18) implies $y_i^* > h_i \geq d$. If $i \in L \setminus T$, then $\alpha_i > 0$ and by optimality we have a contradiction. If $i \in N \setminus L$, then $l_i \leq \max\{h_i, d\} = h_i$. Since $c_i + \alpha_i > 0$, by optimality we must have $y_i^* = h_i$, a contradiction as well. $\diamond$

By Claim 3, we arrive at

$$\sum_{i\in(N\setminus T)^+} y_i^* > d. \tag{19}$$

Claim 4: $|(N \setminus T)^+| \geq 2$.

Since $(N \setminus T)^+$ cannot be empty, suppose $(N \setminus T)^+ = \{i\}$. Then (19) and Claim 3 imply $y_i^* > d > h_i$. Again, if $i \in L \setminus T$, then $\alpha_i > 0$ and we have a contradiction. If $i \in N \setminus L$, then $l_i \leq \max\{h_i, d\} = d$. Since $c_i + \alpha_i > 0$, by optimality we must have $y_i^* = d$, a contradiction as well. $\diamond$

Let

$$i_0 \in \arg\min\left\{c_i + \alpha_i : \ i \in (N \setminus T)^+\right\},$$

and let $i_1 \in (N \setminus T)^+$, $i_1 \neq i_0$, which exists by Claim 4. Recall that from Assumption 3, $c_{i_0} + \alpha_{i_0} > 0$. For $\epsilon > 0$ sufficiently small, define $(\bar{x}, \bar{y})$ as

$$(\bar{x}_i, \bar{y}_i) = \begin{cases} \left(x_{i_0}^* + y_{i_1}^* - \epsilon, y_{i_0}^* + y_{i_1}^* - \epsilon\right) & i = i_0 \\ (0,0) & i = i_1 \\ (x_i^*, y_i^*) & i \neq i_0,\ i \neq i_1. \end{cases}$$

Certainly $\bar{x}_i \geq l_i$ whenever $\bar{x}_i > 0$, $\bar{y}_i \geq h_i$ whenever $y_i > 0$, and $\bar{y}_i \leq \bar{x}_i$ for all $i \in N$. Thus, given that $\sum_{i\in N} y_i^* > d$, we conclude that $(\bar{x}, \bar{y})$ is a feasible solution. Moreover,

$$\begin{aligned} \sum_{i\in N} c_i(x_i^* - \bar{x}_i) + \alpha_i(y_i^* - \bar{y}_i) &= -c_{i_0}(y_{i_1}^* - \epsilon) - \alpha_{i_0}(y_{i_1}^* - \epsilon) + c_{i_1}x_{i_1}^* + \alpha_{i_1}y_{i_1}^* \\ &= -\left(c_{i_0} + \alpha_{i_0}\right)\left(y_{i_1}^* - \epsilon\right) + c_{i_1}x_{i_1}^* + \alpha_{i_1}y_{i_1}^* \\ &> -\left(c_{i_0} + \alpha_{i_0}\right) y_{i_1}^* + c_{i_1}y_{i_1}^* + \alpha_{i_1}y_{i_1}^* \\ &\geq 0, \end{aligned}$$

where the two inequalities follow from $c_{i_0} + \alpha_{i_0} > 0$, $y_{i_1}^* > 0$, and $x_{i_1}^* \geq y_{i_1}^*$, and from the definition of i_0, respectively.

Hence, $(\bar{x}, \bar{y})$ improves upon (x^*, y^*) and we get the required contradiction. $\square$

2.3.2 Extreme points of conv$(S(0,h))$

Since by Theorem 6 an outer description of conv$(S(0,h))$ in terms of linear inequalities is available, we look for an inner description in terms of extreme points.

Proposition 7. *Let (x,y) be an extreme point of* conv$(S(0,h))$*. Then both x and y have exactly one non-zero entry.*

Proof. We claim that if $x_i > 0$, then $y_i > 0$. By contradiction, suppose $x_i > 0$ and $y_i = 0$. We can set

$$(x_i, y_i) = \frac{1}{2}\left[(2x_i, 0) + (0,0)\right].$$

Thus, (x,y) can be written as the average of two distinct points in $S(0,h)$.

Now, suppose that x has more than one non-zero entry, say $x_i > 0$ and $x_j > 0$. By the claim, $y_i > 0$ and $y_j > 0$. We can set

$$h_i \le y_i = \lambda x_i,\ 0 < \lambda \le 1$$

$$h_j \le y_j = \mu x_j,\ 0 < \mu \le 1.$$

Finally, we can write

$$\begin{aligned}(x_i, x_j, y_i, y_j) &= (x_i, x_j, \lambda x_i, \mu x_j)\\ &= \frac{\lambda x_i}{\lambda x_i + \mu x_j}(x_i + \frac{\mu}{\lambda}x_j, 0, \lambda x_i + \mu x_j, 0)\\ &\quad + \frac{\mu x_j}{\lambda x_i + \mu x_j}(0, x_j + \frac{\lambda}{\mu}x_i, 0, \lambda x_i + \mu x_j).\end{aligned}$$

Hence, (x,y) can be written as a strict convex combination of two distinct points in $S(0,h)$. □

Combining Theorem 6 and Proposition 7, we have the following result.

Proposition 8. *If (x,y) is an extreme point of* conv$(S(0,h))$*, then the non-zero entries of (x,y) are one of the following:*

- $i \in N \setminus L \Rightarrow \quad x_i = \max\{d, h_i\}, \;\; y_i = \max\{d, h_i\},$
- $i \in L \Rightarrow \begin{cases} x_i = l_i, \;\; y_i = l_i \\ x_i = l_i, \;\; y_i = \max\{d, h_i\}. \end{cases}$

Proof. Let (x, y) be an extreme point of $\operatorname{conv}(S(0, h))$. From Proposition 7, (x, y) has exactly one pair of non-zero entries, say (x_i, y_i). From Theorem 6, (x_i, y_i) has to satisfy either $y_i \geq \max\{d, h_i\}$ if $i \in N \setminus L$, or both $x_i \geq l_i$ and $y_i \geq \max\{d, h_i\}$ if $i \in L$. From these inequalities together with $y_i \leq x_i$, at least two have to be satisfied at equality since $x_i > 0$, $y_i > 0$, and $y_j = x_j = 0$ for all $j \in N$, $j \neq i$. The possible solutions are exactly the combinations indicated above. $\square$

From Proposition 8, optimization over $S(0, h)$ can be done by enumeration in $\mathcal{O}(n)$ time.

2.3.3 Extended formulation for conv$(S(0, h))$

Now, let us consider the separation problem associated to (14). Given (x^*, y^*), let

$$T^* = \left\{ i \in L : \frac{x_i^*}{l_i} \leq \frac{y_i^*}{\max\{d, h_i\}} \right\}.$$

If (14) is satisfied for T^*, then it is satisfied for any $T \subseteq L$, and if in addition (16) and (15) hold, then (x^*, y^*) belongs to $\operatorname{conv}(S(0, h))$. Otherwise, T^* gives the most violated inequality from (14), and therefore it can be used to separate (x^*, y^*) from $\operatorname{conv}(S(0, h))$. Clearly, computing T^* and its corresponding inequality can be done in $\mathcal{O}(n)$ time.

Further note that (x, y) satisfies (14) for all $T \subseteq L$ if and only if

$$\sum_{i \in N \setminus L} \frac{y_i}{\max\{d, h_i\}} + \sum_{i \in L} \min\left(\frac{x_i}{l_i}, \frac{y_i}{\max\{d, h_i\}} \right) \geq 1.$$

If fact, this is the separation routine for (14) given a point (x, y). Now, the above condition holds if and only if there exists $\pi \in \mathbb{R}^{|L|}$ such that

$$\frac{x_i}{l_i} \geq \pi_i \; \forall i \in L$$

$$\frac{y_i}{\max\{d, h_i\}} \geq \pi_i \; \forall i \in L$$
$$\sum_{i \in N \setminus L} \frac{y_i}{\max\{d, h_i\}} + \sum_{i \in L} \pi_i \geq 1.$$

Thus, introducing additional variables π, we obtain an extended formulation for $\text{conv}(S(0, h))$ in a space of higher dimension given by

$$W = \left\{ (x, y, \pi) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{|L|} : \begin{array}{cr} \sum_{i \in N \setminus L} \frac{y_i}{\max\{d, h_i\}} + \sum_{i \in L} \pi_i \geq 1 & \\ \frac{x_i}{l_i} \geq \pi_i & \forall i \in L \\ \frac{y_i}{\max\{d, h_i\}} \geq \pi_i & \forall i \in L \\ y_i \geq 0 & \forall i \in N \\ x_i - y_i \geq 0 & \forall i \in N \end{array} \right\}.$$

Let $proj_{x,y}(W)$ denote the projection of W onto the (x, y)-space.

Corollary 9. $\text{conv}(S(0, h)) = proj_{x,y}(W)$.

This extended formulation is compact in the sense that we have, at most, doubled the number of variables and constraints.

2.4 *The case* $h = 0$

In this section we assume that $h = 0$ and then $S(t, 0) \subseteq \mathbb{R}^n \times \mathbb{R}^n$ takes the form

$$\sum_{i \in N} y_i \geq d \tag{20}$$
$$y_i \leq t_i + x_i \quad \forall i \in N \tag{21}$$
$$x_i \in \{0\} \cup [l_i, \infty) \quad \forall i \in N \tag{22}$$
$$y_i \geq 0 \quad \forall i \in N. \tag{23}$$

2.4.1 Inequality description of conv$(S(t, 0))$

Proposition 10. *$\sum_{i \in N} y_i \geq d$ is facet-defining for* $\text{conv}(S(t, 0))$.

Proof. Choose a point $\bar{x} \in \mathbb{R}^n$ satisfying $\bar{x}_i > \max\{d, l_i\}$ for all $i \in N$ and set $\bar{y}_i = \frac{d}{n}$ for all $i \in N$. We have that $(\bar{x}, \bar{y})$ belongs to $S(t, 0)$ and satisfies $\sum_{i \in N} \bar{y}_i = d$. Now for each $j \in N$, $j < n$, consider the points $\left(x^j, y^j\right)$ and $\left(x^{n+j}, y^{n+j}\right)$ given by

$$\left(x_i^j, y_i^j\right) = \begin{cases} (\bar{x}_j + \epsilon, \bar{y}_j) & i = j \\ (\bar{x}_i, \bar{y}_i) & i \neq j, \end{cases}$$

$$\left(x_i^{n+j}, y_i^{n+j}\right) = \begin{cases} (\bar{x}_j, \bar{y}_j - \epsilon) & i = j \\ (\bar{x}_n, \bar{y}_n + \epsilon) & i = n \\ (\bar{x}_i, \bar{y}_i) & i \neq j,\ i \neq n. \end{cases}$$

Finally, let $\left(x^{2n}, y^{2n}\right) = (\bar{x}, \bar{y})$ and let (x^n, y^n) be given by

$$(x_i^n, y_i^n) = \begin{cases} (\bar{x}_n + \epsilon, \bar{y}_n) & i = n \\ (\bar{x}_i, \bar{y}_i) & i \neq n. \end{cases}$$

For $\epsilon > 0$ sufficiently small, the collection $\left\{\left(x^j, y^j\right),\ \left(x^{n+j}, y^{n+j}\right) : j \in N\right\}$ is contained in $S(t, 0)$. Moreover, it is an affinely independent set, and since these $2n$ points satisfy $\sum_{i \in N} y_i \geq d$ at equality, this constraint defines a facet of $\text{conv}(S(t, 0))$.

□

Definition 11. *A subset $R \subseteq N$ is a reverse cover if $d_R := d - \sum_{i \in R} t_i > 0$.*

Let $\mathcal{R} \subseteq 2^N$ be the set of all reverse covers. For a reverse cover $R \in \mathcal{R}$, consider the inequality

$$\sum_{i \in R} \frac{x_i}{\max\{l_i, d_R\}} + \sum_{i \in N \setminus R} \frac{y_i}{d_R} \geq 1. \tag{24}$$

Also, let $L_R := \{i \in R :\ l_i > d_R\}$. Note that if $R = \varnothing$, we recover (20).

Proposition 12. *For each reverse cover $R \in \mathcal{R}$, (24) is valid for* $\text{conv}(S(t, 0))$.

Proof. Let $(x, y) \in S(t, 0)$. If there exists $i \in L_R$ with $x_i > 0$, then (24) is satisfied. Otherwise, $x_i = 0$ for all $i \in L_R$. Then

$$d \leq \sum_{i \in N} y_i = \sum_{i \in L_R} y_i + \sum_{i \in R \setminus L_R} y_i + \sum_{i \in N \setminus R} y_i \leq \sum_{i \in L_R} t_i + \sum_{i \in R \setminus L_R} (t_i + x_i) + \sum_{i \in N \setminus R} y_i$$

$$\implies d_R = d - \sum_{i \in R} t_i \le \sum_{i \in R \setminus L_R} x_i + \sum_{i \in N \setminus R} y_i.$$

Since $\max\{l_i, d_R\} = d_R > 0$ for each $i \in R \setminus L_R$, (24) is satisfied. $\square$

Definition 13. *A reverse cover $R \in \mathcal{R}$ is proper if*

1. $L_R \neq \varnothing$.
2. $t_i > 0$ *for all* $i \in R \setminus L_R$.

Proposition 14. *For each reverse cover $R \in \mathcal{R}$, (24) is facet-defining if and only if R is empty or if R is proper.*

Proof. The case $R = \varnothing$ follows from Proposition 10. Thus, let R be a proper reverse cover and let $i \in L_R$. For each $j \in N$, consider the points $\left(x^j, y^j\right)$ and $\left(x^{n+j}, y^{n+j}\right)$ defined as follows.

If $j \in R$,

$$\left(x_k^j, y_k^j\right) = \begin{cases} (\max\{l_j, d_R\}, t_j + d_R) & k = j \\ (0, t_k) & k \in R,\ k \neq j \\ (0,0) & k \in N \setminus R. \end{cases}$$

Then

$$\sum_{k \in N} y_k^j = \sum_{k \in R} t_k + d_R = d$$

and

$$\sum_{k \in R} \frac{x_k^j}{\max\{l_k, d_R\}} + \sum_{k \in N \setminus R} \frac{y_k^j}{d_R} = \frac{\max\{l_j, d_R\}}{\max\{l_j, d_R\}} = 1.$$

If $j \in L_R$,

$$\left(x_k^{n+j}, y_k^{n+j}\right) = \begin{cases} (l_j, t_j + d_R + \epsilon) & k = j \\ (0, t_k) & k \in R,\ k \neq j \\ (0,0) & k \in N \setminus R. \end{cases}$$

Then

$$\sum_{k \in N} y_k^{n+j} = \sum_{k \in R} t_k + d_R + \epsilon \ge d$$

and

$$\sum_{k\in R} \frac{x_k^{n+j}}{\max\{l_k, d_R\}} + \sum_{k\in N\setminus R} \frac{y_k^{n+j}}{d_R} = \frac{l_j}{\max\{l_j, d_R\}} = 1.$$

If $j \in R \setminus L_R$,

$$\left(x_k^{n+j}, y_k^{n+j}\right) = \begin{cases} (l_i, t_i + d_R + \epsilon) & k = i \\ (0, t_j - \epsilon) & k = j \\ (0, t_k) & k \in R,\ k \neq i,\ k \neq j \\ (0, 0) & k \in N \setminus R. \end{cases}$$

Then

$$\sum_{k\in N} y_k^{n+j} = \sum_{k\in R} t_k - \epsilon + d_R + \epsilon = d$$

and

$$\sum_{k\in R} \frac{x_k^{n+j}}{\max\{l_k, d_R\}} + \sum_{k\in N\setminus R} \frac{y_k^{n+j}}{d_R} = \frac{l_i}{\max\{l_i, d_R\}} = 1.$$

If $j \in N \setminus R$,

$$\left(x_k^{j}, y_k^{j}\right) = \begin{cases} (\max\{l_j, d_R\}, d_R) & k = j \\ (0, t_k) & k \in R \\ (0, 0) & k \in N \setminus R,\ k \neq j, \end{cases}$$

$$\left(x_k^{n+j}, y_k^{n+j}\right) = \begin{cases} (\max\{l_j, d_R\} + \epsilon, d_R) & k = j \\ (0, t_k) & k \in R \\ (0, 0) & k \in N \setminus R,\ k \neq j. \end{cases}$$

Then

$$\sum_{k\in N} y_k^{j} = \sum_{k\in N} y_k^{n+j} = \sum_{k\in R} t_k + d_R = d$$

and

$$\sum_{k\in R} \frac{x_k^{j}}{\max\{l_k, d_R\}} + \sum_{k\in N\setminus R} \frac{y_k^{j}}{d_R} = \sum_{k\in R} \frac{x_k^{n+j}}{\max\{l_k, d_R\}} + \sum_{k\in N\setminus R} \frac{y_k^{n+j}}{d_R} = \frac{d_R}{d_R} = 1.$$

Given that $d_R < l_j$ for all $j \in L_R$ and $0 < t_j$ for all $j \in R \setminus L_R$, for $\epsilon > 0$ sufficiently small, we have that $\left\{ \left(x^j, y^j\right),\ \left(x^{n+j}, y^{n+j}\right) : j \in N \right\}$ is contained in $S(t, 0)$.

Moreover, it is an affinely independent set, and since these $2n$ points satisfy (24) at equality, this constraint defines a facet of $\mathrm{conv}(S(t,0))$.

For the converse, let R be a nonempty cover that is not proper, thus either $L_R = \varnothing$ or there exists $i \in R \setminus L_R$ having $t_i = 0$. In the former case, $\max\{l_i, d_R\} = d_R$ for all $i \in R$, and then (24) is generated as the sum of (20) and (21) for $i \in R$. In the latter, since $t_i = 0$, we have $d_{R\setminus\{i\}} = d_R$ and $y_i \leq x_i$. Since $i \in R \setminus L_R$, we also have $\max\{l_i, d_R\} = d_R$. Thus

$$\begin{aligned}
\sum_{j\in R} \frac{x_j}{\max\{l_j, d_R\}} + \sum_{j\in N\setminus R} \frac{y_j}{d_R} &= \sum_{j\in R\setminus\{i\}} \frac{x_j}{\max\{l_j, d_R\}} + \frac{x_i}{\max\{l_i, d_R\}} + \sum_{j\in N\setminus R} \frac{y_j}{d_R} \\
&\geq \sum_{j\in R\setminus\{i\}} \frac{x_j}{\max\{l_j, d_R\}} + \frac{y_i}{d_R} + \sum_{j\in N\setminus R} \frac{y_j}{d_R} \\
&= \sum_{j\in R\setminus\{i\}} \frac{x_j}{\max\{l_j, d_R\}} + \sum_{j\in N\setminus(R\setminus\{i\})} \frac{y_j}{d_R}.
\end{aligned}$$

Hence, the inequality given by R is implied by the one given by $R \setminus \{i\}$, and therefore it cannot be facet-defining. $\square$

We now present the main result of this section.

Theorem 15. $\mathrm{conv}(S(t,0))$ *is given by the following inequalities*

$$\sum_{i\in R} \frac{x_i}{\max\{l_i, d_R\}} + \sum_{i\in N\setminus R} \frac{y_i}{d_R} \geq 1 \quad \forall R \in \mathcal{R}$$

$$y_i \leq x_i + t_i \quad \forall i \in N \tag{25}$$

$$x_i \geq 0 \quad \forall i \in N \tag{26}$$

$$y_i \geq 0 \quad \forall i \in N. \tag{27}$$

Proof. Let $(c, \alpha) \in \mathbb{R}^n \times \mathbb{R}^n$ be a non-zero vector and consider the problem

$$\min \{cx + \alpha y : \ (x,y) \in S(t,0)\}.$$

As in the proof of Theorem 6, we will show that if this problem has finite optimal value, then there exists one inequality from (24)-(27) that is satisfied at equality by all optimal solutions.

Assumption 1: $c \geq 0$ and $c + \alpha \geq 0$.

If for some $i \in N$ we have $c_i < 0$ or $c_i + \alpha_i < 0$, then the problem is unbounded. Thus, we may assume $c \geq 0$ and $c + \alpha \geq 0$. $\diamond$

In particular, Assumption 1 implies that the objective value is bounded and there exists an optimal solution. Let (x^*, y^*) be any such solution.

Assumption 2: $\alpha \geq 0$.

If for some $i \in N$ we have $\alpha_i < 0$, then $y_i^* = t_i + x_i^*$ by optimality, that is, (25) is satisfied at equality. Thus, we may assume $\alpha \geq 0$. $\diamond$

From Assumptions 1 and 2, we have that the optimal value is nonnegative.

Assumption 3: $cx^* + \alpha y^* > 0$.

Suppose that the optimal value is zero. Since $(c, \alpha) \neq (0, 0)$, by Assumptions 1 and 2, there must exist $i \in N$ such that either $\alpha_i > 0$ or $c_i > 0$. By optimality, in the former case we must have $y_i^* = 0$, while in the latter $x_i^* = 0$ must hold. Therefore, either (27) or (26) must be satisfied at equality. Thus, we may assume $cx^* + \alpha y^* > 0$. $\diamond$

Claim 1: $c + \alpha > 0$.

If $c_i = \alpha_i = 0$ for some $i \in N$, then the optimal value is zero, contradicting Assumption 3. $\diamond$

Let $R := \{i \in N : \ \alpha_i = 0\}$. From Assumption 3 and the definition of R, we have $\sum_{i \in R} t_i < d$, since otherwise the optimal value is zero. Hence, R is a reverse cover. We also have $c_i > 0$ for all $i \in R$ by Claim 1, and $\alpha_i > 0$ for all $i \in N \setminus R$.

We claim that

$$\sum_{i \in R} \frac{x_i^*}{\max\{l_i, d_R\}} + \sum_{i \in N \setminus R} \frac{y_i^*}{d_R} = 1.$$

Suppose not. Let $L_R^+ := \{i \in L_R : x_i^* > 0\}$, $(R \setminus L_R)^+ := \{i \in R \setminus L_R : x_i^* > 0\}$, and $(N \setminus R)^+ := \{i \in N \setminus R : y_i^* > 0\}$. Then

$$\sum_{i \in L_R^+} \frac{x_i^*}{l_i} + \sum_{i \in (R \setminus L_R)^+} \frac{x_i^*}{d_R} + \sum_{i \in (N \setminus R)^+} \frac{y_i^*}{d_R} > 1. \tag{28}$$

Claim 2: $L_R^+ = \varnothing$.

Suppose $i \in L_R^+$, that is, $i \in R$ and $x_i^* \geq l_i > d_R$. Note that since $\alpha_j = 0$ for all $j \in R$, we can set $y_j^* = t_j$ for each $j \in R, j \neq i$, and $y_i^* = t_i + d_R$ without affecting the feasibility and objective value of the solution. Recalling that $c_i > 0$ for all $i \in R$ and $\alpha_i > 0$ for all $\in N \setminus R$, from (28) and optimality we have $(R \setminus L_R)^+ = (N \setminus R)^+ = \varnothing$ and $L_R^+ = \{i\}$. Then (28) implies $x_i^* > l_i > d_R$, contradicting optimality since setting $x_i^* = l_i$ improves the objective value. $\diamond$

Now, we have

$$\sum_{i \in (R \setminus L_R)^+} x_i^* + \sum_{i \in (N \setminus R)^+} y_i^* > d_R. \tag{29}$$

Claim 3: $(N \setminus R)^+ = \varnothing$.

From (29) and Claim 2, we have

$$d < \sum_{i \in (R \setminus L_R)^+} x_i^* + \sum_{i \in R} t_i + \sum_{i \in (N \setminus R)^+} y_i^* = \sum_{i \in R} (x_i^* + t_i) + \sum_{i \in (N \setminus R)^+} y_i^*.$$

If $(N \setminus R)^+$ is nonempty, we can set $y_i^* = t_i + x_i^*$ for each $i \in R$ without changing the objective value, and then decrease y_i^* for some $i \in (N \setminus R)^+$, contradicting optimality as $\alpha_i > 0$ for all $i \in (N \setminus R)^+$. $\diamond$

We arrive at

$$\sum_{i \in (R \setminus L_R)^+} x_i^* > d_R.$$

Then we can improve upon (x^*, y^*) by taking $i \in \arg\min\{c_j : j \in (R \setminus L_R)^+\}$ and defining $(\bar{x}, \bar{y})$ by

$$(\bar{x}_j, \bar{y}_j) = \begin{cases} (d_R, t_j + d_R) & j = i \\ (0, t_j) & j \in R,\ j \neq i \\ (0, 0) & j \in N \setminus R. \end{cases}$$

$\square$

2.4.2 Extended formulation for conv$(S(t,0))$

At first sight, it is not clear how to separate the inequalities given by (24). We will show that this can be done using an extended formulation. We first state a result similar to Proposition 7.

Proposition 16. *If (x,y) is an extreme point of* $\operatorname{conv}(S(t,0))$*, then x has at most one non-zero entry.*

Proof. We claim that if $x_i > 0$, then $y_i > t_i$. By contradiction, suppose $x_i > 0$ and $y_i \leq t_i$. We can write

$$(x_i, y_i) = \frac{1}{2}\left[(2x_i, y_i) + (0, y_i)\right].$$

Thus, (x,y) can be written as the average of two distinct points in $S(t,0)$.

Now, suppose that x has more than one non-zero entry, say $x_i > 0$ and $x_j > 0$. By the claim, $y_i > t_i$ and $y_j > t_j$. Thus, there exist $\lambda, \mu \in (0,1]$ such that $y_i = t_i + \lambda x_i$ and $y_j = t_j + \mu x_j$. Then we can write

$$\begin{aligned}(x_i, x_j, y_i, y_j) &= (x_i, x_j, t_i + \lambda x_i, t_j + \mu x_j)\\ &= \frac{\lambda x_i}{\lambda x_i + \mu x_j}(x_i + \frac{\mu}{\lambda}x_j, 0, t_i + \lambda x_i + \mu x_j, t_j)\\ &\quad + \frac{\mu x_j}{\lambda x_i + \mu x_j}(0, x_j + \frac{\lambda}{\mu}x_i, t_i, t_j + \lambda x_i + \mu x_j).\end{aligned}$$

Also, notice that

$$t_i + \lambda x_i + \mu x_j = t_i + \lambda\left(x_i + \frac{\mu}{\lambda}x_j\right) \leq t_i + \left(x_i + \frac{\mu}{\lambda}x_j\right),$$

$$t_j + \lambda x_i + \mu x_j = t_j + \mu\left(\frac{\lambda}{\mu}x_i + x_j\right) \leq t_j + \left(\frac{\lambda}{\mu}x_i + x_j\right).$$

Hence, (x,y) can be written as a strict convex combination of two distinct points in $S(t,0)$. □

Now consider the polyhedra

$$S_0 := \{(x,y) \in S(t,0) : \ x_j = 0 \ \forall j \in N\}$$

$$= \left\{ (x,y) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ : \begin{array}{cc} \sum_{j\in N} y_j \geq d & \\ -y_j \geq -t_j & \forall j \in N \\ -x_j \geq 0 & \forall j \in N \end{array} \right\},$$

$S_i := \{(x,y) \in S(t,0) : \ x_i \geq l_i, \ x_j = 0 \ \forall j \neq i\}$

$$= \left\{ (x,y) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ : \begin{array}{cc} \sum_{j\in N} y_j \geq d & \\ x_i - y_i \geq -t_i & \\ -y_j \geq -t_j & \forall j \neq i \\ x_i \geq l_i & \\ -x_j \geq 0 & \forall j \neq i \end{array} \right\}, \ i \in N.$$

Note that S_i is nonempty for each $i \in N$, while S_0 is nonempty if and only if $\sum_{j\in N} t_j \geq d$. Set $\bar{N} = \{0\} \cup N$.

For a set C, let $\overline{conv}(C)$ denote the closure of its convex hull. If C is convex, let $ext(C)$ and $rec(C)$ denote the set of extreme points and the recession cone of C, respectively.

Proposition 17. $\text{conv}(S(t,0)) = \overline{conv}(\cup_{i\in\bar{N}} S_i)$.

Proof. The reverse inclusion is easy as $S_i \subseteq S(t,0)$ for all $i \in \bar{N}$ and $\text{conv}(S(t,0))$ is closed by Proposition 3.

For the forward inclusion, let $(x,y) \in ext(\text{conv}(S(t,0)))$. From Proposition 16, (x,y) belongs to some S_i, $i \in \bar{N}$, thus $ext(\text{conv}(S(t,0))) \subseteq \text{conv}(\cup_{i\in\bar{N}} S_i)$. It remains to show that $rec(\text{conv}(S(t,0))) \subseteq rec(\overline{conv}(\cup_{i\in\bar{N}} S_i))$. For this, let $(x,y) \in rec(\text{conv}(S(t,0)))$. From Theorem 15, we can conclude that $x \geq 0$, $y \geq 0$, and $x \geq y$. Write $(x,y) = \sum_{i\in N}(x_i e^i, y_i e^i)$, where e^i is the i-th canonical vector in $\mathbb{R}^n$. On the other hand, by disjunctive programming [5], we have $rec(\overline{conv}(\cup_{i\in\bar{N}} S_i)) = \text{conv}(\cup_{i\in\bar{N}} rec(S_i))$. Since $rec(S_i)$ is a convex cone for each $i \in \bar{N}$, we also have $\text{conv}(\cup_{i\in\bar{N}} rec(S_i)) = \sum_{i\in\bar{N}} rec(S_i)$. Given that $(x_i e^i, y_i e^i) \in rec(S_i)$ for each $i \in N$, we have that $(x,y) \in rec(\overline{conv}(\cup_{i\in\bar{N}} S_i))$, which completes the proof. □

From Proposition 17, $\text{conv}(S(t,0))$ admits a compact representation as the projection onto (x,y) of a higher dimensional polyhedron which can be used to find violated inequalities. Specifically, given $(\bar{x},\bar{y}) \in \mathbb{R}^n \times \mathbb{R}^n$, let $P \subseteq \mathbb{R}_+^{(n+1)n} \times \mathbb{R}_+^{(n+1)n} \times \mathbb{R}_+^{n+1}$ be the set of vectors (x,y,λ) satisfying

$$\begin{aligned}
\sum_{j\in N} y_j^0 - d\lambda^0 &\geq 0 & & (\alpha_0)\\
-y_j^0 + t_j\lambda^0 &\geq 0 & \forall j \in N & \quad (\beta_{0j})\\
-x_j &\geq 0 & \forall j \in N & \\
\sum_{j\in N} y_j^i - d\lambda^i &\geq 0 & \forall i \in N & \quad (\alpha_i)\\
x_i^i - y_i^i + t_i\lambda^i &\geq 0 & \forall i \in N & \quad (\beta_{ii})\\
-y_j^i + t_j\lambda^i &\geq 0 & \forall i \in N,\ \forall j \neq i & \quad (\beta_{ij})\\
x_i^i - l_i\lambda^i &\geq 0 & \forall i \in N & \quad (\gamma_i)\\
-x_j^i &\geq 0 & \forall i \in N,\ \forall j \neq i & \\
x_i^i &= \bar{x}_i & \forall i \in N & \quad (\nu_i)\\
\sum_{j\in \bar{N}} y_i^j &= \bar{y}_i & \forall i \in N & \quad (\eta_i)\\
\sum_{j\in \bar{N}} \lambda^j &= 1 & & \quad (\pi).
\end{aligned}$$

Thus, $(\bar{x},\bar{y})$ belongs to $\overline{conv}(\cup_{i\in\bar{N}} S_i)$, and therefore to $\text{conv}(S(t,0))$, if and only if P is nonempty. Let $Q \subseteq \mathbb{R}_+^{n+1} \times \mathbb{R}_+^{(n+1)n} \times \mathbb{R}_+^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$ be the set of vectors $(\alpha, \beta, \gamma, \eta, \nu, \pi)$ such that

$$\begin{aligned}
\alpha_0 - \beta_{0j} + \nu_j &\leq 0 && \forall j \in N \\
-d\alpha_0 + \sum_{j \in N} t_j \beta_{0j} + \pi &\leq 0 \\
\alpha_i - \beta_{ij} + \nu_j &\leq 0 && \forall i \in N, \forall j \in N \\
\beta_{ii} + \gamma_i + \eta_i &\leq 0 && \forall i \in N \\
-d\alpha_i + \sum_{j \in N} t_j \beta_{ij} - l_i \gamma_i + \pi &\leq 0 && \forall i \in N \\
\pi + \sum_{i \in N} \eta_i \bar{x}_i + \sum_{i \in N} \nu_i \bar{y}_i &> 0.
\end{aligned}$$

After removing unnecessary variables and constraints, by Farkas' Lemma, P is nonempty if and only if Q is empty. Moreover, given $(\bar{x}, \bar{y})$ in the continuous relaxation of (20)-(23), there is a violated inequality from (24) if and only if the problem

$$\begin{aligned}
\min \quad & \sum_{i \in N} \eta_i \bar{x}_i + \sum_{i \in N} \nu_i \bar{y}_i - \pi && && (30) \\
\text{s.t.} \quad & \alpha_0 - \beta_{0j} - \nu_j \leq 0 && \forall j \in N \\
& -d\alpha_0 + \sum_{j \in N} t_j \beta_{0j} + \pi \leq 0 \\
& \alpha_i - \beta_{ij} - \nu_j \leq 0 && \forall i \in N, \forall j \in N \\
& \beta_{ii} + \gamma_i - \eta_i \leq 0 && \forall i \in N \\
& -d\alpha_i + \sum_{j \in N} t_j \beta_{ij} - l_i \gamma_i + \pi \leq 0 && \forall i \in N \\
& \sum_{i \in N} \eta_i + \sum_{i \in N} \nu_i + \pi = 1 \\
& \alpha, \beta, \gamma, \eta, \nu, \pi \geq 0
\end{aligned}$$

has negative optimal value. In this case, any optimal solution to (30) yields a valid inequality for $\text{conv}(S(t, 0))$ that is not satisfied by $(\bar{x}, \bar{y})$.

2.5 *A semi-continuous transportation problem*

2.5.1 The problem and its complexity

Consider now the case where we intersect $m \geq 1$ sets of the form $S(t, h)$. Specifically, let $M := \{1, \ldots, m\}$ be a set of nodes that receive flow from nodes in N, where each $j \in M$ has a demand $d_j > 0$ to be met. In this context, we refer to N and M as suppliers and customers, respectively. In this setting, $l \in \mathbb{R}^n_+$ is a vector of lower bounds for supplier capacities, $h \in \mathbb{R}^{nm}_+$ is a vector of lower bounds for arc flows, and $t \in \mathbb{R}^n_+$ is a vector of initial supplier capacities.

Let $S_* \subseteq \mathbb{R}^n \times \mathbb{R}^{nm}$ be the set of vectors (x, y) such that

$$\sum_{i \in N} y_{ij} \geq d_j \qquad \forall j \in M \tag{31}$$

$$\sum_{j \in M} y_{ij} \leq t_i + x_i \qquad \forall i \in N \tag{32}$$

$$x_i \in \{0\} \cup [l_i, \infty) \qquad \forall i \in N \tag{33}$$

$$y_{ij} \in \{0\} \cup [h_{ij}, \infty) \qquad \forall i \in N,\ \forall j \in M. \tag{34}$$

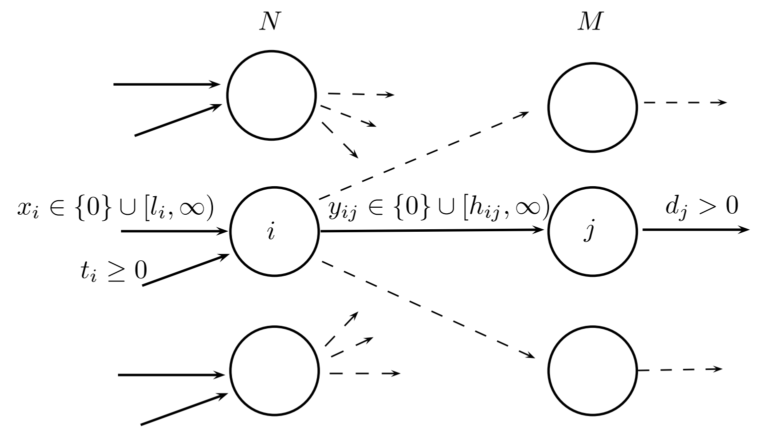
Constraints (31), (33), and (34) are analogous to (1), (3), and (4) of $S(t, h)$, respectively. In addition, constraints (32) ensure that the total outflow from any supplier does not exceed its available capacity. As with the inflow set, a graphical interpretation is given in Figure 6.

Now we address the complexity of optimization over S_*.

Proposition 18. *Optimizing a linear function over S_* is $\mathcal{NP}$-hard, even if $t = 0$ and $h = 0$.*

Proof. We show that the Uncapacitated Facility Location Problem (UFLP), which is $\mathcal{NP}$-hard, can be reduced to optimization of a linear function over S_*. An instance of UFLP is defined by a set of potential facilities N, a set of customers M, and cost functions $f : N \to \mathbb{R}_+$ and $e : N \times M \to \mathbb{R}_+$. The objective is to compute

$$\min_{N' \subseteq N} \left\{ \sum_{i \in N'} f_i + \sum_{j \in M} \min_{i \in N'} e_{ij} \right\}.$$



N
M
$x_i \in \{0\} \cup [l_i, \infty)$
$t_i \geq 0$
i
$y_{ij} \in \{0\} \cup [h_{ij}, \infty)$
j
$d_j > 0$

Figure 6: Semi-continuous transportation problem.

We can formulate UFLP as an integer programming problem. Let $z_i = 1$ if and only if facility i is open, and $w_{ij} = 1$ if and only if customer j is assigned to facility i. The corresponding formulation is

$$\begin{aligned} z_1 = \min \quad & \sum_{i \in N} f_i z_i + \sum_{j \in M} \sum_{i \in N} e_{ij} w_{ij} && \\ \text{s.t.} \quad & w_{ij} \leq z_i && \forall i \in N,\ \forall j \in M \\ & \sum_{i \in N} w_{ij} = 1 && \forall j \in M \\ & w_{ij} \in \{0,1\} && \forall i \in N,\ \forall j \in M \\ & z_i \in \{0,1\} && \forall i \in N. \end{aligned}$$

Given an instance π_1 of UFLP, we want to construct an instance π_2 of linear optimization over S_* with the same objective value. We identify N with the set of supply nodes and M with the set of customers. Let $l_i = m+1$ for all $i \in N$, $d_j = 1$ for all $j \in M$, $c_i = \frac{f_i}{m+1}$ for all $i \in N$, and $\alpha_{ij} = e_{ij}$ for all $i \in N$ and $j \in M$. We also set $t_i = 0$ for each $i \in N$, and $h_{ij} = 0$ for each $i \in N$ and $j \in M$. The corresponding instance π_2 is then

$$\begin{aligned} z_2 = \min \quad & \sum_{i \in N} \frac{f_i}{m+1} x_i + \sum_{i \in N} \sum_{j \in M} e_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j \in M} y_{ij} \leq x_i && \forall i \in N \\ & \sum_{i \in N} y_{ij} \geq 1 && \forall j \in M \\ & y_{ij} \geq 0 && \forall i \in N,\ \forall j \in M \\ & x_i \in \{0\} \cup [m+1, \infty) && \forall i \in N. \end{aligned}$$

Let (z^*, w^*) be an optimal solution to π_1. If we set $x_i = l_i$ if $z_i^* = 1$ and 0 otherwise, and $y_{ij} = d_j$ if $w_{ij}^* = 1$ and 0 otherwise, then we get a feasible solution (x, y) to π_2 with cost z_1. Hence, $z_2 \leq z_1$.

Now, let (x^*, y^*) be an optimal solution to π_2. Since $c \geq 0$, $\alpha \geq 0$, and $l_i \geq m+1$, we may assume that $x_i^* \in \{0, l_i\}$ for all $i \in N$. In addition, by integrality property of networks, we may also assume that $y_{ij}^* \in \{0, d_j\}$ for any $i \in N$ and $j \in M$. Setting $z_i = 1$ if $x_i^* = l_i$ and 0 otherwise, and $w_{ij} = 1$ if $y_{ij}^* = d_j$ and 0 otherwise, we get a feasible solution (z, w) to π_1 with cost z_2. Hence, $z_1 \leq z_2$. $\square$

2.5.2 Analysis of a relaxation of S_*

A special case of S_* arises when $h = 0$, which constitutes a relaxation for this class of problems. In such a case, we shall present structural characteristics of the convex hull of this set that will give us some insight into the complexity of optimization over it. In fact, we will show some results for a slightly more general set.

For lower bounds $l \in \mathbb{R}^n_+$, demands $d \in \mathbb{R}^m_+$, not necessarily positive, and initial capacities $t \in \mathbb{R}^n_+$, we define

$$S_*(l,d,t) := \left\{ (x,y) \in \mathbb{R}^n \times \mathbb{R}^{nm} : \begin{array}{rl} \sum_{i \in N} y_{ij} \geq d_j & \forall j \in M \\ \sum_{j \in M} y_{ij} \leq t_i + x_i & \forall i \in N \\ y_{ij} \geq 0 & \forall i \in N,\ j \in M \\ x_i \in \{0\} \cup [l_i, \infty) & \forall i \in N \end{array} \right\}.$$

Once more, we begin with a result in the spirit of Propositions 7 and 16.

Proposition 19. *If (x,y) is an extreme point of $\operatorname{conv}(S_*(l,d,t))$, then $\sum_{j \in M} y_{ij} > t_i$ for all $i \in N$ such that $x_i > 0$.*

Proof. Suppose that $x_i > 0$ and $\sum_{j \in M} y_{ij} \leq t_i$ for some $i \in N$. Then we can write

$$(x_i, y_{i1}, \ldots, y_{im}) = \frac{1}{2}[(2x_i, y_{i1}, \ldots, y_{im}) + (0, y_{i1}, \ldots, y_{im})],$$

that is, (x,y) is the strict convex combination of two distinct points in $S_*(l,d,t)$, and thus it cannot be an extreme point of $\operatorname{conv}(S_*(l,d,t))$. $\square$

For $(\bar{x}, \bar{y}) \in S_*(l,d,t)$, we define the support $\sigma(\bar{x})$ of $\bar{x}$ as the subset of suppliers with positive production, that is

$$\sigma(\bar{x}) := \{i \in N : \bar{x}_i > 0\}.$$

We will prove that if $(\bar{x}, \bar{y})$ is an extreme point of $\operatorname{conv}(S_*(l,d,t))$, then $|\sigma(\bar{x})| \leq m$. We need the following key lemma.

Lemma 20. *If $t = 0$ and $(\bar{x}, \bar{y})$ is an extreme point of $\operatorname{conv}(S_*(l,d,0))$, then $|\sigma(\bar{x})| \leq m$.*

Proof. For a contradiction, suppose that for some positive integers $n > m$ the claim does not hold. Choose n and m so that $n + m$ is minimum among all such instances. Note that by Proposition 16, $m > 1$. Let $(\bar{x}, \bar{y})$ be an extreme point of $S_*(l,d,0)$ having $|\sigma(\bar{x})| > m$, where $l \in \mathbb{R}^n_+$ and $d \in \mathbb{R}^m_+$.

By minimality of $n+m$, we may assume that $|\sigma(\bar{x})| = n$, since otherwise $x_i = 0$ for some $i \in N$, and removing this supplier from the instance would yield a smaller counterexample.

Claim 1: $n = m+1$.

If $n > m+1$, let $\widehat{N} := N \setminus \{n\}$. We define $\widehat{d} \in \mathbb{R}^m_+$ by

$$\widehat{d}_j := \sum_{i \in \widehat{N}} \bar{y}_{ij} \ \forall j \in M.$$

Let $(\widehat{x}, \widehat{y}) \in \mathbb{R}^{n-1}_+ \times \mathbb{R}^{(n-1)m}_+$ and $\widehat{l} \in \mathbb{R}^{n-1}_+$ be the restrictions of $(\bar{x}, \bar{y})$ and l with respect to $\widehat{N}$, respectively. We have that $(\widehat{x}, \widehat{y})$ is feasible for $S_*(\widehat{l}, \widehat{d}, 0)$ and $|\sigma(\widehat{x})| = n-1 \geq m+1$. By minimality of $n+m$, $(\widehat{x}, \widehat{y})$ cannot be an extreme point of $\text{conv}(S_*(\widehat{l}, \widehat{d}, 0))$. Thus, we can write

$$(\widehat{x}, \widehat{y}) = \sum_{p=1}^{q} \lambda_p (x^p, y^p),$$

where $q \geq 2$, $\{(x^p, y^p) : p = 1, \ldots, q\}$ are distinct points in $S_*(\widehat{l}, \widehat{d}, 0)$, $\lambda_p > 0$ for all $p = 1, \ldots, q$, and $\sum_{p=1}^{q} \lambda_p = 1$. For each $p = 1, \ldots, q$, we extend (x^p, y^p) to $(\widetilde{x}^p, \widetilde{y}^p) \in \mathbb{R}^n \times \mathbb{R}^{nm}$ by setting $\widetilde{x}^p_n = \bar{x}_n$ and $\widetilde{y}^p_{nj} = \bar{y}_{nj}$ for all $j \in M$. Since $x^p \geq \widehat{l}$ and $\bar{x}_n \geq l_n$, we have $\widetilde{x}^p \geq l$. In addition, for each $j \in M$, we have

$$\sum_{i \in N} \widetilde{y}^p_{ij} = \sum_{i \in \widehat{N}} y^p_{ij} + \bar{y}_{nj} \geq \widehat{d}_j + \bar{y}_{nj} \geq d_j.$$

Thus, $\{(\widetilde{x}^p, \widetilde{y}^p) : p = 1, \ldots, q\}$ are distinct points in $S_*(l, d, 0)$. We can see that

$$(\bar{x}, \bar{y}) = \sum_{p=1}^{q} \lambda_p (\widetilde{x}^p, \widetilde{y}^p)$$

and therefore $(\bar{x}, \bar{y})$ cannot be an extreme point of $\text{conv}(S_*(l, d, 0))$. The claim is thus proved. $\diamond$

Let $G = (N \cup M, E)$ be a bipartite graph where $i \in N$ is adjacent to $j \in M$ if and only if $\bar{y}_{ij} > 0$. Notice that since $\sigma(\bar{x}) = N$, by Proposition 19 we have that for each $i \in N$, there exists $j \in M$ having $\bar{y}_{ij} > 0$, and therefore $deg(i) \geq 1$ for all $i \in N$.

Furthermore, we may assume $deg(j) \geq 1$ for all $j \in M$, since if $deg(j) = 0$, then $d_j = 0$ and removing this customer from the instance yields a smaller counterexample. Therefore, given that $n = m + 1$, there must exist some component of G having more suppliers than customers. Hence, we may assume that G is connected, since otherwise some component of G induces a smaller counterexample. We may also assume that G is acyclic, since otherwise we can modify $\bar{y}$ along the arcs in a cycle and write $(\bar{x}, \bar{y})$ as the average of two different solutions in $S_*(l, d, 0)$. Thus, we may assume that G is a tree.

Claim 2: $deg(j) = 2 \ \forall j \in M$.

We first argue that $deg(j) \geq 2$ for all $j \in M$. By contradiction, we may assume that $deg(m) = 1$ and that m is supplied by n. As before, let $\widehat{N} := N \setminus \{n\}$ and $\widehat{M} := M \setminus \{m\}$. We define $\widehat{d} \in \mathbb{R}_+^{m-1}$ by

$$\widehat{d}_j := \sum_{i \in \widehat{N}} \bar{y}_{ij} \ \forall j \in \widehat{M}.$$

Taking the restrictions of $(\bar{x}, \bar{y})$ and l with respect to $\widehat{N}$ and $\widehat{M}$, and proceeding as in the proof of Claim 1, we conclude that $(\bar{x}, \bar{y})$ cannot be an extreme point of $\text{conv}(S_*(l, d, 0))$. Hence, $deg(j) \geq 2 \ \forall j \in M$. However, since G is a tree, we have $|E| = |N \cup M| - 1 = m + 1 + m - 1 = 2m$, and thus $deg(j) = 2$ for each $j \in M$. The claim is thus proved. $\diamond$

Now, for each $i \in N$, let

$$M(i) := \{j \in M : \ (i, j) \in E\},$$

$$N(i) := \{l \in N \setminus \{i\} : \ \exists j \in M \text{ such that } (i, j), (l, j) \in E\}.$$

In other words, $M(i)$ are the customers served by i, while $N(i)$ are the suppliers that share a customer with i, which we refer to as its neighbors. Clearly $l \in N(i)$ if and only if $i \in N(l)$. Note that since G is acyclic, any two suppliers can have at most one common customer. Thus, given neighbors i and l in N, there exists a unique $j =: j(i, l) \in M$ connecting them in G.

Let $(c,\alpha) \in \mathbb{R}^n \times \mathbb{R}^{nm}$ be such that $(\bar{x},\bar{y})$ is the unique minimizer in $S_*(l,d,0)$ with respect to this function. For each $i \in N$, consider the solution (x^i, y^i) given by

$$x^i_l = \begin{cases} 0 & l = i \\ \bar{x}_l + \bar{y}_{ij(i,l)} & l \in N(i) \\ \bar{x}_l & \text{otherwise,} \end{cases} \qquad y^i_{lj} = \begin{cases} 0 & l = i \\ \bar{y}_{lj} + \bar{y}_{ij(i,l)} & l \in N(i),\ j = j(i,l) \\ \bar{y}_{lj} & \text{otherwise.} \end{cases}$$

Thus, we obtain (x^i, y^i) from $(\bar{x},\bar{y})$ by moving the production from i to its neighbors and removing i from the solution. It is straightforward to verify that (x^i, y^i) is feasible to $S_*(l,d,0)$. However, since $(\bar{x},\bar{y})$ is the unique minimizer for (c,α), we have that the cost incurred by $(\bar{x},\bar{y})$ is less than the cost incurred by (x^i, y^i). Since these solutions only differ in the variables associated to i and its neighbors, we have

$$c_i\bar{x}_i + \sum_{j\in M(i)} \alpha_{ij}\bar{y}_{ij} < \sum_{l\in N(i)} (c_l + \alpha_{lj(i,l)})\bar{y}_{ij(i,l)}.$$

Recalling that $\sum_{j\in M(i)} \bar{y}_{ij} \leq \bar{x}_i$, we have

$$\sum_{j\in M(i)} (c_i + \alpha_{ij})\bar{y}_{ij} < \sum_{l\in N(i)} (c_l + \alpha_{lj(i,l)})\bar{y}_{ij(i,l)}.$$

Rewriting the left-hand-side in the last inequality, we obtain

$$\sum_{l\in N(i)} (c_i + \alpha_{ij(i,l)})\bar{y}_{ij(i,l)} < \sum_{l\in N(i)} (c_l + \alpha_{lj(i,l)})\bar{y}_{ij(i,l)}.$$

Hence, there must exist some $l \in N(i)$ such that

$$c_i + \alpha_{ij(i,l)} < c_l + \alpha_{lj(i,l)}.$$

For neighbors i and l, we say that i dominates l if the above inequality holds. Thus, we have that any supplier has to dominate at least one of its neighbors.

Let $G' = (N, E')$ be a graph where $(i,j) \in E'$ if and only if i and j are neighbors in G, and note that G' is also a tree. Let $L \subseteq N$ be the set of leaves of G'. Since $n = m + 1 \geq 3$, L has at least two elements and $N \setminus L$ is nonempty. Note that any leaf dominates its unique neighbor. Now, pick some $r \in L$ as a root of G', and let $i \in N \setminus L$ be such that all of its children are leaves of G'. Since i is dominated by its children, it must dominate its parent. Reasoning by induction, we have that any supplier has to dominate its parent. In particular, we conclude that r is dominated by its child, a contradiction since r is a leaf. This completes the proof. $\square$

With Proposition 19 and Lemma 20 at hand, we can prove the main result of this section.

Theorem 21. *For any $t \geq 0$, if $(\bar{x}, \bar{y})$ is an extreme point of* $\operatorname{conv}(S_*(l, d, t))$*, then $|\sigma(\bar{x})| \leq m$.*

Proof. For a contradiction, suppose that for some positive integers $n > m$ the claim does not hold. Let $(\bar{x}, \bar{y})$ be an extreme point of $S_*(l, d, t)$ having $|\sigma(\bar{x})| > m$, where $l, t \in \mathbb{R}^n_+$ and $d \in \mathbb{R}^m_+$.

For each $i \in N$, let index $j(i) \in M$ be such that $\sum_{j \in M,\, j < j(i)} \bar{y}_{ij} \leq t_i$ and $\sum_{j \in M,\, j \leq j(i)} \bar{y}_{ij} > t_i$. Since $(\bar{x}, \bar{y})$ is an extreme point of $\operatorname{conv}(S_*(l, d, t))$, by Proposition 19, $j(i)$ is well-defined for all $i \in N$. We define $\widehat{y} \in \mathbb{R}^{nm}_+$ and $\widehat{d} \in \mathbb{R}^m_+$ by

$$\widehat{y}_{ij} = \begin{cases} 0 & j < j(i) \\ \displaystyle\sum_{k \in M,\, k \leq j(i)} \bar{y}_{ik} - t_i & j = j(i) \\ \bar{y}_{ij} & j > j(i), \end{cases}$$

$$\widehat{d}_j = \sum_{i \in N} \widehat{y}_{ij} \;\forall j \in M.$$

Also, let $\widehat{x} = \bar{x}$. Then $\sum_{j \in M} \widehat{y}_{ij} = \sum_{j \in M} \bar{y}_{ij} - t_i \leq \bar{x}_i = \widehat{x}_i$ for all $i \in N$. Moreover, $(\widehat{x}, \widehat{y})$ is feasible to $S_*(l, \widehat{d}, 0)$. Since $|\sigma(\widehat{x})| = |\sigma(\bar{x})| > m$, by Lemma 20, $(\widehat{x}, \widehat{y})$ cannot be an extreme point of $\operatorname{conv}(S_*(l, \widehat{d}, 0))$. Thus, we can write

$$(\widehat{x}, \widehat{y}) = \sum_{p=1}^{q} \lambda_p (x^p, y^p),$$

where $q \geq 2$, $\{(x^p, y^p) : p = 1, \ldots, q\}$ are distinct points in $S_*(l, \widehat{d}, 0)$, $\lambda_p > 0$ for all $p = 1, \ldots, q$, and $\sum_{p=1}^{q} \lambda_p = 1$. Notice that for each $p = 1, \ldots, q$ and $i \in N$, $y_{ij}^p = 0$ for all $j < j(i)$. Then we can define $w \in \mathbb{R}^{nm}$ by

$$w_{ij} = \begin{cases} \bar{y}_{ij} & j < j(i) \\ \bar{y}_{ij} - \widehat{y}_{ij} & j = j(i) \\ 0 & j > j(i), \end{cases}$$

and set $\widetilde{x}^p = x^p$ and $\widetilde{y}^p = y^p + w$. Notice that for all $i \in N$,

$$w_{ij(i)} = \bar{y}_{ij(i)} - \widehat{y}_{ij(i)} = \bar{y}_{ij(i)} - \sum_{j \in M,\ j \leq j(i)} \bar{y}_{ij} + t_i = - \sum_{j \in M,\ j < j(i)} \bar{y}_{ij} + t_i \geq 0.$$

Thus, $w \geq 0$ and $\widetilde{y}^p$ is nonnegative for all $p = 1, \ldots, q$. Also, for all $i \in N$ we have

$$\sum_{j \in M} \widetilde{y}_{ij}^p = \sum_{j \in M} y_{ij}^p + \sum_{j \in M,\ j \leq j(i)} \bar{y}_{ij} - \widehat{y}_{ij(i)} = \sum_{j \in M} y_{ij}^p + t_i \leq x_i^p + t_i = \widetilde{x}_i^p + t_i.$$

Finally, for all $j \in M$ we have

$$\begin{aligned} \sum_{i \in N} \widetilde{y}_{ij}^p &= \sum_{i \in N} y_{ij}^p + \sum_{i \in N:\ j \leq j(i)} \bar{y}_{ij} - \sum_{i \in N:\ j = j(i)} \widehat{y}_{ij} \\ &\geq \widehat{d}_j + \sum_{i \in N:\ j \leq j(i)} \bar{y}_{ij} - \sum_{i \in N:\ j = j(i)} \widehat{y}_{ij} \\ &= \sum_{i \in N} \widehat{y}_{ij} + \sum_{i \in N:\ j \leq j(i)} \bar{y}_{ij} - \sum_{i \in N:\ j = j(i)} \widehat{y}_{ij} \\ &= \sum_{i \in N:\ j = j(i)} \widehat{y}_{ij} + \sum_{i \in N:\ j > j(i)} \widehat{y}_{ij} + \sum_{i \in N:\ j \leq j(i)} \bar{y}_{ij} - \sum_{i \in N:\ j = j(i)} \widehat{y}_{ij} \\ &= \sum_{i \in N} \bar{y}_{ij} \\ &\geq d_j. \end{aligned}$$

Thus, $(\widetilde{x}^p, \widetilde{y}^p) \in S_*(l, d, t)$ for all $p = 1, \ldots, q$ and are all distinct by the definition of $\widetilde{y}^p$. Furthermore, it is straightforward to verify that $\sum_{p=1}^{q} \lambda_p (\widetilde{x}^p, \widetilde{y}^p) = (\bar{x}, \bar{y})$. Hence $(\bar{x}, \bar{y})$ cannot be an extreme point of $\operatorname{conv}(S_*(l, d, t))$, yielding the required contradiction. $\square$

Corollary 22. *Minimizing a linear function over $S_*(l, d, t)$ can be done by solving $\mathcal{O}(n^m)$ linear programming problems.*

In other words, optimization over $S_*(l,d,t)$ can be done in polynomial time when m is fixed.

As an algorithmic implication, we can tweak the branch-and-bound procedure when we optimize over $S_*(l,d,t)$: whenever a node of the search-tree has m bounds of the form $x_i \geq l_i$, we can fix the production of the remaining suppliers to 0. However, our experimental experience indicates that a standard branch-and-cut solver does not need to branch that many times, rendering this approach inapplicable for practical purposes.

On the other hand, we can construct relaxations of S_* by considering the subsystem defined by a few customers, say two, and taking $h = 0$. By Theorem 21 and an argument similar to Proposition 17, a compact extended formulation is available for its convex hull from which strong valid inequalities for $\operatorname{conv}(S_*)$ may be devised.

2.6 *Computation*

We test the performance of the inequalities presented in Sections 2.3 and 2.4 on instances of the semi-continuous transportation problem described in Section 2.5. We address the effectivity of the cuts used alone or combined with CPLEX cuts, and the differences between semi-continuous and binary formulations.

Each instance is formulated in CPLEX either declaring all variables as semi-continuous or using auxiliary binary variables to enforce semi-continuity. In the latter case, we introduce constraints of the form $lz \leq x \leq Mz$, where z is a binary variable and $M > 0$ is a valid upper bound that yields an equivalent problem. Letting $\bar{d} := \sum_{j \in M} d_j$, $\bar{l} := \max_{i \in N}\{l_i\}$, $\bar{h} := \max_{i \in N,\, j \in M}\{h_{ij}\}$, and $\bar{t} := \max_{i \in N}\{t_i\}$, we set $M = \max\{\bar{d}, \bar{l}, \bar{h}\} + \bar{t}$.

Also, we consider the cases $t = 0$ and $t > 0$ separately. In the first case, we ignore the initial capacities and therefore cuts of the form (14) may be generated.

In the second case, valid cuts may be generated using the extended formulation (30). In both cases, to separate a fractional solution $(\bar{x}, \bar{y})$, we consider the inflow set corresponding to each customer $j \in M$ and we try to find a cut violated by $(\bar{x}, \bar{y}_j)$. Thus, we may add up to m cuts in a single round. For simplicity, cuts are added only at the root node. In addition, when $t = 0$, we also test an extended formulation where a vector π^j is appended for each $j \in M$. Adding the constraints that define W in Corollary 9 for each $j \in M$, we obtain an extended formulation where all the inequalities describing the inflow relaxation for each customer are already implied, and therefore there is no need to generate cuts on-the-fly. Even though an extended formulation is also available when $t > 0$, its size becomes a bottleneck even when solving the root relaxation, and thus it is not considered in our experimental setup.

In our experiments, we use $n \in \{30, 50, 80\}$ and $m \in \{30, 50, 80\}$. For each combination of these parameters, with the exception of $(n, m) = (80, 80)$ due to time limits, we generate 10 instances as follows:

- $l_i \sim \mathcal{U}[100, 500]\ \forall i \in N$
- $h_{ij} \sim \mathcal{U}\left[0, \frac{2}{m} l_i\right]\ \forall i \in N,\ \forall j \in M$
- $t_i \sim \mathcal{U}[10, 50]\ \forall i \in N$
- $d_j \sim \mathcal{U}\left[10\frac{n}{m}, 50\frac{n}{m}\right]\ \forall j \in M$
- $c_i \sim \mathcal{U}\left[40, 40 + \frac{1000}{l_i}\right]\ \forall i \in N$
- $\alpha_{ij} \sim \mathcal{U}[-10, 90]\ \forall i \in N,\ \forall j \in M,$

where $X \sim \mathcal{U}[a, b]$ means that X is a random variable following a uniform distribution on the interval $[a, b]$. Then, for each instance and for each formulation, we solve using CPLEX 12.2 default branch-and-cut (C), using only our cuts within branch-and-cut (U), using both CPLEX and user cuts (C+U), and solving the extended formulation (E) in the case $t = 0$. All experiments were carried out on a personal computer on a single thread running at 3.33 Ghz with 4 GB of RAM under Linux environment. A time limit of 1800 CPU seconds per instance is enforced.

2.6.1 The case $t = 0$

Table 1 shows the number of instances solved within the time limit, Table 2 shows the average number of explored nodes needed to reach optimality within CPLEX's default tolerance, and Table 3 shows the average time in CPU seconds required by such task. In all cases, columns n and m denote the size of the problem, columns *Semi-continuous* and *Binary* denote the type of formulation being considered, and columns C, U, C+U, and E denote the procedure being used, as explained above. All the averages are with respect to the number of instances that were solved. If no instance was solved for a particular combination of n and m, a dash "-" appears in the corresponding cell.

Table 1: Number of solved instances when $t = 0$.

n	m	*Semi-continuous*				*Binary*			
		C	U	C+U	E	C	U	C+U	E
30	30	10	10	10	10	10	10	10	10
30	50	10	10	10	10	10	10	10	10
30	80	10	10	10	8	10	10	10	10
50	30	10	10	10	10	10	10	10	10
50	50	10	10	10	9	9	10	10	10
50	80	4	10	5	10	1	10	4	10
80	30	5	10	4	10	10	10	10	10
80	50	0	5	0	10	4	10	2	10

Table 1 shows that not all instances were solved within the time limit. This may be a bit surprising, as the underlying problem structure is fairly simple and the number of variables does not exceed a few thousands. Adding our cuts alone and the extended formulation have the best performance in this sense, specially in the binary formulation where all instances where solved by both methods. As we can see from Table 2, the node count of the extended formulation is roughly one or two orders of magnitude smaller when compared to the other procedures in both models. Regarding time, from Table 3 we observe that the extended formulation is the best method in most cases when the semi-continuous formulation is used, whereas this approach is the best only in the largest instances when the binary formulation is considered. Among cutting procedures, adding only user cuts performs better than the rest in both formulations and is the only way to solve the largest instances within the time limit, with time reductions of up to one order of magnitude. Again, this can be somewhat surprising in the case of the binary formulation, as these cuts were not developed with binary variables in mind, and in this case we expected the presolve routines and flow covers to be particularly effective. On the other hand, combining these and CPLEX cuts decrease the overall performance and is comparable to the default solver.

Table 2: Number of nodes needed to prove optimality when $t = 0$.

n	m	*Semi-continuous*				*Binary*			
		C	U	C+U	E	C	U	C+U	E
30	30	3936.2	3266.5	2919.2	**72.8**	313.3	808.4	268.1	**32.7**
30	50	6246.6	4940.7	3653.6	**213.0**	493.3	731.9	618.7	**61.7**
30	80	11764.3	9330.0	6232.5	**930.3**	1142.3	1042.6	840.3	**206.1**
50	30	24045.9	23725.8	20548.9	**297.5**	1501.4	4545.5	1248.5	**84.8**
50	50	49407.0	40399.5	54556.6	**145.1**	3433.0	7446.9	2382.6	**135.1**
50	80	81456.8	159338.0	55918.8	**1019.9**	2086.0	23129.2	2621.3	**470.3**
80	30	56262.8	210466.0	48761.0	**192.8**	4049.3	30828.1	4731.6	**67.6**
80	50	-	438369.0	-	**332.3**	12265.2	114003.0	17426.5	**287.5**

Table 4 shows information regarding number of cuts. Column headers n, m, *Semi-continuous*, *Binary*, U, and C+U have the same meaning as in the previous tables. In addition, columns *Gen* denote the average number of user cuts that were generated, while columns *Appl* denote the average number of cuts that were actually applied. As we let CPLEX decide whether or not to apply user cuts that are generated by our separation routine, the numbers in these columns are different in

Table 3: CPU time needed to prove optimality when $t = 0$.

n	m	*Semi-continuous*				*Binary*			
		C	U	C+U	E	C	U	C+U	E
30	30	17.1	4.0	17.7	**3.4**	23.6	**2.0**	22.9	4.8
30	50	54.7	**12.9**	45.2	16.9	98.7	**4.6**	126.2	20.3
30	80	129.7	**33.9**	117.8	126.5	409.4	**9.3**	332.0	89.1
50	30	256.1	28.9	244.0	**10.0**	148.1	**11.3**	151.9	12.1
50	50	609.0	59.6	724.1	**13.6**	597.4	**21.7**	586.0	37.6
50	80	1399.7	316.7	1155.6	**135.5**	578.2	**98.5**	1144.6	165.3
80	30	924.7	168.2	1018.3	**8.2**	264.5	48.1	234.2	**8.2**
80	50	-	746.4	-	**28.5**	1438.7	354.6	1409.4	**45.7**

general.

Table 4: Number of cuts when $t = 0$.

n	m	*Semi-continuous*				*Binary*			
		U		C+U		U		C+U	
		Gen	*Appl*	*Gen*	*Appl*	*Gen*	*Appl*	*Gen*	*Appl*
30	30	96.0	24.3	96.0	76.4	60.0	15.9	51.0	10.3
30	50	154.6	69.4	154.6	132.8	100.0	45.7	70.0	29.1
30	80	262.0	137.0	262.0	218.3	128.0	69.9	120.0	79.3
50	30	87.4	7.4	87.4	49.6	67.0	6.7	102.6	37.9
50	50	147.6	18.8	147.6	101.9	123.4	15.7	180.5	98.5
50	80	239.9	45.5	239.8	181.2	231.9	43.9	316.5	178.5
80	30	88.3	5.9	87.0	42.8	58.8	4.7	94.0	23.2
80	50	147.2	9.0	-	-	101.3	6.9	173.5	75.0

First, note that more cuts are generated and applied in the semi-continuous formulation than in the binary formulation. Now, in both cases, the proportion of applied cuts with respect to the number of generated cuts is smaller when CPLEX cuts are turned off. Given the results in Table 3, just a few cuts are required to get a non-trivial improvement over the default solver, and the generation of more user cuts than needed seems to increase the running times.

2.6.2 The case $t > 0$

Tables 5, 6, and 7 are analogous to Tables 1, 2, and 3, respectively, with the difference that there is no column E as no extended formulation was tested in this case.

Table 5: Number of solved instances when $t > 0$.

n	m	*Semi-continuous*			*Binary*		
		C	U	C+U	C	U	C+U
30	30	10	10	10	10	10	10
30	50	7	10	8	10	10	10
30	80	2	9	8	10	10	10
50	30	0	10	3	10	10	10
50	50	0	9	1	9	10	8
50	80	0	0	0	1	10	4
80	30	0	5	0	4	10	6
80	50	0	0	0	0	10	0

From Table 5, we see that when $t > 0$, the instances become much harder than in the case $t = 0$. The performance of the semi-continuous formulation is quite poor in general. In contrast, the binary formulation is able to solve all small instances with any procedure, but only when CPLEX cuts are turned off it is possible to solve all large instances as well. Regarding explored nodes, Table 6 shows that the addition of user cuts may reduce the size of the search tree. With respect to computation times, we have that user cuts alone in the binary formulation outperforms all other methods, as shown in Table 7. This procedure is also the best with the semi-continuous formulations. Once again, combining CPLEX and user cuts is comparable to the default solver.

Table 6: Number of nodes needed to prove optimality when $t > 0$.

n	m	*Semi-continuous*			*Binary*		
		C	U	C+U	C	U	C+U
30	30	120194.0	**17069.0**	53748.7	603.1	852.3	**433.5**
30	50	137858.0	52383.7	**40540.5**	651.2	883.4	**476.3**
30	80	83006.5	112944.0	**33035.1**	1153.6	1103.7	**901.6**
50	30	-	**106912.0**	133777.0	3555.5	5927.4	**2596.4**
50	50	-	216427.0	**121396.0**	4991.4	10361.9	**3013.5**
50	80	-	-	-	7998.0	22166.4	**2496.8**
80	30	-	**714998.0**	-	17143.5	77894.5	**16998.0**
80	50	-	-	-	-	104097.0	-

Table 7: CPU time needed to prove optimality when $t > 0$.

n	*m*	*Semi-continuous*			*Binary*		
		C	U	C+U	C	U	C+U
30	30	343.5	**22.1**	211.7	26.6	**5.3**	26.1
30	50	759.3	**116.5**	311.0	74.0	**9.1**	66.4
30	80	945.6	**413.5**	488.8	224.4	**18.6**	202.4
50	30	-	**126.4**	962.1	168.8	**38.7**	157.5
50	50	-	**406.2**	1283.7	601.8	**93.8**	469.9
50	80	-	-	-	1427.6	**256.6**	872.4
80	30	-	**838.1**	-	542.2	**295.8**	804.0
80	50	-	-	-	-	**762.9**	-

Finally, Table 8 shows information regarding cuts, and it is analogous to Table 4. As in the case $t = 0$, when CPLEX and user cuts are combined, the solver attempts to generate and apply more cuts than needed, decreasing the overall performance as follows from Table 7.

Table 8: Number of cuts when $t > 0$.

n	*m*	*Semi-continuous*				*Binary*			
		U		C+U		U		C+U	
		Gen	*Appl*	*Gen*	*Appl*	*Gen*	*Appl*	*Gen*	*Appl*
30	30	101.7	51.2	98.7	86.6	48.0	25.6	42.0	21.1
30	50	159.0	92.1	161.9	143.1	74.9	43.8	94.9	51.8
30	80	253.0	169.4	265.0	235.1	120.0	92.4	152.0	104.7
50	30	89.8	29.5	94.3	87.7	84.0	29.1	101.2	69.4
50	50	147.7	68.8	150.0	144.0	147.9	70.4	182.9	136.5
50	80	-	-	-	-	239.3	102.2	308.8	222.5
80	30	86.4	30.0	-	-	79.0	25.1	111.2	85.3
80	50	-	-	-	-	130.6	37.9	-	-

As we have seen, the proposed valid inequalities, either in their original form or through an extended formulation when possible, are quite useful in solving this class of semi-continuous network flow problems. Although these cuts involve only the original variables of the problem, the introduction of binary variables seems to improve the overall performance.

2.7 *Concluding remarks*

In this chapter we have considered semi-continuous network flow problems from the complexity and polyhedral perspectives. In particular, we introduced the semi-continuous inflow set with variable upper bounds as a relaxation. Two particular cases of this set were considered, for which we presented complete descriptions of the convex hull in terms of linear inequalities and extended formulations. These inequalities proved to be quite efficient in solving a class of semi-continuous transportation problems. In fact, applying these cuts to a binary formulation of such problems turned out to be the most effective method.

We envision at least two possible venues of future research, mainly based on the semi-continuous inflow set. The first one is to consider finite upper bounds on semi-continuous variables. In this case, further connections with [14] may be established. Another direction is to consider semi-continuous inflows and outflows simultaneously. This would lead to a more general set that can be a better relaxation for appropriate problems.

Since the computational results demonstrate that the cuts are particularly effective with a binary formulation, it would be reasonable to try strengthening the cuts by having nonzero coefficients for the binary variables. Relaxations of $\{0, 1\}$ of the form $\{0\} \cup [1, \infty)$ might yield stronger yet tractable inflow relaxations.

CHAPTER III

FORBIDDEN VERTICES

3.1 *Introduction*

Given a nonempty rational polytope $P \subseteq \mathbb{R}^n$, we denote by $\operatorname{vert}(P)$, $\operatorname{faces}(P)$, and $\operatorname{facets}(P)$ the sets of vertices, faces, and facets of P, respectively, and we write $f(P) := |\operatorname{facets}(P)|$. We also denote by $\operatorname{xc}(P)$ the extension complexity of P, that is, the minimum number of inequalities in any linear extended formulation of P, i.e., a description of a polyhedron whose image under a linear map is P (see for instance [21].) Finally, given a set $X \subseteq \operatorname{vert}(P)$, we define $\operatorname{forb}(P, X) := \operatorname{conv}(\operatorname{vert}(P) \setminus X)$. This chapter is devoted to understanding the complexity of the forbidden-vertices problem defined below.

Definition 23. *Given a polytope $P \subseteq \mathbb{R}^n$, a set $X \subseteq \operatorname{vert}(P)$, and a vector $c \in \mathbb{R}^n$, the forbidden-vertices problem is to either assert $\operatorname{vert}(P) \setminus X = \varnothing$, or to return a minimizer of cx over $\operatorname{vert}(P) \setminus X$ otherwise.*

The work in this chapter is motivated by enumerative schemes for stochastic integer programs [34], where a series of potential solutions are evaluated and discarded from the search space. As we will see later, the problem is also related to finding different basic solutions to a linear program. An implementation of some results presented here is given in Chapter 4.

To address the complexity of the forbidden-vertices problem, it is crucial to distinguish between different encodings of a polytope.

Definition 24. *An explicit description of a polytope $P \subseteq \mathbb{R}^n$ is a system $Ax \leq b$ defining P. An implicit description of P is a separation oracle which, given a rational vector $x \in \mathbb{R}^n$, either asserts $x \in P$, or returns a valid inequality for P that is violated by x.*

Note that an extended formulation for P is a particular case of an implicit description. When P admits a separation oracle that runs in time bounded polynomially in the facet complexity of P and the encoding size of the point to separate, we say that P is tractable. We refer the reader to [55, Section 14] for a deeper treatment of the complexity of linear programming.

We also distinguish different encodings of a set of vertices.

Definition 25. *An explicit description of $X \subseteq \text{vert}(P)$ is the list of the elements in X. If $X = \text{vert}(F)$ for some face F of P, then an implicit description of X is an encoding of P and some valid inequality for P defining F.*

Below we summarize our main contributions.

- In Section 3.2, we show that the complexity of optimizing over $\text{vert}(P) \setminus X$ or describing $\text{forb}(P, X)$ changes significantly depending on the encoding of P and/or X. In most situations, however, the problem is hard.

- In Section 3.3 we consider the case of removing a list X of binary vectors from a 0-1 polytope P. When P is the unit cube, we present two compact extended formulations describing $\text{forb}([0,1]^n, X)$. We further extend this result and show that the forbidden-vertices problem is polynomially solvable for tractable 0-1 polytopes.

- Then in Section 3.4 we apply our results to the k-best problem and to binary all-different polytopes, showing the tractability of both. Finally, in Section 3.5, we also provide extensions to integral polytopes.

The complexity results of Sections 3.2 and 3.3 lead to the classification shown in Tables 9 and 10, depending on the encoding of P and X, and whether P has 0-1 vertices only or not. Note that $(*)$ is implied, for instance, by Theorem 40. Although

Table 9: Complexity classification for general polytopes.

		P	
		Explicit	Implicit
X	Explicit	$\mathcal{NP}$-hard (Thm. 33) Polynomial for fixed $\|X\|$ (Prop. 28)	$\mathcal{NP}$-hard for $\|X\| = 1$ (Thm. 31)
	Implicit	$\mathcal{NP}$-hard (Prop. 32)	$\mathcal{NP}$-hard $(*)$

Table 10: Complexity classification for 0-1 polytopes.

		P	
		Explicit	Implicit
X	Explicit	Polynomial	Polynomial (Thm. 38)
	Implicit	$(**)$	$\mathcal{NP}$-hard (Thm. 40)

we were not able to establish the complexity of $(**)$, Proposition 41 presents a tractable subclass.

In constructing extended formulations, disjunctive programming emerges as a practical powerful tool. The lemma below follows directly from [5] and the definition of extension complexity. We will frequently refer to it.

Lemma 26. *Let $P_1, \ldots, P_k$ be nonempty polytopes in $\mathbb{R}^n$. If $P_i = \{x \in \mathbb{R}^n | \; \exists y_i \in \mathbb{R}^{m_i} : E_i x + F_i y_i = h_i, \; y_i \geq 0\}$, then $\operatorname{conv}(\cup_{i=1}^k P_i) = \{x \in \mathbb{R}^n | \; \exists x_i \in \mathbb{R}^n, \; y_i \in \mathbb{R}^{m_i}, \; \lambda \in \mathbb{R}^k : \; x = \sum_{i=1}^k x_i, \; E_i x_i + F_i y_i = \lambda_i h_i, \; \sum_{i=1}^k \lambda_i = 1, \; y_i \geq 0, \; \lambda \geq 0\}$. In particular, we have $\operatorname{xc}\left(\operatorname{conv}(\cup_{i=1}^k P_i)\right) \leq \sum_{i=1}^k (\operatorname{xc}(P_i) + 1)$.*

3.2 General polytopes

We begin with some general results when $P \subseteq \mathbb{R}^n$ is an arbitrary polytope. The first question is how complicated $\operatorname{forb}(P, X)$ is with respect to P.

Proposition 27. *For each n, there exists a polytope $P_n \subseteq \mathbb{R}^n$ and a vertex $v_n \in \operatorname{vert}(P_n)$ such that P_n has $2n + 1$ vertices and $n^2 + 1$ facets, while $\operatorname{forb}(P_n, \{v_n\})$ has 2^n facets.*

Proof. Let $Q_n := [0, 1]^n \cap L$, where $L := \left\{x \in \mathbb{R}^n | \; \mathbf{1}x \leq \frac{3}{2}\right\}$ and $\mathbf{1}$ is the vector of ones. It has been observed [4] that Q_n has $2n + 1$ facets and $n^2 + 1$ vertices.

We translate Q_n and define $Q_n' := Q_n - \frac{1}{n}\mathbf{1} = \left[-\frac{1}{n}, 1-\frac{1}{n}\right]^n \cap L'$, where $L' := \left\{x \in \mathbb{R}^n \mid \mathbf{1}x \leq \frac{1}{2}\right\}$. Since Q_n' is a full-dimensional polytope having the origin in its interior, there is a one-to-one correspondence between the facets of Q_n' and the vertices of its polar $P_n := (Q_n')^*$ and vice versa. In particular, P_n has n^2+1 facets and $2n+1$ vertices. Let $v \in \text{vert}(P_n)$ be the vertex associated with the facet of Q_n' defined by L'. From polarity, we have $\text{forb}(P_n, \{v\})^* = \left[-\frac{1}{n}, 1-\frac{1}{n}\right]^n$. Thus $\text{forb}(P_n, \{v\})^*$ is a full-dimensional polytope with the origin in its interior and 2^n vertices. By polarity, we obtain that $\text{forb}(P_n, \{v\})$ has 2^n facets. □

Note that the above result only states that $\text{forb}(P, X)$ may need exponentially many inequalities to be described, which does not constitute a proof of hardness. Such a result is provided by Theorem 33 at the end of this section. We first show that $\text{forb}(P, X)$ has an extended formulation of polynomial size in $f(P)$ when both P and X are given explicitly and the cardinality of X is fixed.

Proposition 28. *Suppose $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. Using this description of P, and an explicit list of vertices X, we can construct an extended formulation of* $\text{forb}(P, X)$ *that requires at most $f(P)^{|X|+1}$ inequalities, i.e.,* $\text{xc}(\text{forb}(P, X)) \leq f(P)^{|X|+1}$.

Proof. Let $X = \{v_1, \dots, v_{|X|}\}$ and define $\mathcal{F}_X := \{F_1 \cap \dots \cap F_{|X|} \mid F_i \in \text{facets}(P),\ v_i \notin F_i,\ i = 1, \dots, |X|\}$. We claim

$$\text{forb}(P, X) = \text{conv}\left(\cup_{F \in \mathcal{F}_X} F\right).$$

Indeed, let $w \in \text{vert}(P) \setminus X$. For each $i = 1, \dots, |X|$, there exists $F_i \in \text{facets}(P)$ such that $w \in F_i$ and $v_i \notin F_i$. Therefore, letting $F := F_1 \cap \dots \cap F_{|X|}$, we have $F \in \mathcal{F}_X$ and $w \in F$, proving the forward inclusion. For the reverse inclusion, consider $F \in \mathcal{F}_X$. By definition, F is a face of P that does not intersect X, and hence $F \subseteq \text{forb}(P, X)$.

By Lemma 26, we have $\mathrm{xc}(\mathrm{forb}(P, X)) \leq \sum_{F \in \mathcal{F}_X}(\mathrm{xc}(F) + 1)$. Since $\mathrm{xc}(F) \leq f(F) \leq f(P) - 1$ for each proper face F of P and $|\mathcal{F}_X| \leq f(P)^{|X|}$, the result follows. $\square$

Note that when $X = \{v\}$, the above result reduces $\mathrm{forb}(P, \{v\})$ to the convex hull of the union of the facets of P that are not incident to v, which is a more intuitive result. Actually, we can expect describing $\mathrm{forb}(P, X)$ to be easier when the vertices in X are "far" thus can be removed "independently", and more complicated when they are "close". Proposition 28 can be refined as follows.

The graph of a polytope P, or the 1-skeleton of P, is a graph G with vertex set $\mathrm{vert}(P)$ such that two vertices are adjacent in G if and only if they are adjacent in P.

Proposition 29. *Let G be the graph of P. Let $X \subseteq \mathrm{vert}(P)$ and let $(X_1, \ldots, X_m)$ be a partition of X such that X_i and X_j are independent in G, i.e., there is no edge connecting X_i to X_j, for all $1 \leq i < j \leq m$. Then*

$$\mathrm{forb}(P, X) = \bigcap_{i=1}^{m} \mathrm{forb}(P, X_i).$$

Proof. We only need to show $\mathrm{forb}(P, X) \supseteq \bigcap_{i=1}^{m} \mathrm{forb}(P, X_i)$. For this, it is enough to show that $\max\{cx : x \in \mathrm{forb}(P, X)\} \geq \max\{cx : x \in \bigcap_{i=1}^{m} \mathrm{forb}(P, X_i)\}$ for each c. Given c, let v be an optimal solution to the maximization problem in the right-hand side, and let $W \subseteq \mathrm{vert}(P)$ be the set of vertices w of P such that $cw \geq cv$. Observe that W induces a connected subgraph of the graph G of P since the simplex method applied to $\max\{cx : x \in P\}$ starting from a vertex in W visits elements in W only. Hence, due to the independence of $X_1, \ldots, X_m$, either there is some $w \in W$ with $w \notin X_1 \cup \cdots \cup X_m$, in which case we have $w \in \mathrm{forb}(P, X)$ and $cw \geq cv$ as desired, or $W \subseteq X_i$ for some i, which yields the contradiction $v \in \mathrm{forb}(P, X_i) \subseteq \mathrm{forb}(P, W)$ with $cx < cv$ for all $x \in \mathrm{vert}(P) \setminus W$. $\square$

Conversely, we may be tempted to argue that if $\mathrm{forb}(P,X) = \mathrm{forb}(P,X_1) \cap \mathrm{forb}(P,X_2)$, then X_1 and X_2 are "far". However, this is not true in general. For instance, consider P being a simplex. Then any $X \subseteq \mathrm{vert}(P)$ is a clique in the graph of P, and yet $\mathrm{forb}(P,X) = \mathrm{forb}(P,X_1) \cap \mathrm{forb}(P,X_2)$ for any partition (X_1,X_2) of X.

Proposition 29 generalizes the main result of [40] regarding cropped cubes. Moreover, the definition of being "croppable" in [40] in the case of the unit cube coincides with the independence property of Proposition 29.

Recall that a vertex of an n-dimensional polytope is simple if it is contained in exactly n facets. Proposition 29 also implies the following well-known fact.

Corollary 30. *If X is independent in the graph of P and all its elements are simple, then*

$$\mathrm{forb}(P,X) = P \cap \bigcap_{v \in X} H_v,$$

where H_v is the half-space defined by the n neighbors of v that does not contain v.

Proof. The result follows from Proposition 29 since, as X is simple, we have immediately $\mathrm{forb}(P,\{v\}) = P \cap H_v$ for any $v \in X$. $\square$

Observe that when P is given by an extended formulation or a separation oracle, $f(P)$ may be exponentially large with respect to the size of the encoding, and the bound given in Proposition 28 is not interesting. In fact, in this setting and using recent results on the extension complexity of the cut polytope [20], we show that removing a single vertex can render an easy problem hard.

Let $K_n = (V_n, E_n)$ be the complete graph on n nodes. We denote by $\mathrm{CUT}(n)$, $\mathrm{CUT}^0(n)$, and st-$\mathrm{CUT}(n)$ the convex hull of the characteristic vectors of all cuts, nonempty cuts, and st-cuts of K_n, respectively.

Theorem 31. *For each n, there exists a set $S_n \subseteq \mathbb{R}^{n(n-1)/2}$ with $|S_n| = 2^{n-1} + n - 1$ and a point $v_n \in S_n$ such that linear optimization over S_n can be done in polynomial* *time and* $\text{xc}(\text{conv}(S_n))$ *is polynomially bounded, but linear optimization over* $S_n \setminus \{v_n\}$ *is* $\mathcal{NP}$*-hard and* $\text{xc}(\text{conv}(S_n \setminus \{v_n\}))$ *grows exponentially.*

Proof. Let $T_n := \{n^2\mathbf{1}_e |\ e \in E_n\}$, where $\mathbf{1}_e$ is the e-th unit vector, and define $S_n := \text{vert}\left(\text{CUT}^0(n)\right) \cup T_n$.

We have that linear optimization over S_n can be done in polynomial time. To see this, suppose we are minimizing cx over S_n. Let x^T and x^C be the best solution in T_n and $\text{CUT}^0(n)$, respectively. Note that computing x^T is trivial, and if c has a negative component, then x^T is optimal. Otherwise, c is nonnegative and x^C can be found with a max-flow/min-cut algorithm. Then the best solution among x^T and x^C is optimal. Now, consider the dominant of $\text{CUT}^0(n)$ defined as $\text{CUT}^0(n)_+ := \text{CUT}^0(n) + \mathbb{R}_+^{n(n-1)/2}$. From [10], we have that $\text{CUT}^0(n)_+$ is an unbounded polyhedron having the same vertices as $\text{CUT}^0(n)$, and moreover, it has an extended formulation of polynomial size in n. Let $L := \{x \in \mathbb{R}^{n(n-1)/2} |\ \sum_{e \in E_n} x_e \leq n^2\}$. Then $\text{CUT}^0(n)_+ \cap L$ is a polytope having two classes of vertices: those corresponding to $\text{vert}\left(\text{CUT}^0(n)\right)$ and those belonging to the hyperplane defining L. Let W be the latter set. Since $\text{conv}(W) \subsetneq \text{conv}(T_n)$, we obtain $\text{conv}(S_n) = \text{conv}\left(\text{CUT}^0(n) \cup T_n\right) = \text{conv}\left((\text{CUT}^0(n) \cup W) \cup T_n)\right) = \text{conv}\left((\text{CUT}^0(n)_+ \cap L) \cup T_n\right)$. Applying disjunctive programming in the last expression yields a compact extended formulation for $\text{conv}(S_n)$.

Now, let v_n be any point from T_n, say the one corresponding to $\{s, t\} \in E$. We claim that linear optimization over $S_n \setminus \{v_n\}$ is $\mathcal{NP}$-hard. To prove this, consider an instance of $\max\{cx |\ x \in st\text{-CUT}(n)\}$, where c is a positive vector. Let $\bar{c} := \max\{c_e |\ e \in E\}$. Let d be obtained from c as

$$d_e = \begin{cases} c_e & e \neq \{s,t\} \\ c_e + \bar{c}n^2 & e = \{s,t\} \end{cases}$$

and consider the problem $\max\{dx |\ x \in S_n \setminus \{v_n\}\}$. We have that every optimal solution to this problem must satisfy $x_{st} = 1$. Indeed, if $x \in T_n \setminus \{v_n\}$, then for some $e \in E_n \setminus \{\{s,t\}\}$ we have $dx = d_e x_e = c_e n^2$. If $x \in \text{vert}(\text{CUT}^0(n))$ is not an st-cut, then $x_{st} = 0$ and thus $dx \leq \bar{c}n^2$. On the other hand, if x is an st-cut, then $x_{st} = 1$ and thus $dx \geq d_{st}x_{st} = c_{st} + \bar{c}n^2$. Therefore $x_{st} = 1$ in any optimal solution, and in particular, such a solution must define an st-cut of maximum weight. Finally, since $x_{st} \leq 1$ defines a face of $\text{conv}(S_n \setminus \{v_n\})$ and $\text{conv}(S_n \setminus \{v_n\}) \cap \{x \in \mathbb{R}^{n(n-1)/2} | \ x_{st} = 1\} = st\text{-CUT}(n)$, we conclude that $\text{xc}(\text{conv}(S_n \setminus \{v_n\}))$ is exponential in n, for otherwise applying disjunctive programming over all pairs of nodes s and t would yield an extended formulation for $\text{CUT}(n)$ of polynomial size, contradicting the results in [20]. □

Contrasting Proposition 28 and Theorem 31 shows that the complexity of optimization over $\text{forb}(P, X)$ depends on the encoding of P. On the other hand, in all cases analyzed so far, X has been explicitly given as a list. Now we consider the case where $X = \text{vert}(F)$ for some face F of P.

Proposition 32. *Given a polytope $P \subseteq \mathbb{R}^n$ and a face F, both described in terms of the linear inequalities defining them, optimizing a linear function over* $\text{vert}(P) \setminus \text{vert}(F)$ *is $\mathcal{NP}$-hard. Moreover,* $\text{xc}(\text{conv}(\text{vert}(P) \setminus \text{vert}(F)))$ *cannot be polynomially bounded in the encoding length of the inequality description of P and thus not in n.*

Proof. Let $a \in \mathbb{Z}^n_+$ and $b \in \mathbb{Z}_+$, and consider the binary knapsack set $S := \{x \in \{0,1\}^n | \ ax \leq b\}$. Let $P := \{x \in [0,1]^n | \ 2ax \leq 2b+1\}$ and note that $S = P \cap \mathbb{Z}^n$. It is straightforward to verify that $x \in \text{vert}(P)$ is fractional if and only if $2ax = 2b+1$. Then, if F is the facet of P defined by the previous constraint, we have $S = \text{vert}(P) \setminus \text{vert}(F)$. The second part of the statement is a direct consequence of [51] using multipliers 4^i as discussed after Remark 3.4 of that reference. □

It follows from Theorem 31 and Proposition 32 that only when P and X are explicitly given there is hope for efficient optimization over $\text{forb}(P, X)$.

In a similar vein, when the linear description of P is provided, we can consider the vertex-enumeration problem, which consists of listing all the vertices of P. We say that such a problem is solvable in polynomial time if there exists an algorithm that returns the list in time bounded by a polynomial of n, $f(P)$, and the output size $|\text{vert}(P)|$. In [31] it is shown that given a partial list of vertices, the decision problem "is there another vertex?" is $\mathcal{NP}$-hard for (unbounded) polyhedra, and in [9] this result is strengthened to polyhedra having 0-1 vertices only. Building on these results, we show hardness of the forbidden-vertices problem (Def. 23) for general polytopes.

Theorem 33. *The forbidden-vertices problem is $\mathcal{NP}$-hard, even if both P and X are explicitly given.*

Proof. Let $Q = \{x \in \mathbb{R}^n : \ Ax = b, \ x \geq 0\}$ be an unbounded polyhedron such that $\text{vert}(Q) \subseteq \{0,1\}^n$. In [9], it is shown that given the linear description of Q and a list $X \subseteq \text{vert}(Q)$, it is $\mathcal{NP}$-hard to decide whether $X \neq \text{vert}(Q)$. Let P be the polytope obtained by intersecting Q with the half-space defined by $\sum_{i=1}^{n} x_i \leq n+1$, and let F be the facet of P associated with this constraint. Then we have $\text{vert}(P) = \text{vert}(Q) \cup \text{vert}(F)$, $\sum_{i=1}^{n} x_i \leq n$ for $x \in \text{vert}(Q)$, and $\sum_{i=1}^{n} x_i = n+1$ for $x \in \text{vert}(F)$. Now, given the description of P and a list $X \subseteq \text{vert}(Q) \subseteq \text{vert}(P)$, consider the instance of the forbidden-vertices problem $\min\{\sum_{i=1}^{n} x_i : \ x \in \text{vert}(P) \setminus X\}$. The optimal value is equal to $n+1$ if and only if $X = \text{vert}(Q)$. Since the reduction is clearly polynomial, the result follows. □

In fact, it also follows from [9] that the forbidden-vertices problem for general polytopes becomes hard already for $|X| = n$. Fortunately, the case of 0-1 polytopes is amenable to good characterizations.

3.3 *0-1 polytopes*

We consider polytopes having binary vertices only. We show that $\mathrm{forb}(P, X)$ is tractable as long as P is and X is explicitly given. Our results for $P = [0,1]^n$ allow us to obtain tractability in the case of general 0-1 polytopes.

3.3.1 The 0-1 cube

In this subsection we have $P = [0,1]^n$, and therefore $\mathrm{vert}(P) = \{0,1\}^n$. We show the following result.

Theorem 34. *Let X be a list of n-dimensional binary vectors. Then $\mathrm{xc}(\mathrm{forb}([0,1]^n, X)) \leq \mathcal{O}(n|X|)$.*

For this, we present two extended formulations involving $\mathcal{O}(n|X|)$ variables and constraints. The first one is based on an identification between nonnegative integers and binary vectors. The second one is built by recursion and lays ground for a simple combinatorial algorithm to optimize over $\mathrm{forb}([0,1]^n, X)$ and for an extension to remove vertices from general 0-1 polytopes.

3.3.1.1 First extended formulation

Let $N := \{1, \ldots, n\}$ and $\mathcal{N} := \{0, \ldots, 2^n - 1\}$. There exists a bijection between $\{0,1\}^n$ and $\mathcal{N}$ given by the mapping $\sigma(v) := \sum_{i \in N} 2^{i-1} v_i$ for all $v \in \{0,1\}^n$. Therefore, we can write $\{0,1\}^n = \{v^0, \ldots, v^{2^n - 1}\}$, where v^k gives the binary expansion of k for each $k \in \mathcal{N}$, that is, $v^k = \sigma^{-1}(k)$. Let $X = \{v^{k_1}, \ldots, v^{k_m}\}$, where without loss of generality we assume $k_l < k_{l+1}$ for all $l = 1, \ldots, m-1$. Also, let $\mathcal{N}_X := \{k \in \mathcal{N} \mid v^k \in X\}$. Then we have

$$\{0,1\}^n \setminus X = \left\{ x \in \{0,1\}^n \mid \sum_{i \in N} 2^{i-1} x_i \notin \mathcal{N}_X \right\}.$$

Now, for integers a and b, let

$$K(a,b) = \left\{ x \in \{0,1\}^n \mid a \leq \sum_{i \in N} 2^{i-1} x_i \leq b \right\}.$$

If $b < a$, then $K(a, b)$ is empty. Set $k_0 = -1$ and $k_{m+1} = 2^n$. Then we can write

$$\{0,1\}^n \setminus X = \bigcup_{l=0}^{m} K(k_l + 1, k_{l+1} - 1).$$

Thus

$$\text{forb}([0,1]^n, X) = \text{conv}\left(\bigcup_{l=0}^{m} \text{conv}(K(k_l + 1, k_{l+1} - 1))\right). \tag{35}$$

For $k \in \mathcal{N}$, let $N^k := \{i \in N \mid v_i^k = 1\}$. From [46] we have

$$\text{conv}(K(a,b)) = \left\{ x \in [0,1]^n : \begin{array}{cc} \sum_{j \notin N^a \mid j > i} x_j \geq 1 - x_i & \forall i \in N^a \\ \sum_{j \in N^b \mid j > i} (1 - x_j) \geq x_i & \forall i \notin N^b \end{array} \right\},$$

thus $\text{conv}(K(a,b))$ has $\mathcal{O}(n)$ facets. Finally, combining this and (35), by Lemma 26, we have that $\text{forb}([0,1]^n, X)$ can be described by an extended formulation having $\mathcal{O}(n|X|)$ variables and constraints.

3.3.1.2 *Second extended formulation*

Given $X \subseteq \{0,1\}^n$, let X' denote the projection of X onto the first $n-1$ coordinates. Also, let $\widehat{X} := \widetilde{X} \setminus X$, where $\widetilde{X}$ is constructed from X by flipping the last coordinate of each of its elements. The result below is key in giving a recursive construction of $\text{forb}([0,1]^n, X)$.

Proposition 35. *$\{0,1\}^n \setminus X = \left[(\{0,1\}^{n-1} \setminus X') \times \{0,1\}\right] \cup \widehat{X}$.*

Proof. Given $v \in \{0,1\}^n$, let $v' \in \{0,1\}^{n-1}$ and $\widetilde{v} \in \{0,1\}^n$ be the vectors obtained from v by removing and by flipping its last coordinate, respectively.

Let $v \in \{0,1\}^n \setminus X$. If $\widetilde{v} \in X$, since $v \notin X$, we have $v \in \widehat{X}$. Otherwise $v' \notin X'$, and thus $v \in (\{0,1\}^{n-1} \setminus X') \times \{0,1\}$.

For the converse, note that $\widehat{X} \subseteq \{0,1\}^n \setminus X$. Finally, if $v \in (\{0,1\}^{n-1} \setminus X') \times \{0,1\}$, then $v' \notin X'$ and thus $v \notin X$. $\square$

The second proof of Theorem 34 follows from Proposition 35 by induction. Suppose that $\mathrm{forb}([0,1]^{n-1}, X')$ has an extended formulation with at most $(n-1)(|X'|+4)$ inequalities, which holds for $n=2$. Then we can describe $\mathrm{forb}([0,1]^{n-1}, X') \times \{0,1\}$ using at most $(n-1)(|X'|+4)+2$ inequalities. Since the polytope $\mathrm{conv}(\widehat{X})$ requires at most $|\widehat{X}|$ inequalities in an extended formulation, we obtain an extended formulation for $\mathrm{forb}([0,1]^n, X)$ of size no more than $[(n-1)(|X'|+4)+2+1]+[|\widehat{X}|+1] \leq n(|X|+4)$.

3.3.2 General 0-1 polytopes

In this subsection we analyze the general 0-1 case. We show that the encoding of X plays an important role in the complexity of the problem.

3.3.2.1 Explicit X

In order to prove tractability of the forbidden vertices problem corresponding to general 0-1 tractable polytopes, we introduce the notion of X-separating faces for the 0-1 cube.

Definition 36. *Given $X \subseteq \{0,1\}^n$, we say that $\mathcal{F} \subseteq \mathrm{faces}([0,1]^n)$ is X-separating if $\{0,1\}^n \setminus X = \cup_{F\in\mathcal{F}} F \cap \{0,1\}^n$. We denote by $\mu(X)$ the minimal cardinality of an X-separating set.*

Clearly, if $\mathcal{F}$ is X-separating, then

$$\min\{cx \mid x \in \{0,1\}^n \setminus X\} = \min_{F\in\mathcal{F}} \min\{cx \mid x \in F \cap \{0,1\}^n\}.$$

Thus, if we can find an X-separating family of cardinality bounded by a polynomial on n and $|X|$, then we can optimize in polynomial time over $\{0,1\}^n \setminus X$ by solving the inner minimization problem for each $F \in \mathcal{F}$ and then picking the smallest value.

Proposition 37. *For every nonempty set $X \subseteq \{0,1\}^n$, we have $\mu(X) \leq n|X|$.*

Proof. For each $y \in \{0,1\}^n \setminus X$, let $0 \leq k \leq n-1$ be the size of the longest common prefix between y and any element of X, and consider the face $F = F(y) := \{x \in [0,1]^n | \, x_i = y_i \, \forall 1 \leq i \leq k+1\} = (y_1, \ldots, y_k, y_{k+1}) \times [0,1]^{n-k-1}$. Then the collection $\mathcal{F} := \{F(y) | \, y \in \{0,1\}^n \setminus X\}$ is X-separating since any $y \in \{0,1\}^n \setminus X$ belongs to $F(y)$ and no element of X lies in any $F(y)$ by maximality of k. Clearly, $|\mathcal{F}| \leq n|X|$ since each face in $\mathcal{F}$ is of the form $(v_1, \ldots, v_k, 1 - v_{k+1}) \times [0,1]^{n-k-1}$ for some $v \in X$. $\square$

In other words, letting X^i be the projection of X onto the first i components and $\widehat{X}^i := (X^{i-1} \times \{0,1\}) \setminus X^i$, where $\widehat{X}^1 := \{0,1\} \setminus X^1$, we have

$$\{0,1\}^n \setminus X = \bigcup_{i=1}^{n} \left[\widehat{X}^i \times \{0,1\}^{n-i} \right].$$

Moreover, it also follows from the proof of Proposition 37 that $\mu(X)$ is at most the number of neighbors of X since if $(v_1, \ldots, v_k, 1 - v_{k+1}, v_{k+2}, \ldots, v_n)$ is a neighbor of $v \in X$ that also lies in X, then the face $\{(v_1, \ldots, v_k, 1 - v_{k+1})\} \times [0,1]^{n-k-1}$ in not included in $\mathcal{F}$ in the construction above.

Now, let $P \subseteq \mathbb{R}^n$ be an arbitrary 0-1 polytope. Note that $\text{vert}(P) \setminus X = \text{vert}(P) \cap (\{0,1\}^n \setminus X)$. On the other hand, if $\mathcal{F} \subseteq \text{faces}([0,1]^n)$ is X-separating, then $\{0,1\}^n \setminus X = \cup_{F \in \mathcal{F}} F \cap \{0,1\}^n$. Combining these two expressions, we get

$$\text{vert}(P) \setminus X = \bigcup_{F \in \mathcal{F}} \text{vert}(P) \cap F \cap \{0,1\}^n = \bigcup_{F \in \mathcal{F}} P \cap F \cap \{0,1\}^n.$$

Note that since P has 0-1 vertices and F is a face of the unit cube, then $P \cap F$ is a 0-1 polytope. Moreover, if P is tractable, so is $P \cap F$. Recalling that $\mu(X) \leq n|X|$ from Proposition 37, we obtain

Theorem 38. *If $P \subseteq \mathbb{R}^n$ is a tractable 0-1 polytope, then the forbidden-vertices problem is polynomially solvable.*

In fact, a compact extended formulation for $\text{vert}(P) \setminus X$ is available when P has one.

Proposition 39. *For every 0-1 polytope P and for every nonempty set $X \subseteq \operatorname{vert}(P)$, we have*

$$\operatorname{xc}(\operatorname{forb}(P, X)) \leq \mu(X)(\operatorname{xc}(P) + 1).$$

Proof. The result follows from

$$\operatorname{forb}(P, X) = \operatorname{conv}\left(\bigcup_{F \in \mathcal{F}} P \cap F \cap \{0,1\}^n\right) = \operatorname{conv}\left(\bigcup_{F \in \mathcal{F}} F\right),$$

Lemma 26, and $\operatorname{xc}(F) \leq \operatorname{xc}(P)$ for any face F of P. □

Observe that when P is tractable but its facet description is not provided, Theorem 38 is in contrast to Theorem 31. Having all vertices with at most two possible values for each component is crucial to retain tractability when X is given as a list. However, when X is given by a face of P, the forbidden-vertices problem can become intractable even in the 0-1 case.

3.3.2.2 Implicit X

Let $\operatorname{TSP}(n)$ denote the convex hull of the characteristic vectors of Hamiltonian cycles in the complete graph K_n. Also, let $\operatorname{SUB}(n)$ denote the subtour-elimination polytope for K_n with edge set E_n.

Theorem 40. *For each n, there exists a 0-1 polytope $P_n \subseteq \mathbb{R}^{n(n-1)/2}$ and a facet $F_n \in \operatorname{facets}(P_n)$ such that linear optimization over P_n can be done in polynomial time and $\operatorname{xc}(P_n)$ is polynomially bounded, but linear optimization over $\operatorname{vert}(P_n) \setminus \operatorname{vert}(F_n)$ is $\mathcal{NP}$-hard and $\operatorname{xc}(\operatorname{forb}(P_n, \operatorname{vert}(F_n)))$ grows exponentially.*

Proof. Given a positive integer n, consider $T_n^+ := \{x \in \{0,1\}^{E_n} | \sum_{e \in E_n} x_e = n+1\}$, $T_n^- := \{x \in \{0,1\}^{E_n} | \sum_{e \in E_n} x_e = n-1\}$, and $H_n := \operatorname{TSP}(n) \cap \{0,1\}^{E_n}$. The idea is to "sandwich" H_n between T_n^- and T_n^+ to obtain tractability, and then remove T_n^- to obtain hardness.

We first show that linear optimization over $T_n^- \cup H_n \cup T_n^+$ is polynomially solvable. Given $c \in \mathbb{R}^{n(n-1)/2}$, consider $\max\{cx| \ x \in T_n^- \cup H_n \cup T_n^+\}$. Let x^- and x^+ be the best solution in T_n^- and T_n^+, respectively, and note that x^- and x^+ are trivial to find. Let m be the number of nonnegative components of c. If $m \geq n+1$, then x^+ is optimal. If $m \leq n-1$, then x^- is optimal. If $m = n$, let $x^n \in \{0,1\}^{E_n}$ have a 1 at position e if and only if $c_e \geq 0$. If x^n belongs to H_n, which is easy to verify, then it is optimal. Otherwise either x^- or x^+ is an optimal solution.

Now we show that linear optimization over $H_n \cup T_n^+$ is $\mathcal{NP}$-hard. Given $c \in \mathbb{R}^{n(n-1)/2}$ with $c > 0$, consider $\min\{cx| \ x \in H_n\}$. Let $\bar{c} := \max\{c_e| \ e \in E_n\}$ and define $d_e := c_e + n\bar{c}$. Consider $\min\{dx| \ x \in H_n \cup T_n^+\}$. For any $x \in T_n^+$, we have $dx = (n+1)n\bar{c} + cx > (n+1)n\bar{c}$. For any $x \in H_n$, we have $dx = n^2\bar{c} + cx \leq n^2\bar{c} + n\bar{c} = (n+1)n\bar{c}$. Hence, the optimal solution to the latter problem belongs to H_n and defines a tour of minimal length with respect to c.

Letting $P_n := \text{conv}(T_n^- \cup H_n \cup T_n^+)$, we have that P_n is a tractable 0-1 polytope, $\sum_{e \in E_n} x_e \geq n-1$ defines a facet F_n of P_n, and $\text{vert}(P_n) \setminus \text{vert}(F_n) = H_n \cup T_n^+$, which is an intractable set. Now, since $\text{forb}(P_n, \text{vert}(F_n)) = \text{conv}(H_n \cup T_n^+)$, we have that $\sum_{e \in E_n} x_e \geq n$ defines a facet of $\text{forb}(P_n, \text{vert}(F_n))$ and $\text{forb}(P_n, \text{vert}(F_n)) \cap \{x \in \mathbb{R}^{n(n-1)/2}| \ \sum_{e \in E_n} x_e = n\} = \text{TSP}(n)$. Therefore, $\text{xc}(\text{forb}(P_n, \text{vert}(F_n)))$ is exponential in n [54]. It remains to show that $\text{xc}(P_n)$ is polynomial in n.

Let $T_n := \{x \in \{0,1\}^{E_n}| \ \sum_{e \in E_n} x_e = n\}$ and let $\overline{H}_n := T_n \setminus H_n$ be the set of incidence vectors of n-subsets of E_n that do not define a Hamiltonian cycle. Given $x \in \{0,1\}^{E_n}$, let $N(x)$ be the set of neighbors of x in $[0,1]^{E_n}$, let $L(x)$ be the half-space spanned by $N(x)$ that does not contain x, and let $C(x) := [0,1]^{E_n} \setminus L(x)$. Finally, let $\Delta_n := \text{conv}(T_n^- \cup T_n \cup T_n^+) = \{x \in [0,1]^{E_n}| \ n-1 \leq \sum_{e \in E_n} x_e \leq n+1\}$.

We claim that $P_n = \text{conv}(T_n^- \cup \text{SUB}(n) \cup T_n^+)$ holds. By definition, we have $P_n \subseteq \text{conv}(T_n^- \cup \text{SUB}(n) \cup T_n^-)$. To show the reverse inclusion, it suffices to show $\text{SUB}(n) \subseteq P_n$. Note that any two distinct elements in T_n can have at most $|E_n| - 2$

tight inequalities in common from those defining Δ_n. Thus, T_n defines an independent set in the graph of Δ_n. Moreover, for each $x \in T_n$ the set of neighbors in Δ_n is $N(x)$ and thus all vertices in T_n are simple. As $\overline{H}_n \subseteq T_n$, we have that $\overline{H}_n$ is simple and independent, and by Corollary 30 we have

$$P_n = \Delta_n \cap \bigcap_{x \in \overline{H}_n} L(x) = \Delta_n \setminus \bigcup_{x \in \overline{H}_n} C(x).$$

Since $\mathrm{SUB}(n) \subseteq \Delta_n$, from the second equation above, it suffices to show $C(x) \cap \mathrm{SUB}(n) = \varnothing$ for all $x \in \overline{H}_n$. For this, note that for any $x \in \overline{H}_n$, there exists a set $\varnothing \neq S \subsetneq V_n$ such that $x(\delta(S)) \leq 1$, which implies $y(\delta(S)) \leq 2$ for all $y \in N(x)$. Thus $C(x) \cap \mathrm{SUB}(n) = \varnothing$ as $x(\delta(S)) \geq 2$ is valid for $\mathrm{SUB}(n)$.

Finally, applying disjunctive programming and since $\mathrm{xc}(\mathrm{SUB}(n))$ is polynomial in n [60], we conclude that P_n has an extended formulation of polynomial size. $\square$

To conclude this section, consider the case where P is explicitly given and X is given as a facet of P. Although we are unable to establish the complexity of the forbidden-vertices problem in this setting, we present a tractable case and discuss an extension.

Proposition 41. *Let $P = \{x \in \mathbb{R}^n |\ Ax \leq b\}$ be a 0-1 polytope, where A is TU and b is integral. Let F be the face of P defined by $a_i x = b_i$. Then*

$$\mathrm{forb}(P, \mathrm{vert}(F)) = P \cap \{x \in \mathbb{R}^n |\ a_i x \leq b_i - 1\}.$$

Proof. We have

$$\mathrm{vert}(P) \setminus \mathrm{vert}(F) = P \cap \{x \in \{0,1\}^n |\ a_i x \leq b_i - 1\}.$$

Since A is TU and b in integral, the set $P \cap \{x \in \mathbb{R}^n |\ a_i x \leq b_i - 1\}$ is an integral polyhedron contained in P, which is a 0-1 polytope. $\square$

Since any face is the intersection of a subset of facets, the above result implies that removing a single face can be efficiently done by disjunctive programming in the context of Proposition 41. Also, if we want to remove a list of facets, that is, $X = \cup_{F \in \mathcal{F}} \text{vert}(F)$ and $\mathcal{F}$ is a subset of the facets of P, then we can solve the problem by removing one facet at a time. However, if $\mathcal{F}$ is a list of faces, then the problem becomes hard in general.

Proposition 42. *If $\mathcal{F}$ is a list of faces of $[0,1]^n$, then optimizing a linear function over $\{0,1\}^n \setminus \cup_{F \in \mathcal{F}} \text{vert}(F)$ is $\mathcal{NP}$-hard.*

Proof. Let $G = (V, E)$ be a graph. Consider the problem of finding a minimum cardinality vertex cover of G, which can be formulated as

$$\begin{array}{rl} \min & \sum_{i \in V} x_i \\ \text{s.t.} & x_i + x_j \geq 1 \quad \forall \{i,j\} \in E \\ & x_i \in \{0,1\} \quad \forall i \in V. \end{array}$$

Construct $\mathcal{F}$ by adding a face of the form $F = \{x \in [0,1]^n |\ x_i = 0,\ x_j = 0\}$ for each $\{i,j\} \in E$. Then the vertex cover problem, which is $\mathcal{NP}$-hard, reduces to optimization of a linear function over $\{0,1\}^n \setminus \cup_{F \in \mathcal{F}} \text{vert}(F)$. □

3.4 Applications

3.4.1 k-best solutions

The k-best problem defined below is closely related to removing vertices.

Definition 43. *Given a nonempty 0-1 polytope $P \subseteq \mathbb{R}^n$, a vector $c \in \mathbb{R}^n$, and a positive integer k, the k-best problem is to either assert $|\text{vert}(P)| \leq k$ and return $\text{vert}(P)$, or to return $v_1, \ldots, v_k \in \text{vert}(P)$, all distinct, such that $\max\{cv_i |\ i = 1, \ldots, k\} \leq \min\{cv |\ v \in \text{vert}(P) \setminus \{v_1, \ldots, v_k\}\}$.*

Since we can sequentially remove vertices from 0-1 polytopes, we can prove the following.

Proposition 44. *Let $P \subseteq [0,1]^n$ be a tractable 0-1 polytope. Then, for any $c \in \mathbb{R}^n$, the k-best problem can be solved in polynomial time on k and n.*

Proof. For each $i = 1, \ldots, k$, solve the problem

$$
\begin{aligned}
(\mathcal{P}_i)\ \min \quad & cx \\
\text{s.t.} \quad & x \in P_i,
\end{aligned}
$$

where $P_1 := P$, $P_i := \text{forb}(P_{i-1}, \{v_{i-1}\}) = \text{forb}(P, \{v_1, \ldots, v_{i-1}\})$ for $i = 2, \ldots, k$, and $v_i \in \text{vert}(P_i)$ is an optimal solution to $(\mathcal{P}_i)$, if one exists, for $i = 1, \ldots, k$. From Theorem 38, we can solve each of these problems in polynomial time. In particular, if $(\mathcal{P}_i)$ is infeasible, we return $v_1, \ldots, v_{i-1}$. Otherwise, by construction, $v_1, \ldots, v_k$ satisfy the required properties. Clearly, the construction is done in polynomial time. □

The above complexity result was originally obtained in [38] building on ideas from [47] by applying a branch-and-fix scheme.

3.4.2 Binary all-different polytopes

With edge-coloring of graphs in mind, the binary all-different polytope has been introduced in [39]. It was furthermore studied in [42] and [41]. We consider a more general setting.

Definition 45. *Given a positive integer k, nonempty 0-1 polytopes $P_1, \ldots, P_k$ in $\mathbb{R}^n$, and vectors $c_1, \ldots, c_k \in \mathbb{R}^n$, the binary all-different problem is to solve*

$$
\begin{aligned}
(\mathcal{P})\ \min \quad & \textstyle\sum_{i=1}^k c_i x_i \\
\text{s.t.} \quad & x_i \in \text{vert}(P_i) \quad i = 1, \ldots, k \\
& x_i \neq x_j \qquad 1 \leq i < j \leq k.
\end{aligned}
$$

In [39], it was asked whether the above problem is polynomially solvable in the case $P_i = [0,1]^n$ for all $i = 1, \ldots, k$. Using the tractability of the k-best problem, we give a positive answer even for the general case of distinct polytopes.

Given a graph $G = (V, E)$ and $U \subseteq V$, a U-matching in G is a matching $M \subseteq E$ such that each vertex in U is contained in some element of M.

Theorem 46. *If $P_i \subseteq \mathbb{R}^n$ is a tractable nonempty 0-1 polytope for $i = 1, \ldots, k$, then the binary all-different problem is polynomially solvable.*

Proof. For each $i = 1, \ldots, k$, let S_i be the solution set of the k-best problem (Def. 43) for P_i and c_i. Observe that $|S_i| \leq k$. Now, consider the bipartite graph $G = (S \cup R, E)$, where $S := \cup_{i=1}^{k} S_i$ and $R := \{1, \ldots, k\}$. For each $v \in S$ and $i \in R$, we include the arc $\{v, i\}$ in E if and only if $v \in S_i$. Finally, for each $\{v, i\} \in E$, we set $w_{vi} := c_i v$.

We claim that $(\mathcal{P})$ reduces to finding an R-matching in G of minimum weight with respect to w. It is straightforward to verify that an R-matching in G defines a feasible solution to $(\mathcal{P})$ of equal value. Thus, it is enough to show that if $(\mathcal{P})$ is feasible, then there exists an R-matching with the same optimal value. Indeed, let $(x_1, \ldots, x_k)$ be an optimal solution to $(\mathcal{P})$ that does not define an R-matching, that is, such that $x_i \notin S_i$ for some $i = 1, \ldots, k$. Then, we must have $|\text{vert}(P_i)| > k$ and $|S_i| = k$. This latter condition and $x_i \notin S_i$ imply the existence of $v \in S_i$ such that $v \neq x_j$ for all $j = 1, \ldots, k$. Furthermore, by the definition of S_i, we also have $c_i v \leq c_i x_i$. Therefore, the vector $(x_1, \ldots, x_{i-1}, v, x_{i+1}, \ldots, x_k)$ is an optimal solution to $(\mathcal{P})$ having its i-th subvector in S_i. Iteratively applying the above reasoning to all components, we obtain an optimal solution to $(\mathcal{P})$ given by an R-matching as desired. □

3.5 *Extension to integral polytopes*

In this section, we generalize the forbidden-vertices problem to integral polytopes, that is, to polytopes having integral extreme points, even allowing the removal of points that are not vertices. We show that for an important class of integral polytopes the resulting problem is tractable.

For an integral polytope $P \subseteq \mathbb{R}^n$ and $X \subseteq P \cap \mathbb{Z}^n$, we define $\text{forb}_I(P, X) := \text{conv}((P \cap \mathbb{Z}^n) \setminus X)$.

Definition 47. *Given an integral polytope $P \subseteq \mathbb{R}^n$, a set $X \subseteq P \cap \mathbb{Z}^n$ of integral vectors, and a vector $c \in \mathbb{R}^n$, the forbidden-vectors problem asks to either assert $(P \cap \mathbb{Z}^n) \setminus X = \varnothing$, or to return a minimizer of cx over $(P \cap \mathbb{Z}^n) \setminus X$ otherwise.*

Given vectors $l, u \in \mathbb{R}^n$ with $l \leq u$, we denote $[l, u] := \{x \in \mathbb{R}^n | \ l_i \leq x_i \leq u_i, i = 1, \ldots, n\}$. We term these sets as boxes.

Definition 48. *An integral polytope $P \subseteq \mathbb{R}^n$ is box-integral if for any pair of vectors $l, u \in \mathbb{Z}^n$ with $l \leq u$, the polytope $P \cap [l, u]$ is integral.*

Polytopes defined by a TU matrix and an integral right-hand-side, or by a box-TDI system, are examples of box-integral polytopes. Further note that if P is tractable and box-integral, so is $P \cap [l, u]$. When both conditions are met, we say that P is box-tractable.

With arguments analogous to that of the 0-1 case, we can verify the following result.

Theorem 49. *If $P \subseteq \mathbb{R}^n$ is a box-tractable polytope, then, given a list $X \subseteq P \cap \mathbb{Z}^n$, the forbidden-vectors problem is polynomially solvable. Moreover,*

$$\text{xc}(\text{forb}_I(P, X)) \leq 2n|X|(\text{xc}(P) + 1).$$

Proof. Since P is bounded, it is contained in a box. Without lost of generality and to simplify the exposition, we may assume that $P \subseteq [0, r-1]^n$ for some $r \geq 2$. As in the 0-1 case, we first address the case $P = [0, r-1]^n$, for which we provide two extended formulations for $\text{forb}_I(P, X)$ involving $\mathcal{O}(n|X|)$ variables and constraints.

The first extended formulation relies on the mapping $\phi(x) := \sum_{i=1}^{n} r^{i-1}x_i$ for $x \in [0, r-1]^n$, which defines a bijection with $\{0, \ldots, r^n - 1\}$. Letting $K_r(a,b) := \{x \in \{0,\ldots,r-1\}^n |\ a \leq \phi(x) \leq b\}$, we have that $\text{forb}_I(P,X)$ is the convex hull of the union of at most $|X| + 1$ sets of the form $K_r(a,b)$. Since $\text{conv}(K_r(a,b))$ has $\mathcal{O}(n)$ facets [26], by disjunctive programming we obtain an extended formulation for $\text{forb}_I(P,X)$ having $\mathcal{O}(n|X|)$ inequalities.

For the second extended formulation, let X' denote the projection of X onto the first $n-1$ coordinates and set $\widehat{X} := (X' \times \{0,\ldots,r-1\}) \setminus X$. Along the lines of Proposition 35, we have

$$\{0,\ldots,r-1\}^n \setminus X = \left[\left(\{0,\ldots,r-1\}^{n-1} \setminus X'\right) \times \{0,\ldots,r-1\}\right] \cup \widehat{X}.$$

Although $\widehat{X}$ can have up to $r|X|$ elements, we also see that $\widehat{X}$ is the union of at most $2|X|$ sets of the form $v \times \{\alpha,\ldots,\beta\}$ for $v \in X'$ and integers $0 \leq \alpha \leq \beta \leq r-1$. More precisely, for each $v \in X'$, there exist integers $0 \leq \alpha_1^v \leq \beta_1^v < \alpha_2^v \leq \beta_2^v < \cdots < \alpha_{q_v}^v \leq \beta_{q_v}^v \leq r-1$ such that

$$\widehat{X} = \bigcup_{v \in X'} \bigcup_{l=1}^{q_v} v \times \{\alpha_l^v, \ldots, \beta_l^v\}$$

and $\sum_{v \in X'} q_v \leq 2|X|$. Therefore, $\text{conv}(\widehat{X})$ can be described with $\mathcal{O}(|X|)$ inequalities. Then a recursive construction of an extended formulation for $\text{forb}_I(P,X)$ is analogous to the binary case and involves $\mathcal{O}(n|X|)$ variables and constraints.

In order to address the general case, we first show how to cover $\{0,\ldots,r-1\}^n \setminus X$ with boxes. For each $i = 1,\ldots,n$, let X^i be the projection of X onto the first i components and let $\widehat{X}^i := (X^{i-1} \times \{0,\ldots,r-1\}) \setminus X^i$, where $\widehat{X}^1 := \{0,\ldots,r-1\} \setminus X^1$. Working the recursion backwards yields

$$\{0,\ldots,r-1\}^n \setminus X = \bigcup_{i=1}^{n} \left[\widehat{X}^i \times \{0,\ldots,r-1\}^{n-i}\right].$$

Combining the last two expressions, we arrive at

$$\{0,\ldots,r-1\}^n \setminus X = \bigcup_{i=1}^{n} \bigcup_{v \in X^{i-1}} \bigcup_{l=1}^{q_v} v \times \{\alpha_l^v, \ldots, \beta_l^v\} \times \{0,\ldots,r-1\}^{n-i}.$$

The right-hand-side defines a family $\mathcal{B}$ of at most $2n|X|$ boxes in $\mathbb{R}^n$, yielding

$$\{0,\ldots,r-1\}^n \setminus X = \bigcup_{[l,u]\in\mathcal{B}} [l,u] \cap \mathbb{Z}^n.$$

Finally, if $P \subseteq [0,r-1]^n$, then

$$(P\cap\mathbb{Z}^n)\setminus X = (P\cap\mathbb{Z}^n)\cap(\{0,\ldots,r-1\}^n\setminus X) = \bigcup_{[l,u]\in\mathcal{B}} P\cap[l,u]\cap\mathbb{Z}^n.$$

Moreover, if P is box-tractable, then

$$\mathrm{forb}_I(P,X) = \mathrm{conv}\left(\bigcup_{[l,u]\in\mathcal{B}} \mathrm{conv}\left(P\cap[l,u]\cap\mathbb{Z}^n\right)\right) = \mathrm{conv}\left(\bigcup_{[l,u]\in\mathcal{B}} P\cap[l,u]\right),$$

where each term within the union is a tractable set. □

The k-best problem and the binary all-different problem can be extended to the case of integral vectors as follows.

Definition 50. *Given a nonempty integral polytope $P \subseteq \mathbb{R}^n$, a vector $c \in \mathbb{R}^n$, and a positive integer k, the integral k-best problem is to either assert $|P\cap\mathbb{Z}^n| \leq k$ and return $P\cap\mathbb{Z}^n$, or to return $v_1,\ldots,v_k \in P\cap\mathbb{Z}^n$, all distinct, such that* $\max\{cv_i|\ i=1,\ldots,k\} \leq \min\{cv|\ v\in\ (P\cap\mathbb{Z}^n)\setminus\{v_1,\ldots,v_k\}\}$.

Definition 51. *Given a positive integer k, nonempty integral polytopes $P_1,\ldots,P_k$ in $\mathbb{R}^n$, and vectors $c_1,\ldots,c_k \in \mathbb{R}^n$, the integral all-different problem is to solve*

$$\begin{array}{lll} (\mathcal{P})\ \min & \sum_{i=1}^k c_i x_i & \\ \text{s.t.} & x_i \in P\cap\mathbb{Z}^n & i=1,\ldots,k \\ & x_i \neq x_j & 1\leq i<j\leq k. \end{array}$$

The above problems can be shown to be polynomially solvable if the underlying polytopes are box-tractable.

3.6 Concluding remarks

Having shown the tractability of the binary all-different problem, it is natural to ask whether we can find a complete linear description for the case $P_i = [0,1]^n$ for all i. In a mixed-integer program, all-different constraints can be formulated by including additional variables and linear constraints to enforce $x_i \neq x_j$. However, with the exception of $k = 2$, this formulation is not ideal in the sense that its linear relaxation does not yield the convex hull of the feasible set. In practice, it is crucial to have tight formulations to improve the performance of branch-and-cut solvers, and usually cutting-planes are generated and added on-the-fly. Although some families of facet-defining inequalities are presented in [39], [41] and [42], finding a complete linear description remains an open question.

CHAPTER IV

IMPROVING THE INTEGER L-SHAPED METHOD

4.1 *Introduction*

In this chapter we consider mixed-integer programs of the form

$$
\begin{aligned}
\text{(IP)} \quad \min_{x,z,\theta} \quad & cx + dz + \theta \\
\text{s.t.} \quad & Ax + Cz \leq b && (36)\\
& Q(x) - \theta \leq 0 && (37)\\
& x \in \{0,1\}^n && (38)\\
& z \geq 0,\ z \in Z, && (39)
\end{aligned}
$$

where Z is a mixed-integer set and $Q(x)$ is a real-valued function taking a binary vector x as argument. We say that (x^*, z^*, θ^*) is a candidate solution if (x^*, z^*) satisfies (36), (38), and (39). If in addition (37) holds, then we say (x^*, z^*, θ^*) is a feasible (candidate) solution. Constraint (37) together with the presence of θ in the objective function ensures $\theta = Q(x)$ is satisfied by any optimal solution to (IP). A fundamental assumption is that given x, $Q(x)$ can be computed with a reasonable amount of effort.

In the context of two-stage stochastic integer programming, we usually have

$$Q(x) := \mathbb{E}_\xi \left[\min_y \{qy :\ Wy = h - Tx,\ y \in Y\} \right],$$

which denotes the expected second-stage cost of x with respect to the random data $\xi = (q, W, T, h)$. We assume that Y imposes some integrality requirements on y. When ξ has a finite set of possible outcomes, we have $Q(x) = \sum_\xi p_\xi Q_\xi(x)$, where $Q_\xi(x)$ denotes the optimal second-stage value of the scenario associated to ξ, and

p_ξ is the probability of occurrence of ξ. Thus, (IP) can be cast as a large-scale mixed integer program. When the burden of solving (IP) is mainly due to the presence of a large number of scenarios, schemes similar to Benders' decomposition [7] and the L-shaped method [59] can be effective. The idea is to relax (37) and consider θ as an underestimator of $Q(x)$, and successively add cuts in the (x, θ)-space to better approximate the shape of $Q(x)$. This is done until an optimal solution (x^*, z^*, θ^*) satisfying $\theta^* = Q(x^*)$ is found. When the second-stage problem is a linear program, $Q(x)$ is convex in x and thus can be approximated by subgradients using optimal dual solutions. In contrast, when the second-stage problem is a mixed-integer program, such a nice property does not hold, and moreover, $Q(x)$ can even be discontinuous. Thus, the decomposition approaches of the linear case have to be modified to accommodate integer variables in the second stage. In [34], such a modification, the integer L-shaped method, is introduced. It is designed for two-stage stochastic integer problems having binary first-stage variables as it exploits the facial property of 0-1 sets. More generally, the integer L-shaped method can be applied to any mixed-integer problem having the form of (IP) as long as $Q(x)$ is computable from binary x. In particular, it also fits situations where $Q(x)$ can be evaluated with a closed-form analytical formula, but it does not have an amenable mixed-integer formulation. Applications of this method include vehicle routing [37], [23], probabilistic traveling salesman problems [35], location problems[36], and generalized assignment [1], among others.

Next we describe the integer L-shaped method. Let X be the projection of the feasible region of (IP) onto the x-space, and let $L \in \mathbb{R}$ be a lower bound on $Q(x)$ over X. Then (IP) can be equivalently formulated as

$$
\begin{aligned}
\text{(MP)}\quad \min \quad & cx + dz + \theta \\
\text{s.t.} \quad & Ax + Cz \leq b \\
& \Pi x - \mathbf{1}\theta \leq \pi_0 \qquad (40) \\
& x \in \{0,1\}^n \\
& z \geq 0,\ z \in Z \\
& \theta \geq L,
\end{aligned}
$$

where **1** denotes a vector of ones of appropriate size, as long as for each $x^* \in X$ constraints (40) include a cut of the form $\pi^k x - \theta \leq \pi_0^k$ such that $\pi^k x - \pi_0^k \leq Q(x)$ for all $x \in X$ and $\pi^k x^* - \pi_0^k = Q(x^*)$. In other words, the affine function $\pi^k x - \pi_0^k$ underestimates $Q(x)$ on X, and the estimate is tight at x^*. The optimality cuts of Laporte and Louveaux [34] define such a cut family and form the basis of the integer L-shaped method.

Given $x^* \in \{0,1\}^n$, let $S(x^*) := \{i : x_i^* = 1\}$. In [34], the (standard) integer optimality cut at x^* is defined as

$$\theta \geq (Q(x^*) - L)\left(\sum_{i \in S(x^*)} x_i - \sum_{i \notin S(x^*)} x_i - |S(x^*)|\right) + Q(x^*). \qquad (41)$$

Let $\Delta_{x^*}(x) := |S(x^*)| - \sum_{i \in S(x^*)} x_i + \sum_{i \notin S(x^*)} x_i$ be the Hamming distance between x and x^*, and note that $0 \leq \Delta_{x^*}(x) \leq n$ with $\Delta_{x^*}(x) = 0$ if and only if $x = x^*$. Thus, if $x = x^*$, then the right-hand side of (41) attains its maximum value $Q(x^*)$. If $x \in \{0,1\}^n \setminus \{x^*\}$, then it takes a value less than or equal to L. Since $\theta \geq L$, combining both cases, we have that (41) models the implication $x = x^* \Rightarrow \theta \geq Q(x)$. Observe also that as we have one cut per element in X, (40) might have exponentially many constraints. Thus, (40) is omitted from the initial formulation (MP) and cuts (41) are added on-the-fly as new solutions are discovered.

It is important to keep in mind that given the enumerative nature of (41), in practice these cuts are complemented with other inequalities that, albeit not tight, help to improve the global lower bound on $Q(x)$. When $Q(x)$ is the expected second-stage value of x given by the value function of a mixed-integer program, the most obvious inequalities to add are the subgradient cuts given by the continuous relaxation $Q_{LP}(x)$ of $Q(x)$. They have the form

$$\theta \geq s(x - x^*) + Q_{LP}(x^*), \tag{42}$$

where s is a subgradient of $Q_{LP}(x)$ at x^*.

An implementation of the integer L-shaped method with a current state-of-the-art solver works as follows. Having computed a lower bound L on $Q(x)$ and solved the continuous relaxation of (IP) with Benders' decomposition, we end up with a linear master problem that includes subgradient cuts of the form (42). Then we reinforce the binary requirements on x and any integrality restrictions on z, leading to a mixed-integer master problem of the form (MP), but where the system (40) is a relaxation of (IP), so that an optimal solution to the current problem may not be feasible to (IP). The idea now is to solve the mixed-integer master problem in a way such that all integer solutions are checked against feasibility with respect to (IP) before being accepted as an incumbent. For this, the solver proceeds in a similar fashion to branch-and-cut, that is, it generates a search tree by solving linear subproblems, branching, and adding cutting planes. The main difference is that when a candidate integer solution (x^*, z^*, θ^*) satisfying (36), (38) and (39) is found at a node of the search tree, an additional routine, the so-called optimality cut function, is called in order to assert feasibility and add optimality cuts. If the solution is infeasible to the true problem (IP), i.e., $\theta^* < Q(x^*)$, this function generates an optimality cut that is applied to all pending nodes in the master problem tree, ensuring that this solution is discarded. Then the solver continues exploring the tree with the guarantee that any discarded, and thus infeasible, solution will not appear again. If the solution is actually feasible to (IP), then it is accepted by the optimality cut function and the current incumbent is updated accordingly. A modern implementation of the (standard) integer L-shaped method is presented in Algorithm 1 below.

Algorithm 1 Integer L-shaped method

```
Input: $A, C, b, c, d, Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}$
Output: Optimal solution $x^*$ to (IP) and optimal value
1: Compute a lower bound $L$ of $Q(x)$
2: Solve the LP relaxation of (IP) with Benders' decomposition
3: Declare $x$ variables as binary in master problem
4: Initialize the optimality cut function
5: Solve the integer master problem using the optimality cut function to assert
   feasibility of solutions and add optimality cuts
6: return $x^*$ and optimal value
```

In line 4 of Algorithm 1 we initialize any additional structures that may be needed by the optimality cut function before invoking the solver in line 5. In particular, as there may be several solutions sharing the same x subvector, we keep a list V of first-stage x for which $Q(x)$ has been computed to avoid duplicate evaluations. In a standard implementation, the optimality cut function has the form shown in Algorithm 2

The optimality cut function returns `TRUE` if the candidate integer solution is indeed feasible to (IP). Otherwise it returns `FALSE` to reject the solution and apply the optimality cut. Note that the steps in lines 4 and 5 in Algorithm 2 are not needed for convergence of the method, but help to improve the global lower bound on $Q(x)$.

The optimality cut (41) relies on exact evaluations of $Q(x)$, which can be very time-consuming in the case where $Q(x)$ is given by a complicated mixed-integer program. Also, observe that (41) depends on x^* and $Q(x^*)$ only, i.e., it only depends on the point to be cut-off. In particular, it does not take into account the information provided by other solutions that we may have found while exploring

Algorithm 2 Standard optimality cut function

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}, V$
Output: true if solution is feasible, false otherwise
 1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
 2:     return true
 3: end if
 4: Compute $Q_{LP}(x^*)$
 5: Add the subgradient cut (42)
 6: Compute $Q(x^*)$
 7: $V \leftarrow V \cup \{x^*\}$
 8: if $\theta^* < Q(x^*)$ then
 9:     Add the integer optimality cut (41)
10:     return false
11: else
12:     return true
13: end if
```

the first-stage set. To improve the performance of the integer L-shaped method, we propose two approaches to deal with the above issues. First, in Section 4.2, we present a simple modification that alternates between exact and approximate evaluations of $Q(x)$. Then, in Section 4.3, we introduce of a new type of optimality cut that includes information obtained from different solutions; in particular, evaluations and estimates of $Q(x)$ at different points. These new cuts are obtained through a cut-generating linear program which is constructed based on ideas from disjunctive programming and the forbidden-vertices problem from Chapter 3. Then, in Section 4.4, we outline an implementation that combines both modifications in a single method. Finally, in Section 4.5, we present computational results of the proposed variants on two classes of stochastic integer programs.

4.2 Alternating cuts

In this section we present a simple cut strategy to decrease the overall effort incurred in computing the function $Q(x)$.

Suppose that while solving (IP) with the integer L-shaped method, a candidate solution (x^*, z^*, θ^*) has been found along the search tree of (MP). Recall that we reject the solution if $\theta^* < Q(x^*)$. Since $Q_{LP}(x) \leq Q(x)$, a sufficient condition to reject (x^*, z^*, θ^*) is $\theta^* < Q_{LP}(x^*)$. Given that $Q_{LP}(x)$ is convex, we have that the subgradient cut (42) is a valid inequality that cuts off the pair (x^*, θ^*) in the (x, θ)-space. Therefore, instead of evaluating $Q(x^*)$ exactly, we first evaluate $Q_{LP}(x^*)$ and check whether $\theta^* < Q_{LP}(x^*)$. If so, we add the subgradient cut (42) to remove (x^*, θ^*). Otherwise, we compute $Q(x^*)$ and check whether $\theta^* < Q(x^*)$. If so, we add the integer optimality cut (41). Otherwise, we accept the solution. The key idea is to use $Q_{LP}(x)$ as a proxy for $Q(x)$ to check feasibility of a candidate solution, preventing unnecessary, and more costly, computations of $Q(x)$.

The modification just described is similar in spirit to sequential approximation schemes such as [56], [22], [33], and [52], where the second-stage cost function $Q(x)$ is approximated by linear programs which, starting with $Q_{LP}(x)$, are iteratively strengthened with additional cuts. Although these methods are shown to converge after a finite number of steps, the convergence can be very slow and in practice exact evaluations of $Q(x)$ may be required. In contrast, in the context of the integer L-shaped method, we propose to use $Q_{LP}(x)$ as the unique intermediate approximation for $Q(x)$, which is a simple yet useful modification whose implementation is rather straightforward and, to the best of our knowledge, has not been reported in the literature.

To implement the approach presented above, in addition to V, we also keep a list V_{LP} of visited first-stage solutions x for which the continuous relaxation $Q_{LP}(x)$ has been computed. The modified strategy, which we call alternating cuts, proceeds as shown in Algorithm 3.

Note that if $x^* \notin V_{LP}$ satisfies (42), then x^* is included into V_{LP} and thus the steps in lines 12–19 of Algorithm 3 are applied to check whether (x^*, z^*, θ^*) is a feasible solution or not. As we shall see in Section 4.5, this simple modification yields speedups of one order of magnitude on instances from the literature.

Algorithm 3 Optimality cut function with alternating cut strategy

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}, V, V_{LP}$
Output: true if solution is feasible, false otherwise
 1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
 2:     return true
 3: end if
 4: if $x^* \notin V_{LP}$ then
 5:     Compute $Q_{LP}(x^*)$
 6:     $V_{LP} \leftarrow V_{LP} \cup \{x^*\}$
 7:     if $\theta^* < Q_{LP}(x^*)$ then
 8:         Add the subgradient cut (42).
 9:         return false
10:     end if
11: end if
    // Now we have $x^* \in V_{LP}$ and $\theta^* \geq Q_{LP}(x^*)$
12: Compute $Q(x^*)$.
13: $V \leftarrow V \cup \{x^*\}$
14: if $\theta^* < Q(x^*)$ then
15:     Add the integer optimality cut (41)
16:     return false
17: else
18:     return true
19: end if
```

4.3 New optimality cuts

In this section, we present a new class of integer optimality cuts that can be used as an alternative to the standard cut (41). After providing an overview of the approach, we show how to construct a cut-generating linear program to separate these new inequalities and then we discuss some implementation details.

Let S be the projection of the feasible set of (MP) onto the (x, θ)-space, which corresponds to the epigraph of $Q(x)$ over X, i.e.,

$$S = \{(x, \theta) \in X \times \mathbb{R} : \; \theta \geq Q(x)\}.$$

Let $V \subseteq X$ be such that $Q(x)$ is known for all $x \in V$. We have

$$S \subseteq S(X, V) := \bigcup_{x \in V} \{(x, \theta) : \; \theta \geq Q(x)\} \cup (X \setminus V) \times \{\theta : \; \theta \geq L\}.$$

In some sense, $S(X, V)$ is the best approximation of S when only the values of $Q(x)$ for $x \in V$ are known and only the trivial lower bound L is available over $X \setminus V$. We consider the relaxation $S(V)$ of $S(X, V)$ given by

$$S(X,V) \subseteq S(V) := \bigcup_{x \in V} \{(x, \theta) : \ \theta \geq Q(x)\} \cup (\{0,1\}^n \setminus V) \times \{\theta : \ \theta \geq L\}.$$

Observe that $S(V) \subseteq S(U)$ for any $U \subseteq V$, and in particular, $S(V) \subseteq S(\{x\})$ for $x \in V$. Moreover, $S(V) = \bigcap_{x \in V} S(\{x\})$. Since (41) is a valid inequality for $\operatorname{conv}(S(\{x\}))$, it is also valid for $\operatorname{conv}(S(V))$. Actually, (41) is the only nontrivial cut needed to describe $\operatorname{conv}(S(\{x\}))$. However, in general, $\operatorname{conv}(S(V)) \subseteq \bigcap_{x \in V} \operatorname{conv}(S(\{x\}))$ holds with strict containment, i.e., adding (41) for all $x \in V$ does not yield $\operatorname{conv}(S(V))$. Our goal is to derive a compact extended formulation for $\operatorname{conv}(S(V))$ and use it to generate optimality cuts for a point (x^*, θ^*) in the (x, θ)-space that take into account the values of $Q(x)$ for $x \in V$.

Several steps of the construction of our cut-generating linear program rely on Lemma 52 below, which follows from disjunctive programming [5] applied in the context of linear extended formulations of polyhedra.

Lemma 52. *Let $P_1, \ldots, P_k$ be nonempty polyhedra in $\mathbb{R}^n$ having the same recession cone. If $P_i = \{x \in \mathbb{R}^n | \ \exists y_i \in \mathbb{R}^{m_i} : \ E_i x + F_i y_i \geq h_i\}$, then conv $\left(\cup_{i=1}^k P_i\right) = \{x \in \mathbb{R}^n | \ \exists x_i \in \mathbb{R}^n, \ y_i \in \mathbb{R}^{m_i}, \ \lambda \in \mathbb{R}^k : \ x = \sum_{i=1}^k x_i, \ E_i x_i + F_i y_i \geq \lambda_i h_i, \ \sum_{i=1}^k \lambda_i = 1, \ \lambda \geq 0\}$.*

4.3.1 Construction of CGLP

Clearly, we have $\operatorname{conv}(S(V)) = \operatorname{conv}\left(P_Q(V) \cup P_L(V)\right)$, where

$$P_Q(V) := \operatorname{conv}\left(\bigcup_{x \in V} \{(x, \theta) : \ \theta \geq Q(x)\}\right)$$

and

$$P_L(V) := \operatorname{conv}(\{0,1\}^n \setminus V) \times \{\theta : \ \theta \geq L\}.$$

Thus to describe $\operatorname{conv}(S(V))$ it suffices to provide compact extended formulations for $P_Q(V)$ and $P_L(V)$ and then apply disjunctive programming to their union.

Describing $P_Q(V)$ is trivial: letting $V = \{x^1, \dots, x^t\}$, then $P_Q(V)$ is the set of vectors $(x^Q, \theta^Q) \in \mathbb{R}^n \times \mathbb{R}$ for which there exists $\phi \in \mathbb{R}^t$ satisfying

$$\begin{aligned}
&-x^Q + \sum_{s=1}^{t} \phi_s x^s = 0 \\
&-\theta^Q + \sum_{s=1}^{t} \phi_s Q(x^s) \leq 0 \\
&\sum_{s=1}^{t} \phi_s = 1 \\
&\phi \geq 0.
\end{aligned}$$

To describe $P_L(V)$, it is enough to describe conv $(\{0,1\}^n \setminus V)$ and then take the Cartesian product with $\{\theta : \theta \geq L\}$. We build on results from the forbidden-vertices problem in Chapter 3 to do this.

Let V^i be the projection of V onto the first i coordinates. Define $\hat{V}^1 := \{0,1\} \setminus V^1$, $\hat{V}^i := [V^{i-1} \times \{0,1\}] \setminus V^i \subseteq \{0,1\}^i$ for $i \geq 2$, and write $\hat{V}^i = \{v^i_1, \dots, v^i_{k_i}\}$. Finally, for all i, let $W^{ij} := \hat{V}^i \times \{0\}^{j-i} = \{w^{ij}_1, \dots, w^{ij}_{k_i}\} \subseteq \{0,1\}^j$ for all $j \geq i$ and define $W^i := W^{in} = \{w^i_1, \dots, w^i_{k_i}\} \subseteq \{0,1\}^n$.

From Proposition 35 in Chapter 3, for all $1 \leq j \leq n-1$ we have

$$\{0,1\}^{j+1} \setminus X^{j+1} = \left[\left(\{0,1\}^j \setminus X^j \right) \times \{0,1\} \right] \cup \hat{X}^{j+1}. \tag{43}$$

The idea behind (43) is that any vector in $\{0,1\}^{j+1} \setminus X^{j+1}$ is such that either its projection onto $\{0,1\}^j$ does not lie in V^j or it is obtained by flipping the value of the last component of a vector in V^{j+1} otherwise.

We use the recursion (43) to derive an explicit linear extended formulation for conv $(\{0,1\}^n \setminus V)$ having $\mathcal{O}(n|V|)$ variables and constraints.

Proposition 53. *For all $2 \leq j \leq n$, conv $\left(\{0,1\}^j \setminus V^j\right)$ is given by all $x \in \mathbb{R}^j$ for which* *there exist vectors y, λ, and μ satisfying*

$$\begin{aligned}
&-x + y + \sum_{i=1}^{j} \sum_{l=1}^{k_i} \mu_l^i w_l^{ij} = 0 \\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
&\sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \le i \le j-1 \\
&\sum_{l=1}^{k_j} \mu_l^j + \lambda_{j-1} = 1 \\
&y_1 = 0 \\
&y_i - \lambda_{i-1} \le 0 \quad \forall 2 \le i \le j \\
&y \ge 0,\ \lambda \ge 0,\ \mu \ge 0.
\end{aligned}$$

Proof. We apply induction on $2 \le j \le n$. The base case reduces to proving that $\operatorname{conv}\left(\{0,1\}^2 \setminus V^2\right)$ is given by

$$\begin{aligned}
&-x + y + \sum_{i=1}^{2} \sum_{l=1}^{k_i} \mu_l^i w_l^{i2} = 0 \\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
&\sum_{l=1}^{k_2} \mu_l^2 + \lambda_1 = 1 \\
&y_1 = 0 \\
&y_2 - \lambda_1 \le 0 \\
&y \ge 0,\ \lambda \ge 0,\ \mu \ge 0.
\end{aligned} \tag{44}$$

Indeed, from (43), we have

$$\{0,1\}^2 \setminus X^2 = \left[\left(\{0,1\}^1 \setminus X^1\right) \times \{0,1\}\right] \cup \hat{X}^2. \tag{45}$$

By definition, we have $W^{12} = \hat{V}^1 \times \{0\} = (\{0,1\} \setminus V^1) \times \{0\}$. Then observe that

$$\left(\{0,1\}^1 \setminus X^1\right) \times \{0,1\} = W^{12} + \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\},$$

and thus

$$\text{conv}\left(\left(\{0,1\}^1 \setminus X^1\right) \times \{0,1\}\right) = \text{conv}\left(W^{12}\right) + \left\{y \in \mathbb{R}^2 : \; y_1 = 0, \; 0 \leq y_2 \leq 1\right\}.$$

Writing $W^{12} = \{w_1^{12}, \dots, w_{k_1}^{12}\}$, then it follows that conv $\left((\{0,1\}^1 \setminus X^1) \times \{0,1\}\right)$ is given by $p \in \mathbb{R}^2$ such that

$$\begin{aligned} & -p + y + \sum_{l=1}^{k_1} \mu_l^1 w_l^{12} = 0 \\ & \sum_{l=1}^{k_1} \mu_l^1 = 1 \\ & y_1 = 0 \\ & y_2 \leq 1 \\ & y \geq 0, \; \mu^1 \geq 0. \end{aligned}$$

By definition, we also have $\hat{V}^2 = W^{22} = \{w_1^{22}, \dots, w_{k_2}^{22}\}$, and thus conv $\left(\hat{V}^2\right)$ is given by $q \in \mathbb{R}^2$ such that

$$\begin{aligned} & -q + \sum_{l=1}^{k_2} \mu_l^2 w_l^{22} = 0 \\ & \sum_{l=1}^{k_2} \mu_l^2 = 1 \\ & \mu^2 \geq 0. \end{aligned}$$

From (45), we apply Lemma 52 to the above polytopes: we introduce a multiplier $0 \leq \lambda_1 \leq 1$, we include the equation $x = p + q$, and we multiply the right-hand-side vectors of the first and second systems by λ_1 and $1 - \lambda_1$, respectively. After eliminating p and q, we immediately obtain the desired system (44) for conv $\left(\{0,1\}^2 \setminus \; V^2\right)$.

Now, assuming that the claim holds for some $2 \leq j \leq n-1$, we will prove that it also holds for $j+1$. Since conv $\left((\{0,1\}^j \setminus \; V^j) \times \{0,1\}\right) = $ conv $\left((\{0,1\}^j \setminus \; V^j)\right) \times [0,1]$, by the inductive hypothesis, we have that conv $\left((\{0,1\}^j \setminus \; V^j) \times \{0,1\}\right)$ is given by $p \in \mathbb{R}^{j+1}$ satisfying

$$\begin{aligned}
&-p + y + \sum_{i=1}^{j}\sum_{l=1}^{k_i} \mu_l^i w_l^{ij+1} = 0 \\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
&\sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \leq i \leq j-1 \\
&\sum_{l=1}^{k_j} \mu_l^j + \lambda_{j-1} = 1 \\
&y_1 = 0 \\
&y_i - \lambda_{i-1} \leq 0 \quad \forall 2 \leq i \leq j \\
&y_{j+1} \leq 1 \\
&y \geq 0,\ \lambda \geq 0,\ \mu \geq 0,
\end{aligned}$$

where we have appended a new variable $0 \leq y_{j+1} \leq 1$ and vectors w_l^{ij} have been extended to w_l^{ij+1} by appending another component with value 0.

We also have that conv $\left(\hat{V}^{j+1}\right)$ is given by $q \in \mathbb{R}^{j+1}$ satisfying

$$\begin{aligned}
&-q + \sum_{l=1}^{k_{j+1}} \mu_l^{j+1} w_l^{j+1j+1} = 0 \\
&\sum_{l=1}^{k_{j+1}} \mu_l^{j+1} = 1 \\
&\mu^{j+1} \geq 0.
\end{aligned}$$

From (43), it is enough to apply Lemma 52 to the above polytopes to find an extended formulation for conv $\left(\{0,1\}^{j+1} \setminus X^{j+1}\right)$. Analogously to the base case, we introduce a multiplier $0 \leq \lambda_j \leq 1$, we include the equation $x = p + q$, and we multiply the right-hand-side vectors of the first and second systems by λ_j and $1 - \lambda_j$, respectively. After eliminating p and q, we immediately obtain the desired system for conv $\left(\{0,1\}^{j+1} \setminus\ V^{j+1}\right)$. □

From Proposition 53, we obtain that conv $(\{0,1\}^n \setminus V)$ is given by the vectors $x^L \in \mathbb{R}^n$ such that

$$\begin{aligned}
&-x^L + y + \sum_{i=1}^{n}\sum_{l=1}^{k_i} \mu_l^i w_l^i = 0\\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0\\
&\sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \le i \le n-1\\
&\sum_{l=1}^{k_n} \mu_l^n + \lambda_{n-1} = 1\\
&y_1 = 0\\
&y_i - \lambda_{i-1} \le 0 \quad \forall 2 \le i \le n\\
&y \ge 0,\ \lambda \ge 0,\ \mu \ge 0.
\end{aligned}$$

Appending the variable θ^L and the constraint $\theta^L \ge L$ to the above system gives an extended formulation for $P_L(V)$. Note that excluding the nonnegativity restrictions, the constraint matrix has $3n$ rows and $3n + \mathcal{O}(n|V|)$ columns, i.e, only its width changes with V. In particular, updating the formulation can be done columnwise, which is a desirable property from the computational point of view.

Once again, we apply disjunctive programming, but this time to $P_L(V)$ and $P_Q(V)$ to derive an extended formulation for conv $(S(V))$. Note that both $P_L(V)$ and $P_Q(V)$ have $\{(0,\theta) \in \mathbb{R}^n \times \mathbb{R} : \theta \ge 0\}$ as their recession cone and thus Lemma 52 applies. We introduce a multiplier $0 \le \delta \le 1$, we include the equations $x = x^L + x^Q$ and $\theta = \theta^L + \theta^Q$, and we multiply the right-hand-side vectors of the systems defining $P_L(V)$ and $P_Q(V)$ by δ and $1-\delta$, respectively.

Recall that in the definition of $S(V)$ we have dropped the dependence on X. To recover part of that information, we can describe a polyhedron that lies between conv (S) and conv $(S(V))$. For that, $P_L(V)$ can be coupled with any valid inequality for (MP). In particular, including variables $z \ge 0$ and the system $Ax^L + Cz \le b$

tightens the formulation. Lower bounds of the form $\Pi x^L - \mathbf{1}\theta^L \leq \pi_0$ can be useful too to better approximate the shape of the epigraph S of $Q(x)$. Thus we may assume that both types of constraints are added to the formulation of $P_L(V)$, and that $\theta^L \geq L$ is absorbed in $\Pi x^L - \mathbf{1}\theta^L \leq \pi_0$.

Finally, we obtain that if (x^*, θ^*) does not belong to conv $(S(V))$, and thus not to conv (S), then the following system is infeasible:

$$
\begin{aligned}
(\alpha) \quad & x^L + x^Q = x^* \\
(\beta) \quad & \theta^L + \theta^Q = \theta^* \\
(\sigma) \quad & -x^L + y + \sum_{i=1}^{n} \sum_{l=1}^{k_i} \mu_l^i w_l^i = 0 \\
(\rho_1) \quad & \sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
(\rho_i) \quad & \sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \le i \le n-1 \\
(\rho_n) \quad & \sum_{l=1}^{k_n} \mu_l^n + \lambda_{n-1} - \delta = 0 \\
(\varphi_1) \quad & y_1 = 0 \\
(\varphi_i) \quad & y_i - \lambda_{i-1} \le 0 \quad \forall 2 \le i \le n \\
(\psi) \quad & \Pi x^L - \mathbf{1}\theta^L - \pi_0 \delta \le 0 \\
(\nu) \quad & A x^L + C z - b\delta \le 0 \\
(\gamma) \quad & -x^Q + \sum_{s=1}^{t} \phi_s x^s = 0 \\
(\tau) \quad & -\theta^Q + \sum_{s=1}^{t} \phi_s Q(x_s) \le 0 \\
(\eta) \quad & \sum_{s=1}^{t} \phi_s + \delta = 1 \\
& y \ge 0,\ \lambda \ge 0,\ \mu \ge 0 \\
& \phi \ge 0 \\
& \delta \ge 0.
\end{aligned}
$$

By Farkas' Lemma, and after removing redundancies, we arrive at the alternative system

$$\begin{array}{rcll}
x^*\alpha + \theta^*\beta + \eta & < & 0 & \\
\alpha - \sigma + A^\top \nu + \Pi^\top \psi & = & 0 & \\
\beta - \mathbf{1}\psi & = & 0 & \\
-\rho_n + \eta - b\nu - \pi_0\psi & \geq & 0 & \\
C^\top \nu & \geq & 0 & \\
\sigma_i + \varphi_i & \geq & 0 & 2 \leq i \leq n \\
-\rho_i + \rho_{i+1} + \varphi_{i+1} & \geq & 0 & 1 \leq i \leq n-1 \\
w_l^i \sigma + \rho_i & \geq & 0 & 1 \leq n,\ 1 \leq l \leq k_i \\
x^s\alpha + Q(x^s)\beta + \eta & \geq & 0 & 1 \leq s \leq t \\
\beta \geq 0,\ \varphi \geq 0,\ \nu \geq 0,\ \psi \geq 0. & & &
\end{array}$$

Thus, any feasible solution to the above system yields an inequality $\alpha x + \beta\theta \geq -\eta$ that is valid for conv (S), but is violated by (x^*, θ^*).

For finite termination of the integer L-shaped method, we need a tightness condition at the current solution, namely $\alpha x^* + \beta Q(x^*) = -\eta$. Including this condition yields $0 > x^*\alpha + \theta^*\beta + \eta = x^*\alpha + \beta Q(x^*) + \eta - \beta Q(x^*) + \theta^*\beta = \beta(\theta^* - Q(x^*))$. Since $\theta^* < Q(x^*)$, we conclude that $\beta > 0$ in any feasible tight solution. Therefore, we replace the condition $x^*\alpha + \theta^*\beta + \eta < 0$ with $x^*\alpha + Q(x^*)\beta + \eta = 0$ and the normalization $\beta = 1$. Note that the objective function of the resulting linear program is fixed to zero, and we only need to find a feasible solution, which always exists by definition of the system; in particular, (41) is feasible. The final system, denoted CGLP, reads

$$
\begin{aligned}
\alpha - \sigma + A^\top \nu + \Pi^\top \psi &= 0 && \\
\mathbf{1}\psi &= 1 && \\
-\rho_n + \eta - b\nu - \pi_0 \psi &\geq 0 && \\
C^\top \nu &\geq 0 && \\
\sigma_i + \varphi_i &\geq 0 && 2 \leq i \leq n \\
-\rho_i + \rho_{i+1} + \varphi_{i+1} &\geq 0 && 1 \leq i \leq n-1 \\
w_l^i \sigma + \rho_i &\geq 0 && 1 \leq i \leq n,\ 1 \leq l \leq k_i \qquad (46) \\
x^s \alpha + Q(x^s) + \eta &\geq 0 && 1 \leq s < t \qquad (47) \\
x^t \alpha + Q(x^t) + \eta &= 0 && \qquad (48) \\
\varphi \geq 0,\ \nu \geq 0,\ \psi \geq 0. & &&
\end{aligned}
$$

Having set $x^t := x^*$, we solve CGLP to find a feasible solution to the system. In particular, we obtain α and η defining a CGLP-based optimality cut of the form

$$\alpha x + \theta \geq -\eta \qquad (49)$$

which by construction cuts off (x^*, θ^*) with $\theta^* < Q(x^*)$.

4.3.2 Implementation

The main difference that we are proposing with the standard implementation is the use of the CGLP-based cut (49) in place of (41). This requires keeping a list V of first-stage solutions for which $Q(x)$ has been computed and updating CGLP accordingly. Algorithm 4 shows the procedure.

A key step is found in line 7 of Algorithm 4 as conv $(S(V))$ has to be recomputed whenever a new vector x^* is added to V. Of course, we could derive CGLP from scratch every time. Doing so requires computing the sets W^i and thus creating $\mathcal{O}(n|V|)$ constraints in (46). Instead, we propose to perform marginal updates

Algorithm 4 Optimality cut function with CGLP-based optimality cuts

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}, V$
Output: true if solution is feasible, false otherwise
 1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
 2:     return true
 3: end if
 4: Compute $Q_{LP}(x^*)$
 5: Add the subgradient cut (42)
 6: Compute $Q(x^*)$
 7: Update CGLP.
 8: $V \leftarrow V \cup \{x^*\}$
 9: if $\theta^* < Q(x^*)$ then
10:     Solve CGLP to obtain $\alpha$ and $\eta$
11:     Add the integer optimality cut (49)
12:     return false
13: else
14:     return true
15: end if
```

from an iteration to the next one using the fact that $W^i = \hat{V}^i \times \{0\}^{n-i}$.

Let $V_t = \{x^1, \ldots, x^t\}$ be the set of the first t solutions found along the master tree. Similarly, let V_t^i be the projection of V_t onto the first i components and set $\hat{V}_t^i := [V_t^{i-1} \times \{0,1\}] \setminus V_t^i$ with $\hat{V}_t^1 := \{0,1\} \setminus V_t^1$. Suppose a new vector $x^{t+1} = (x_1, \ldots, x_n)$ is to be included and let V_{t+1}, V_{t+1}^i, $\hat{V}_{t+1}^i$ be the updated sets. Let $\bar{x}^i := (x_1, \ldots, x_{i-1}, x_i)$ and $\hat{x}^i := (x_1, \ldots, x_{i-1}, 1 - x_i)$. Clearly, we have $V_{t+1} = V_t \cup \{x^{t+1}\}$ and $V_{t+1}^i = V_t^i \cup \{\bar{x}^i\}$. Now, to obtain $\hat{V}_{t+1}^i$, observe that

$$\begin{aligned}
\hat{V}_{t+1}^i &= \left[V_{t+1}^{i-1} \times \{0,1\}\right] \setminus V_{t+1}^i \\
&= \left[V_t^{i-1} \times \{0,1\} \cup \{\hat{x}^i, \bar{x}^i\}\right] \setminus \left[V_t^i \cup \{\bar{x}^i\}\right] \\
&= \left[V_t^{i-1} \times \{0,1\} \cup \{\hat{x}^i\}\right] \setminus \left[V_t^i \cup \{\bar{x}^i\}\right] \\
&= \left(\left[V_t^{i-1} \times \{0,1\}\right] \setminus \left[V_t^i \cup \{\bar{x}^i\}\right]\right) \cup \left(\{\hat{x}^i\} \setminus \left[V_t^i \cup \{\bar{x}^i\}\right]\right) \\
&= \left(\hat{V}_t^i \setminus \{\bar{x}^i\}\right) \cup \left(\{\hat{x}^i\} \setminus \ V_t^i\right).
\end{aligned}$$

Therefore, if $\hat{x}^i \notin \hat{V}_t^i$ and $\hat{x}^i \notin V_t^i$, then $\hat{x}^i$ is included in $\hat{V}_{t+1}^i$. Also, if $\bar{x}^i \in \hat{V}_t^i$, then $\bar{x}^i$ is removed to obtain $\hat{V}_{t+1}^i$. Further observe that both operations cannot occur at the same iteration since the equivalence

$$\bar{x}^i \in \hat{V}_t^i \iff \hat{x}^i \in V_t^i \ \wedge \ \bar{x}^i \notin V_t^i$$

implies that $\hat{x}^i \notin V_t^i$ and $\bar{x}^i \in \hat{V}_t^i$ cannot hold true at the same time.

It follows that updating V involves adding or removing at most one vector for each W^i, totaling at most n such operations. The system CGLP is updated accordingly by appending or dropping at most n rows in (46). Also, x^{t+1} takes the place of x^t in (48) and the cut corresponding to x^t now takes the form (47) by changing the equality sign into inequality. The procedure to update CGLP is shown in Algorithm 5.

Algorithm 5 Updating CGLP

```
Input: CGLP, $V^i$, $\hat{V}^i$, $t$, $x^{t+1} = (x_1, \dots, x_n)$
Output: Updated CGLP, $V^i$, $\hat{V}^i$
 1: for $1 \le i \le n$ do
 2:     $\bar{x}^i \leftarrow (x_1, \dots, x_{i-1}, x_i)$
 3:     $\hat{x}^i \leftarrow (x_1, \dots, x_{i-1}, 1 - x_i)$
 4:     if $\hat{x}^i \notin \hat{V}^i$ and $\hat{x}^i \notin V^i$ then
 5:         $w \leftarrow \hat{x}^i \times \{0\}^{n-i}$
 6:         Add $w\sigma + \rho_i \ge 0$ to (46)
 7:         $\hat{V}^i \leftarrow \hat{V}^i \cup \{\hat{x}^i\}$
 8:     end if
 9:     if $\bar{x}^i \in \hat{V}^i$ then
10:         $w \leftarrow \bar{x}^i \times \{0\}^{n-i}$
11:         Remove $w\sigma + \rho_i \ge 0$ from (46)
12:         $\hat{V}^i \leftarrow \hat{V}^i \setminus \{\bar{x}^i\}$
13:     end if
14:     $V^i \leftarrow V^i \cup \{\bar{x}^i\}$
15: end for
16: Add $x^t\alpha + Q(x^t) + \eta \ge 0$ to (47)
17: Replace (48) with $x^{t+1}\alpha + Q(x^{t+1}) + \eta = 0$
```

4.4 Combined method

Now we outline an implementation of the integer L-shaped method that combines the alternating strategy discussed in Section 4.2 with the new optimality cuts presented in Section 4.3.

We keep two disjoint lists of first-stage solutions: in V_{LP} we include solutions for which only $Q_{LP}(x)$ has been computed, while in V we keep solutions for which $Q(x)$ has been evaluated. At any given stage, we assume that for each $x \in V$ we have added an optimality cut that is tight at x. Now, when a candidate integer solution (x^*, z^*, θ^*) is found in the master tree, we check whether $x^* \in V$ or not. If so, we accept the solution as we already know $Q(x^*) \leq \theta^*$. Now, if $x^* \notin V_{LP}$, then we compute $Q_{LP}(x^*)$, we add x^* into V_{LP}, and in case $\theta < Q_{LP}(x^*)$, we add the subgradient cut (42). At this point, if (x^*, θ^*) has been neither accepted nor rejected, we have $x^* \in V_{LP}$ and $\theta^* \geq Q_{LP}(x^*)$. Thus we compute $Q(x^*)$, we move x^* from V_{LP} to V, and in case $\theta < Q(x^*)$, we add the CGLP-based cut (49) and accept the solution otherwise. Algorithm 6 below presents the method.

4.5 Results

In this section we address the performance of the variants of the integer L-shaped method discussed so far. Given that the implementations differ in the cut strategy used and in the type of optimality cut added, we consider the following combinations:

1. *Std-Std*: standard cut strategy and standard optimality cut (41); see Section 4.1.

2. *Alt-Std*: alternating cut strategy and standard optimality cut (41); see Section 4.2.

3. *Std-CGLP*: standard cut strategy and new optimality cut (49); see Section 4.3.

4. *Alt-CGLP*: alternating cut strategy and new optimality cut (49); see Section 4.4.

In other words, *Std-Std* corresponds to the usual implementation of the integer L-shaped method, on top of which the variants are built.

Algorithm 6 Optimality cut function with alternating cut strategy and CGLP-based optimality cuts

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}, V, V_{LP}$
Output: true if solution is feasible, false otherwise
 1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
 2:     return true
 3: end if
 4: if $x^* \notin V_{LP}$ then
 5:     Compute $Q_{LP}(x^*)$.
 6:     $V_{LP} \leftarrow V_{LP} \cup \{x^*\}$.
 7:     if $\theta^* < Q_{LP}(x^*)$ then
 8:         Add the subgradient cut (42).
 9:         return false
10:     end if
11: end if
    // Now we have $x^* \in V_{LP}$ and $\theta^* \geq Q_{LP}(x^*)$
12: Compute $Q(x^*)$.
13: Update CGLP.
14: $V \leftarrow V \cup \{x^*\}$.
15: $V_{LP} \leftarrow V_{LP} \setminus \{x^*\}$
16: if $\theta^* < Q(x^*)$ then
17:     Solve CGLP to obtain $\alpha$ and $\eta$
18:     Add the integer optimality cut (49)
19:     return false
20: else
21:     return true
22: end if
```

Our computational implementation uses CPLEX 12.5.0.1 as a solver and its Callable Library for advanced control routines. Since either optimality cuts (41) or (49) are part of the complete formulation (MP) but not included from the beginning, we have to add them on-the-fly through the optimality cut function. This routine is called every time the solver finds a candidate integer solution to the master problem and is in charge of generating an optimality cut if needed. In the case of CGLP, it calls additional subroutines to make the required updates to generate (49). We include the formulation of the first-stage set in CGLP, along the subgradients cuts derived from the linear relaxation of $Q(x)$ used in Benders' decomposition.

The experiments were carried out on a personal computer with 3.33 Ghz CPU, 4 Gb of RAM, and running Linux. A relative optimality gap of 0.01% was set as stopping criterion and a time limit of 7200 seconds was imposed. We do not report on the extensive form of the instances as solving them using an off-the-shelf solver is much slower than the decomposition approaches.

4.5.1 Stochastic server location problem

The stochastic server location problem is described in [48]. Given n locations, in the first stage we are asked to decide where to place servers so that the demand given by m potential customers is satisfied in the second stage. The uncertain data is the set of customers to be served in the second stage and the objective is to maximize the expected second-stage revenue minus the first-stage installation costs. In minimization form, the problem can be written as

$$\begin{aligned} \min \quad & cx + Q(x) \\ \text{s.t.} \quad & x \in \{0,1\}^n, \end{aligned}$$

where $Q(x) := \mathbb{E}_\xi[Q_\xi(x)]$ and

$$\begin{aligned} Q_\xi(x) := \quad \min \quad & q_1 y + q_2 s \\ \text{s.t.} \quad & W_1 y + W_2 s \geq h(\xi) - Tx \\ & y \in \{0,1\}^{nm} \\ & s \in \mathbb{R}^n_+. \end{aligned}$$

The random right-hand-side vector $h(\xi)$ represents the set of active customers in a given scenario.

We tested our methods on the instances presented in [49]. Instances named `SSLP.n.m.k` have n locations, m customers, and k scenarios, leading to n binary variables in the first stage and nm binary variables and n nonnegative variables per scenario in the second stage. For each n and m, five replications with k scenarios each are considered. We did not include instances having $n = 5$ as all of them took less than 1 second to solve with any method.

Tables 11 and 12 summarize our results. In both tables, column *Instance* indicates the combination of n, m and k as above. Headers *Std-Std*, *Alt-Std*, *Std-CGLP*, *Alt-CGLP* denote the type of implementation under consideration. Here we present the averages over the five replications of each instance. Detailed results are given in Tables 16 and 17 in Section 4.7.

In Table 11 we present the overall results for all four methods. Columns *Nodes* show the average number of nodes explored in the master problem. Columns *Time* show the average total time spent to reach optimality, which includes computing an initial lower bound L, solving the LP relaxation with Benders' decomposition, and exploring and evaluating candidate solutions in the master problem. Best running times are in bold.

From Table 11, we see that there is no significant variation in the number of explored nodes among the different methods. Now, the implementations that use the alternating cut strategy clearly outperform the other two methods, with speedups of one order of magnitude. On the other hand, with a few exceptions, the use of CGLP-based cuts does not cause major changes in the total running time, especially when combined with the alternating cut strategy. This can be explained by the fact that in these problems, the first-stage is very simple as $X = \{0, 1\}^n$ with $n \leq 15$, which does not present a challenge for CPLEX.

To understand the effect of alternating cuts, in Table 12 we present details regarding subproblems. Recall that every time a candidate integer solution is found, we have to check whether it is feasible, by either solving a series of MIPs or LPs, one per scenario, and then add a cut to discard the solution if necessary. Headers #*LP* and #*MIP* denote the average number of times a candidate solution was

Table 11: Stochastic server location: overall results.

Instance	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
SSLP.10.50.50	402.4	70.9	394.8	71.5	406.8	**6.8**	404.2	**6.8**
SSLP.10.50.100	370.2	91.1	373.0	90.5	371.8	**13.2**	371.0	13.6
SSLP.10.50.500	381.0	548.5	385.0	561.7	386.8	**64.0**	385.0	65.5
SSLP.10.50.1000	360.0	1294.1	357.8	1307.1	367.4	**128.2**	368.2	129.3
SSLP.10.50.2000	392.2	3298.0	371.4	3160.7	404.4	339.3	404.6	**336.7**
SSLP.15.45.5	772.6	81.5	750.2	89.0	763.4	**2.7**	764.6	2.8
SSLP.15.45.10	1408.0	400.9	1370.8	353.6	1450.8	**6.1**	1414.0	6.5
SSLP.15.45.15	1500.0	534.3	1498.4	539.1	1526.0	**11.7**	1523.6	11.9
SSLP.15.45.20	495.6	358.4	481.4	347.8	500.4	**8.0**	502.4	8.1
SSLP.15.45.25	733.0	708.4	698.8	704.9	737.8	**16.7**	732.2	17.4

checked using linear or mixed-integer subproblems, while headers *Time LP* and *Time MIP* indicate the average time spent in each case. We focus only on the implementations *Std-Std* and *Alt-Std* as the comparison for the remaining pair is similar.

From Table 12, we see that with the alternating cut strategy the number of MIP evaluations reduces considerably. This means that in the problems we tested, most of the time it is not necessary to compute the exact second-stage value of a given first-stage solution to reject it. Furthermore, only a small fraction of these solutions are visited twice, and only in those cases we have to solve MIP subproblems. The benefits are evident.

Table 12: Stochastic server location: subproblems details.

Instance	Std-Std				Alt-Std			
	#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
SSLP.10.50.50	147.6	147.6	1.8	65.8	148.6	3.4	1.7	1.8
SSLP.10.50.100	131.6	131.6	3.3	81.0	130.8	3.8	3.0	3.5
SSLP.10.50.500	131.6	131.6	16.4	497.2	130.6	3.0	14.8	15.2
SSLP.10.50.1000	132.0	132.0	33.6	1193.3	127.2	3.0	30.2	32.8
SSLP.10.50.2000	142.6	142.6	72.4	3082.5	143.4	4.2	67.3	133.2
SSLP.15.45.5	143.0	143.0	0.3	80.6	143.2	5.8	0.3	1.9
SSLP.15.45.10	262.0	262.0	1.1	398.2	268.5	5.3	1.1	3.6
SSLP.15.45.15	310.6	310.6	1.9	530.1	317.4	6.0	1.9	7.9
SSLP.15.45.20	99.4	99.4	0.7	356.1	98.4	3.2	0.7	5.9
SSLP.15.45.25	162.4	162.4	1.5	704.3	163.0	5.4	1.4	12.8

4.5.2 Stochastic multiple binary knapsack problem

The second benchmark set corresponds to a class of stochastic multiple binary knapsack problems. They have the form

$$\begin{aligned} \min \quad & cx + dz + Q(x) \\ \text{s.t.} \quad & Ax + Cz \geq b \\ & x \in \{0,1\}^n \\ & z \in \{0,1\}^n, \end{aligned}$$

where $Q(x) := \mathbb{E}_\xi[Q_\xi(x)]$,

$$\begin{aligned} Q_\xi(x) := \quad \min \quad & q(\xi)y \\ \text{s.t.} \quad & Wy \geq h - Tx \\ & y \in \{0,1\}^n, \end{aligned}$$

and all data are nonnegative integers. In the second-stage problem, only the objective vector $q(\xi)$ is random, following a discrete distribution with finitely many scenarios.

We generated 30 instances of the above problem with $n = 120$ and 20 equiprobable scenarios. The systems $Ax + Cz \geq b$ and $Wy \geq h - Tx$ have 50 and 5 rows, respectively. The entries of A, C, T, W, c, d, and q are i.i.d. sampled from the uniform distribution over $\{1, \ldots, 100\}$. We set $b = \frac{3}{4}(A\mathbf{1} + C\mathbf{1})$ and $h = \frac{3}{4}(T\mathbf{1} + W\mathbf{1})$, with $\mathbf{1}$ denoting the n-dimensional vector of ones.

We divided the instances intro three groups depending on how much time the standard implementation took to solve each of them: *Easy* (less than 200 seconds, instances 1–6), *Medium* (between 200 and 1000 seconds, instances 7–18), and *Hard* (more than 1000 seconds, instances 19–29). We omitted instance 30 since none of the methods was able to solve it to optimality within the time limit.

Tables 13, 14, and 15 below summarize the results. Column *Difficulty* denotes the instance class. The remaining headers and columns are as in Tables 11 and 12. Detailed results are given in Tables 18, 19, and 20 in Section 4.7.

Table 13: Stochastic multiple knapsack: overall results.

Difficulty	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
Easy	151531.5	87.1	127696.0	**82.5**	154611.7	86.0	133724.2	82.8
Medium	945487.8	520.8	714822.5	453.8	940249.3	516.1	748502.7	**446.7**
Hard	3356158.1	2125.7	2654448.1	1833.6	3371088.5	2065.2	2656526.5	**1756.3**

From Table 13, we see that the application of the alternating cut strategy does not yield the time savings we saw with the stochastic server location problems. On the other hand, in most instances, adding CGLP-based cuts instead of standard cuts yields reductions in both the number of nodes and the total time, regardless of the cut strategy being used. We would like to conclude that these improvements are due to the fact that CGLP-based cuts help to explore the master tree. However, at this point, that is not completely clear, as for example, time reductions could be consequence of less evaluations of $Q(x)$ and not because of the strength of the new cuts.

To aid our analysis, in Table 14 we report the average number of candidate solutions for which $Q_{LP}(x)$ and $Q(x)$ were evaluated and the average time spent doing so. This time we compare *Std-Std* and *Std-CGLP*, and the notation is similar to that of Table 12.

Table 14: Stochastic multiple knapsack: subproblems details.

Difficulty	Std-Std				Std-CGLP			
	#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
Easy	13.7	13.7	0.0	25.7	14.7	14.7	0.0	28.1
Medium	49.2	49.2	0.1	99.1	54.1	54.1	0.1	107.2
Hard	112.3	112.3	0.2	231.6	114.9	114.9	0.2	235.8

We observe that both implementations require roughly the same number of evaluations of both $Q_{LP}(x)$ and $Q(x)$, which explains why alternating cuts does not outperform the standard cut strategy. Moreover, the difference in the time solving subproblems is very small compared to the total running times presented in Table 13. Thus, the reductions observed in Table 13 can be attributed to the better approximation of the first-stage set given by the CGLP-based cuts and not to the variability of the evaluations. In this regard, it is important to stress that, in principle, having a better description of the first-stage set does not have a direct relationship with the number of candidates solutions found in the master tree, and actually, having more candidates could hurt the total running time if their evaluation is too costly. However, in situations where after decomposing the problem the burden of the computation lies on the master problem, our improved cuts may prove beneficial as exemplified by our results.

Finally, in Table 15 we present the overhead incurred by using CGLP to generate cuts, that is, the time spent in additional operations to maintain and solve CGLP through the method. For each class, column $|V|$ shows the average final size of V, which is the number of candidate solutions for which $Q(x)$ was evaluated exactly. Headers *Update* and *Generate* denote the average total time spent updating the formulation of CGLP and actually solving the system to find an optimality cut, respectively. This additional time is already included in the total running time presented in Table 13.

Table 15: Stochastic multiple knapsack: CGLP overhead.

Difficulty	$\|V\|$	Update	Generate
Easy	14.7	0.0	0.5
Medium	54.1	0.2	7.2
Hard	114.9	0.4	24.7

As expected, the overhead increases as more solutions are included in the extended formulation. Updating CGLP takes practically no time, whereas generating the cut takes a nonnegligible amount of time. However, compared to the total running time, the overhead is very small and the effort of computing improved cuts pays off as shown in Table 13. For more complicated problems where the number of binary first-stage variables is too large or where too many candidate solutions are evaluated, the cost of maintaining CGLP is likely to be higher. In those cases, we can enforce rules to limit the number of calls to CGLP, such as using the standard optimality cuts as a baseline and applying the improved cuts only once in a while.

4.6 Concluding remarks

We have presented two modifications to the integer L-shaped method with the objective of reducing the running time of the algorithm. The first one, termed alternating cuts strategy, seeks to avoid expensive evaluations of the second-stage cost function, while the second, the use of CGLP-based optimality cuts, helps to better approximate the shape of the epigraph of the cost function when evaluations at different points are available. Our computational results suggest the following:

1. The alternating cuts strategy works better in problems where the computational bottleneck of (IP) is in evaluating $Q(x)$. Even when that is not the case, this modification does not seem to hurt the total running times and thus it could be considered as the base method on top of which more evolved algorithms can be built.

2. CGLP-based cuts are a viable alternative when the first-stage set is difficult to explore and computing $Q(x)$ is a relatively cheap operation. As the sole purpose of these new cuts is to have a better representation of the epigraph of the second-stage cost function within the master problem, there is no guarantee about the number or the sequence of solutions for which $Q(x)$ is evaluated, and thus, in general, this method performs well when the impact of this variability is small compared with the effort of solving the master problem.

3. We also point out that our overall computational experience indicates that CGLP-based cuts are particularly suitable for problems having additional integer variables in the set Z, since a deep cut discarding a point (x^*, θ^*) in the (x, θ)-space may also prove effective in discarding a large number of points of the form (x^*, z, θ^*) for $z \in Z$.

4. As favorable conditions for both modifications are unlikely to be attained at the same time, we observe that time reductions in a combined method are mainly consequence of one strategy or the other, but not because of the combination of both. That being said, it would be interesting to experiment with implementations where CGLP also incorporates approximations of $Q(x)$ such as subgradient cuts or ad-hoc lower bounds rather than exact evaluations only. That would require also keeping track of first-stage vectors x for which estimates of $Q(x)$ have been computed.

5. Finally, in more general settings where $Q(x)$ is an easy-to-evaluate nonconvex function for which a tractable convex underestimator is not available, CGLP-based cuts may prove helpful in solving problems having the form (IP). Situations where $Q(x)$ is given by black-box computations remain a case study to be explored.

4.7 Detailed computational results

4.7.1 Stochastic server location problem

Table 16: Stochastic server location: overall results per instance.

Instance	Rep.	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
	a	478	61.1	485	67.8	468	**7.0**	466	7.1
	b	452	91.3	434	85.5	472	**6.7**	466	**6.7**
SSLP.10.50.50	c	300	79.3	297	80.0	303	**7.0**	298	7.1
	d	237	25.3	224	26.9	230	**5.0**	230	**5.0**
	e	545	97.5	534	97.2	561	**8.2**	561	8.3
	a	452	109.7	434	106.2	462	**17.1**	470	18.3
	b	497	80.3	494	83.7	493	**11.0**	493	11.1
SSLP.10.50.100	c	313	95.3	291	98.0	302	**12.8**	289	13.6
	d	216	49.5	224	43.6	229	10.7	229	**10.5**
	e	373	120.5	422	121.0	373	**14.2**	374	14.4
	a	466	605.5	470	643.9	472	**63.3**	476	63.5
	b	441	482.6	447	492.9	449	**57.7**	449	57.8
SSLP.10.50.500	c	277	571.7	292	557.6	275	**64.0**	271	64.8
	d	235	348.8	239	353.5	247	**57.5**	247	57.6
	e	486	733.8	477	760.8	491	**77.5**	482	84.0
	a	481	1542.1	473	1549.6	486	**134.5**	487	135.4
	b	473	1128.7	477	1142.2	460	**114.5**	466	116.8
SSLP.10.50.1000	c	276	1509.3	261	1509.7	282	**124.2**	279	125.5
	d	225	752.8	227	782.2	229	**113.2**	229	113.7
	e	345	1537.6	351	1551.8	380	**154.4**	380	155.3
	a	466	3777.1	467	3769.2	472	382.7	478	**373.2**
	b	472	2565.3	471	2751.8	483	**246.7**	478	251.0
SSLP.10.50.2000	c	286	3189.4	286	3158.8	302	368.9	300	**360.5**
	d	219	1937.1	219	1994.5	223	**249.0**	225	249.4
	e	518	5021.2	414	4129.2	542	**449.4**	542	449.6
	a	230	11.3	233	11.6	244	**0.7**	244	**0.7**
	b	261	2.9	262	3.0	270	**0.5**	262	**0.5**
SSLP.15.45.5	c	2364	320.9	2288	354.9	2298	**9.9**	2294	10.2
	d	870	56.2	826	58.7	872	**1.4**	888	1.7
	e	138	16.4	142	16.8	133	**1.0**	135	**1.0**
	a	430	79.0	442	80.1	429	**2.7**	428	2.8
	b	284	189.0	251	190.9	256	**6.2**	278	7.6
SSLP.15.45.10	c	2384	245.0	2240	236.6	2512	**7.4**	2449	7.7
	d	2534	1090.7	2550	906.8	2606	**7.9**	2501	8.0
	a	1408	1646.1	1329	1594.5	1368	**13.1**	1358	13.3
	b	223	55.7	216	55.5	212	**2.3**	219	**2.3**
SSLP.15.45.15	c	2676	580.6	2718	611.0	2791	19.0	2785	**18.9**
	d	2986	359.1	2994	404.1	3038	**22.3**	3024	23.0
	e	207	30.1	235	30.2	221	**1.9**	232	**1.9**
	a	498	186.4	469	181.4	506	**4.0**	523	4.1
	b	351	87.2	335	87.5	341	**7.6**	331	**7.6**
SSLP.15.45.20	c	380	196.8	358	193.4	380	**5.1**	387	5.2
	d	552	873.0	548	898.1	560	**20.7**	562	20.9
	e	697	448.4	697	378.5	715	**2.8**	709	**2.8**
	a	658	554.1	629	532.0	662	**18.4**	633	18.5
	b	671	324.7	620	435.1	670	9.0	680	**6.7**
SSLP.15.45.25	c	433	165.2	399	160.7	447	**11.8**	422	11.9
	d	965	435.2	946	465.7	967	**26.9**	1001	32.3
	e	938	2062.7	900	1931.0	943	**17.6**	925	17.8

Table 17: Stochastic server location: subproblems details per instance.

Instance	Rep.	Std-Std				Alt-Std			
		#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
SSLP.10.50.50	a	189	189	2.2	55.5	185	3	2.1	1.7
	b	154	154	1.6	86.2	166	3	1.6	1.6
	c	113	113	1.6	74.5	109	4	1.6	2.3
	d	44	44	0.6	21.3	45	2	0.6	0.9
	e	238	238	2.8	91.6	238	5	2.7	2.6
SSLP.10.50.100	a	181	181	4.3	98.4	183	6	3.9	6.4
	b	176	176	4.0	69.7	175	2	3.7	0.9
	c	112	112	3.1	86.2	109	5	2.9	4.1
	d	51	51	1.3	40.9	50	2	1.2	2.2
	e	138	138	3.7	109.8	137	4	3.3	4.0
SSLP.10.50.500	a	178	178	21.1	549.8	179	2	19.1	11.0
	b	152	152	17.8	428.5	150	2	15.2	7.4
	c	89	89	13.7	523.9	89	4	12.0	18.6
	d	56	56	8.1	303.3	56	2	8.1	12.4
	e	183	183	21.1	680.4	179	5	19.8	26.7
SSLP.10.50.1000	a	188	188	46.4	1429.6	185	3	41.9	29.4
	b	163	163	36.6	1028.5	156	2	32.4	21.2
	c	106	106	29.6	1410.1	95	3	25.8	30.2
	d	56	56	16.0	665.2	55	2	15.3	27.3
	e	147	147	39.4	1433.1	145	5	35.5	56.1
SSLP.10.50.2000	a	184	184	92.6	3548.0	181	5	82.4	169.9
	b	158	158	70.3	2352.7	156	2	65.4	44.2
	c	98	98	60.7	2980.4	103	5	56.6	167.0
	d	59	59	34.2	1746.0	58	2	31.2	62.1
	e	214	214	104.3	4785.4	219	7	101.0	222.7
SSLP.15.45.5	a	28	28	0.1	10.9	28	2	0.1	0.2
	b	42	42	0.1	2.5	41	4	0.1	0.2
	c	481	481	1.0	318.3	496	17	0.9	7.7
	d	154	154	0.3	55.2	141	4	0.3	0.6
	e	10	10	0.0	16.1	10	2	0.0	0.7
SSLP.15.45.10	a	93	93	0.3	77.8	90	2	0.3	1.6
	b	68	68	0.2	188.2	67	5	0.2	5.4
	c	501	501	2.2	240.1	538	9	2.3	2.9
	d	386	386	1.7	1086.8	379	5	1.7	4.4
SSLP.15.45.15	a	263	263	1.6	1642.3	262	4	1.5	9.7
	b	41	41	0.2	54.4	39	2	0.2	1.0
	c	623	623	4.3	572.8	645	16	4.4	11.8
	d	597	597	3.3	352.0	613	6	3.1	16.2
	e	29	29	0.2	29.0	28	2	0.2	0.8
SSLP.15.45.20	a	134	134	0.9	183.7	132	2	0.9	1.4
	b	63	63	0.4	85.1	61	2	0.4	5.6
	c	61	61	0.4	195.2	60	4	0.4	3.6
	d	148	148	1.1	870.2	145	6	1.0	18.0
	e	91	91	0.7	446.2	94	2	0.7	0.7
SSLP.15.45.25	a	156	156	1.3	550.0	147	4	1.2	14.4
	b	135	135	1.3	321.0	148	4	1.4	5.3
	c	73	73	0.6	162.1	74	4	0.6	8.8
	d	213	213	2.2	430.0	215	7	2.1	21.9
	e	235	235	2.0	2058.5	231	8	1.9	13.7

4.7.2 Stochastic multiple binary knapsack problem

Table 18: Stochastic multiple knapsack: overall results per instance.

Instance	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
1	27705	26.4	27615	27.0	27837	25.5	26295	**24.9**
2	63528	41.1	55448	38.6	65213	41.1	57170	**38.2**
3	93185	59.9	87560	60.2	101121	57.8	81480	**50.9**
4	137303	101.1	121687	97.3	132782	**89.1**	127963	**89.1**
5	224063	107.2	183462	**94.1**	244755	112.1	251017	128.3
6	363405	186.6	290404	177.5	355962	190.1	258420	**165.5**
7	503998	245.8	401809	204.4	517313	250.4	397677	**200.2**
8	436738	267.5	310136	218.0	431356	249.3	334569	**214.8**
9	470356	273.3	451931	269.7	502174	280.5	450104	**254.8**
10	507120	315.6	320672	**251.1**	518329	333.1	342582	257.5
11	623424	379.4	675292	404.9	637749	422.5	615580	**342.6**
12	887595	468.7	672117	**422.7**	954211	502.8	741931	436.2
13	1099397	541.0	1024147	692.2	1172464	579.1	984003	**527.4**
14	1416129	686.6	880154	**516.9**	1484427	711.5	1057895	600.3
15	1650580	714.4	1120524	**509.8**	1692521	726.8	1148229	516.0
16	1322774	749.9	832266	**533.7**	1013473	572.2	956447	579.2
17	1197577	771.1	900476	**652.4**	1192205	753.2	974525	686.0
18	1230166	836.7	988346	769.7	1166769	811.6	978490	**745.9**
19	2189204	1158.0	1618305	**950.0**	2225393	1160.4	1713778	962.0
20	2395096	1460.9	1663945	1142.5	2383548	1404.2	1756720	**1109.5**
21	3277812	1488.2	2789613	**1328.8**	3563188	1603.1	3144784	1499.3
22	2702878	1664.7	2244862	**1422.6**	2816341	1714.0	2087732	1430.1
23	2309196	1825.3	1919811	1711.6	2306792	1715.5	1833302	**1520.5**
24	3301135	1998.1	2690441	1771.6	3101311	1816.8	2580654	**1620.4**
25	3346788	2310.7	2987190	2149.9	3541754	2346.8	2998747	**2068.1**
26	3024670	2319.8	2966064	2373.0	3087399	2258.1	2806757	**2172.3**
27	3890594	2344.4	3225433	2099.7	3787260	2210.1	3128508	**1980.4**
28	4762714	3223.2	3253202	2425.3	4449516	2890.2	3285741	**2311.3**
29	5717652	3589.2	3840063	2795.1	5819471	3597.8	3885068	**2645.9**

Table 19: Stochastic multiple knapsack: subproblems details per instance.

Instance	Std-Std				Std-CGLP			
	#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
1	9	9	0.0	14.4	9	9	0.0	14.7
2	9	9	0.0	15.7	9	9	0.0	15.8
3	14	14	0.0	24.7	14	14	0.0	26.0
4	24	24	0.0	46.2	24	24	0.0	45.9
5	12	12	0.0	21.0	12	12	0.0	19.8
6	14	14	0.0	32.1	20	20	0.0	46.2
7	10	10	0.0	17.3	10	10	0.0	17.2
8	40	40	0.1	80.3	40	40	0.1	80.7
9	34	34	0.1	74.5	36	36	0.1	74.6
10	46	46	0.1	97.9	49	49	0.1	102.1
11	46	46	0.1	77.6	47	47	0.1	78.9
12	45	45	0.1	108.5	51	51	0.1	123.0
13	45	45	0.1	87.2	87	87	0.2	160.9
14	51	51	0.1	124.4	51	51	0.1	123.9
15	22	22	0.0	29.9	26	26	0.1	36.2
16	79	79	0.2	128.8	74	74	0.2	119.3
17	80	80	0.2	168.0	81	81	0.2	167.5
18	92	92	0.2	194.7	97	97	0.2	202.1
19	66	66	0.1	134.3	65	65	0.1	131.9
20	97	97	0.2	193.0	98	98	0.2	193.9
21	49	49	0.1	99.8	48	48	0.1	97.7
22	93	93	0.2	245.2	91	91	0.2	237.2
23	175	175	0.4	341.7	176	176	0.4	339.1
24	89	89	0.2	211.2	92	92	0.2	221.1
25	127	127	0.3	222.2	127	127	0.3	221.8
26	155	155	0.3	331.5	157	157	0.3	331.4
27	103	103	0.2	246.0	111	111	0.2	264.1
28	150	150	0.3	263.7	152	152	0.3	268.0
29	131	131	0.3	259.4	147	147	0.3	287.6

Table 20: Stochastic multiple knapsack: CGLP overhead per instance.

Instance	$\|V\|$	Update	Generate
1	9	0.0	0.2
2	9	0.0	0.2
3	14	0.0	0.4
4	24	0.0	1.0
5	12	0.0	0.3
6	20	0.0	0.8
7	10	0.0	0.2
8	40	0.1	2.7
9	36	0.1	2.5
10	49	0.1	3.4
11	47	0.1	4.7
12	51	0.1	4.5
13	87	0.2	13.2
14	51	0.1	5.3
15	26	0.1	1.3
16	74	0.2	11.9
17	81	0.3	15.7
18	97	0.4	21.2
19	65	0.2	6.9
20	98	0.3	21.7
21	48	0.1	4.0
22	91	0.3	18.8
23	176	0.9	41.2
24	92	0.3	13.8
25	127	0.4	29.8
26	157	0.6	37.0
27	111	0.4	25.1
28	152	0.6	39.6
29	147	0.6	33.3

REFERENCES

[1] Albareda-Sambola, M., van der Vlerk, M., and Fernández, E., "Exact solutions to a class of stochastic generalized assignment problems," *European Journal of Operational Research*, vol. 173, pp. 465–487, 2006.

[2] Anbil, R., Gelman, E., Patty, B., and Tanga, R., "Recent advances in crew-pairing optimization at American Airlines," *Interfaces*, vol. 21, pp. 62–74, 1991.

[3] Audet, C., Brimberg, J., Hansen, P., Digabel, S. L., and Mladenović, N., "Pooling problem: Alternate formulations and solution methods," *Management Science*, vol. 50, pp. 761–776, 2004.

[4] Avis, D., Bremner, D., and Seidel, R., "How good are convex hull algorithms?," *Computational Geometry*, vol. 7, pp. 265–301, 1997.

[5] Balas, E., "Disjunctive Programming," in *Discrete Optimization II* (Johnson, E., Hammer, P., and Korte, B., eds.), vol. 5 of *Annals of Discrete Mathematics*, pp. 3–51, Elsevier, 1979.

[6] Ben-Tal, A., Eiger, G., and Gershovitz, V., "Global minimization by reducing the duality gap," *Mathematical Programming*, vol. 63, pp. 193–212, 1994.

[7] Benders, J., "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[8] Bienstock, D., "Computational study of a family of mixed-integer quadratic programming problems," *Mathematical Programming*, vol. 74, pp. 121–140, 1996.

[9] Boros, E., Elbassioni, K., Gurvich, V., and Tiwary, H., "The negative cycles polyhedron and hardness of checking some polyhedral properties," *Annals of Operations Research*, vol. 188, pp. 63–76, 2011.

[10] Conforti, M., Cornuéjols, G., and Zambelli, G., "Extended formulations in combinatorial optimization," *4OR*, vol. 8, pp. 1–48, 2010.

[11] Dadush, D., Dey, S., and Vielma, J., "The split closure of a strictly convex body," *Operations Research Letters*, vol. 39, pp. 121–126, 2011.

[12] Dantzig, G., "Discrete-variable extremum problems," *Operations Research*, vol. 5, pp. 266–288, 1957.

[13] Dantzig, G., *Linear programming and extensions.* Princeton university press, 1965.

[14] de Farias, I., "Semi-continuous Cuts for Mixed-Integer Programming," in *Integer Programming and Combinatorial Optimization* (Bienstock, D. and Nemhauser, G., eds.), vol. 3064 of *Lecture Notes in Computer Science*, pp. 163–177, Springer Berlin Heidelberg, 2004.

[15] de Farias, I., Johnson, E., and Nemhauser, G., "Branch-and-cut for combinatorial optimization problems without auxiliary binary variables," *The Knowledge Engineering Review*, vol. 16, pp. 25–39, 2 2001.

[16] de Farias, I. and Nemhauser, G., "A polyhedral study of the cardinality constrained knapsack problem," *Mathematical Programming*, vol. 96, pp. 439–467, 2003.

[17] de Farias, I. and Zhao, M., "A polyhedral study of the semi-continuous knapsack problem," *Mathematical Programming*, 2012.

[18] Dey, S. S. and Gupte, A., "Analysis of MILP techniques for the Pooling Problem,"

[19] Epstein, R., Neely, A., Weintraub, A., Valenzuela, F., Hurtado, S., Gonzalez, G., Beiza, A., Naveas, M., Infante, F., Alarcon, F., and others, "A Strategic Empty Container Logistics Optimization in a Major Shipping Company," *Interfaces*, vol. 42, pp. 5–16, 2012.

[20] Fiorini, S., Massar, S., Pokutta, S., Tiwary, H., and de Wolf, R., "Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds," in *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, (New York, NY, USA), pp. 95–106, ACM, 2012.

[21] Fiorini, S., Massar, S., Pokutta, S., Tiwary, H., and de Wolf, R., "Exponential Lower Bounds for Polytopes in Combinatorial Optimization," *arXiv:1111.0837v4*, 2013.

[22] Gade, D., Küçükyavuz, S., and Sen, S., "Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs," *Mathematical Programming*, pp. 1–26, 2012.

[23] Gendreau, M., Laporte, G., and Séguin, R., "An exact algorithm for the vehicle routing problem with stochastic demands and customers," *Transportation Science*, vol. 29, pp. 143–155, 1995.

[24] Gomory, R., "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, pp. 275–278, 1958.

[25] Gomory, R., "An algorithm for the mixed integer problem," tech. rep., DTIC Document, 1960.

[26] Gupte, A., "n-dimensional superincreasing knapsack polytopes have $\mathcal{O}(n)$ facets," *Optimization Online*, 2013.

[27] HAVERLY, C. A., "Studies of the behavior of recursion for the pooling problem," *ACM SIGMAP Bulletin*, pp. 19–28, 1978.

[28] JEROSLOW, R., "On defining sets of vertices of the hypercube by linear inequalities," *Discrete Mathematics*, vol. 11, pp. 119–124, 1975.

[29] KALLRATH, J., "Mixed Integer Optimization in the Chemical Process Industry: Experience, Potential and Future Perspectives," *Chemical Engineering Research and Design*, vol. 78, pp. 809–822, 2000.

[30] KALLRATH, J., "Combined strategic and operational planning – an MILP success story in chemical industry," *OR Spectrum*, vol. 24, pp. 315–341, 2002.

[31] KHACHIYAN, L., BOROS, E., BORYS, K., ELBASSIONI, K., and GURVICH, V., "Generating all vertices of a polyhedron is hard," *Discrete & Computational Geometry*, vol. 39, pp. 174–190, 2008.

[32] KROON, L., HUISMAN, D., ABBINK, E., FIOOLE, P.-J., FISCHETTI, M., MARÓTI, G., SCHRIJVER, A., STEENBEEK, A., and YBEMA, R., "The new Dutch timetable: The OR revolution," *Interfaces*, vol. 39, pp. 6–17, 2009.

[33] KÜÇÜKYAVUZ, S. and ZHANG, M., "Finitely Convergent Decomposition Algorithms for Two-Stage Stochastic Pure Integer Programs," *Optimization Online*, 2013.

[34] LAPORTE, G. and LOUVEAUX, F., "The integer L-shaped method for stochastic integer programs with complete recourse," *Operations Research Letters*, vol. 13, pp. 133–142, 1993.

[35] LAPORTE, G., LOUVEAUX, F., and MERCURE, H., "A priori optimization of the probabilistic traveling salesman problem," *Operations research*, vol. 42, pp. 543–549, 1994.

[36] LAPORTE, G., LOUVEAUX, F., and VAN HAMME, L., "Exact solution to a location problem with stochastic demands," *Transportation Science*, vol. 28, pp. 95–103, 1994.

[37] LAPORTE, G., LOUVEAUX, F., and VAN HAMME, L., "An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands," *Operations Research*, vol. 50, pp. 415–423, 2002.

[38] LAWLER, E., "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, vol. 18, pp. 401–405, 1972.

[39] LEE, J., "All-Different Polytopes," *Journal of Combinatorial Optimization*, vol. 6, pp. 335–352, 2002.

[40] LEE, J., "Cropped cubes," *Journal of Combinatorial Optimization*, vol. 7, pp. 169–178, 2003.

[41] LEE, J., LEUNG, J., and DE VRIES, S., "Separating type-I odd-cycle inequalities for a binary-encoded edge-coloring formulation," *Journal of Combinatorial Optimization*, vol. 9, pp. 59–67, 2005.

[42] LEE, J. and MARGOT, F., "On a binary-encoded ILP coloring formulation," *INFORMS Journal on Computing*, vol. 19, pp. 406–415, 2007.

[43] LI, Y. and RICHARD, J.-P., "Cook, Kannan and Schrijver's example revisited," *Discrete Optimization*, vol. 5, pp. 724–734, 2008.

[44] LOVÁSZ, L., "Graph Theory and Integer Programming," in *Discrete Optimization I* (JOHNSON, E., HAMMER, P., and KORTE, B., eds.), vol. 4 of *Annals of Discrete Mathematics*, pp. 141–158, Elsevier, 1979.

[45] MISENER, R. and FLOUDAS, C. A., "Advances for the pooling problem: Modeling, global optimization, and computational studies," *Applied and Computational Mathematics*, vol. 8, pp. 3–22, 2009.

[46] MULDOON, F., ADAMS, W., and SHERALI, H., "Ideal representations of lexicographic orderings and base-2 expansions of integer variables," *Operations Research Letters*, vol. 41, pp. 32–39, 2013.

[47] MURTY, K., "Letter to the Editor—An Algorithm for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, vol. 16, pp. 682–687, 1968.

[48] NTAIMO, L. and SEN, S., "The million-variable march for stochastic combinatorial optimization," *Journal of Global Optimization*, vol. 32, pp. 385–400, 2005.

[49] NTAIMO, L. and TANNER, M., "Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs," *Journal of Global Optimization*, vol. 41, pp. 365–384, 2008.

[50] PEROLD, A., "Large-scale portfolio optimization," *Management Science*, vol. 30, pp. 1143–1160, 1984.

[51] POKUTTA, S. and VYVE, M. V., "A note on the extension complexity of the knapsack polytope," *Operations Research Letters*, vol. 41, pp. 347–350, 2013.

[52] QI, Y. and SEN, S., "Ancestral Benders' Cuts and Multi-term Disjunctions for Mixed-Integer Recourse Decisions in Stochastic Programming," *Optimization Online*, 2013.

[53] ROCKAFELLAR, R., *Convex Analysis*, vol. 28. Princeton University Press, 1996.

[54] ROTHVOSS, T., "The matching polytope has exponential extension complexity," *arXiv:1311.2369*, 2013.

[55] SCHRIJVER, A., *Theory of linear and integer programming*. Wiley, 1998.

[56] SEN, S. and HIGLE, J., "The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification," *Mathematical Programming*, vol. 104, pp. 1–20, 2005.

[57] TAWARMALANI, M. and SAHINIDIS, N. V., *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, vol. 65. Springer, 2002.

[58] TIMPE, C. and KALLRATH, J., "Optimal planning in large multi-site production networks," *European Journal of Operational Research*, vol. 126, pp. 422–435, 2000.

[59] VAN SLYKE, R. and WETS, R., "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics*, vol. 17, pp. 638–663, 1969.

[60] YANNAKAKIS, M., "Expressing combinatorial optimization problems by linear programs," *Journal of Computer and System Sciences*, vol. 43, pp. 441–466, 1991.

VITA

Gustavo I. Angulo Olivares was born in Santiago, Chile, on August 28 1982. After graduating from high school, he enrolled at the School of Physical and Mathematical Sciences at Universidad de Chile. In 2007-2008, Gustavo moved to Valparaíso to work full-time in a collaborative research project between Compañía Sudamericana de Vapores and Universidad de Chile. In 2009, he received a Mathematical Engineering degree from the Department of Mathematical Engineering and a Master's degree in Operations Management from the Department of Industrial Engineering under the supervision of Andrés Weintraub.

In August 2009, Gustavo moved to Atlanta to pursue doctoral studies at the Georgia Institute of Technology. Under the supervision of Shabbir Ahmed and Santanu Dey, he collaborated with ExxonMobil in conducting research on mixed-integer programming and stochastic programming. He worked as a summer intern at ExxonMobil Research and Engineering in 2011 and at Kimberly-Clark Corporation in 2012.

In 2011, Gustavo was awarded the Ramón Salas Edwards Award by the Chilean Institute of Engineers and was nominated finalist for the Franz Edelman Award by INFORMS. In 2013 he obtained an honorable mention in the George Nicholson Student Paper Competition from INFORMS.