# MIXED INTEGER PROGRAMMING APPROACHES FOR NONLINEAR AND STOCHASTIC PROGRAMMING

A Thesis
Presented to
The Academic Faculty

by

Juan Pablo Vielma Centeno

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

Georgia Institute of Technology
August 2009

# MIXED INTEGER PROGRAMMING APPROACHES FOR NONLINEAR AND STOCHASTIC PROGRAMMING

Approved by:


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

Professor Shabbir Ahmed, Advisor
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Professor William J. Cook
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Dr. Zonghao Gu
Gurobi Optimization

Professor Ellis Johnson
School of Industrial and Systems Engineering
*Georgia Institute of Technology*



Date Approved: 2 July 2009

*To my wife Johana,*

*and my mother María Angélica.*

# ACKNOWLEDGEMENTS

I would like to thank my advisors Prof. George L. Nemhauser and Prof. Shabbir Ahmed for their support throughout my PhD. I am extremely grateful for their advice, guidance and inspiration in my research and career. Thanks also to the remaining member of my committee Professor William J. Cook, Dr. Zonghao Gu and Professor Ellis Johnson.

I would specially like to thank my wife and parents for their unconditional love and constant encouragement. I am deeply appreciate all the sacrifices they did to support me over many years of study.

I would also like to thank the faculty and staff of the H. Milton Stewart School of Industrial and Systems Engineering for the high quality education I received during my PhD. I would specially like to thank my fellow students and friends for helping make the time I spent at Georgia Tech some of the best years of my life.

Finally, I would like to acknowledge the partial support for this research from National Science Foundation grants DMI-0121495, DMI-0522485, DMI-0133943, CMMI-0522485 and CMMI-0758234, AFOSR grant FA9550-07-1-0177, a grant from Exxon Mobil Upstream Research Company and the John Morris Fellowship from the Georgia Institute of Technology.

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# SUMMARY

In this thesis we study how to solve some nonconvex optimization problems by using methods that capitalize on the success of Linear Programming (LP) based solvers for Mixed Integer Linear Programming (MILP). A common aspect of our solution approaches is the use, development and analysis of small but strong extended LP/MILP formulations and approximations.

In the first part of this work we develop an LP based branch-and-bound algorithm for mixed integer conic quadratic programs. The algorithm is based on a higher dimensional or lifted polyhedral relaxation of conic quadratic constraints introduced by Ben-Tal and Nemirovski. The algorithm is different from other LP based branch-and-bound algorithms for mixed integer nonlinear programs in that, it is not based on cuts from gradient inequalities and it sometimes branches on integer feasible solutions. We test the algorithm on a series of portfolio optimization problems and show that it significantly outperforms commercial and open source solvers based on both linear and nonlinear relaxations.

In the second part we study the modeling of a class of disjunctive constraints with a logarithmic number of binary variables and constraints. Many combinatorial constraints over continuous variables such as SOS1 and SOS2 constraints can be interpreted as disjunctive constraints that restrict the variables to lie in the union of a finite number of specially structured polyhedra. Known mixed integer formulations for these constraints have a number of binary variables and extra constraints linear in the number of polyhedra. We give sufficient conditions for constructing formulations for these constraints with a number of binary variables and extra constraints logarithmic in the number of polyhedra. Using these conditions we introduce mixed integer binary formulations for SOS1 and SOS2 constraints that have a number of binary variables and extra constraints logarithmic in the number of continuous variables. We also introduce the first mixed integer binary formulations for piecewise linear functions of one and two variables that use a number of binary variables and

extra constraints logarithmic in the number of linear pieces of the functions. We prove that the new formulations for piecewise linear functions have favorable tightness properties and present computational results showing that they can significantly outperform other mixed integer binary formulations.

In the third part we study the modeling of non-convex piecewise linear functions as MILPs. We review several new and existing MILP formulations for continuous piecewise linear functions with special attention paid to multivariate non-separable functions. We compare these formulations with respect to their theoretical properties and their relative computational performance. In addition, we study the extension of these formulations to lower semicontinuous piecewise linear functions.

Finally, in the fourth part we study the strength of MILP formulations for LPs with Probabilistic Constraints. We first study the strength of existing MILP formulations that only considers one row of the probabilistic constraint at a time. We then introduce an extended formulation that considers more than one row of the constraint at a time and use it to computationally compare the relative strength between formulations that consider one and two rows at a time.

# CHAPTER I

# INTRODUCTION

## *1.1 Background*

A Mixed Integer Linear Programming (MILP) problem is a nonconvex optimization problem given by

$$z_{\mathsf{MILPP}} := \max \quad cx + dy \tag{1a}$$

$$s.t.$$

$$Dx + Ey \leq f \tag{1b}$$

$$y \in \mathbb{R}^p \tag{1c}$$

$$x \in \mathbb{Z}^n \tag{1d}$$

where $\mathbb{R}$ is the set of real numbers, $\mathbb{Z}$ is the set of integers, $c \in \mathbb{R}^n$, $d \in \mathbb{R}^p$, $D \in \mathbb{R}^{m \times n}$, $E \in \mathbb{R}^{m \times p}$ and $f \in \mathbb{R}^m$. We denote this problem by MILPP and say a solution $(\tilde{x}, \tilde{y})$ is *feasible* for MILPP if it satisfies constraints (1b)–(1d).

In its 50+ years of history, MILP theory and algorithms have been significantly developed [22, 52, 58, 80, 104, 115, 141] and MILP is now considered standard practice in many applications areas (e.g. [20, 47, 57, 76, 96, 106, 107, 129, 135]). Two reasons for the success of MILP are its modeling flexibility [40, 67, 139] and the effectiveness of state of the art Linear Programming (LP) based solvers [24, 25, 69].

### 1.1.1 LP based solvers

LP based solvers for MILP rely heavily on the LP relaxation of MILPP given by (1a)–(1c), which we denote by LPP. The basis for these solvers is the Branch-and-Bound algorithm [77] that performs an intelligent enumeration of the feasible solutions to MILPP by solving a series of LP problems based on LPP. The simplest version of this algorithm can be described as follows. For any $(l^k, u^k) \in (\mathbb{Z} \cup \{-\infty\})^n \times (\mathbb{Z} \cup \{+\infty\})^n$ we denote by $\mathsf{LPP}(l^k, u^k)$ and $\mathsf{MILPP}(l^k, u^k)$ the problem obtained by adding constraints $l^k \leq x \leq u^k$ to LPP and MILPP

respectively. We also denote by $z_{\mathsf{LPP}(l^k,u^k)}$ and $z_{\mathsf{MILPP}(l^k,u^k)}$ the optimal objective value of $\mathsf{LPP}(l^k,u^k)$ and $\mathsf{MILPP}(l^k,u^k)$. In addition, we say that a solution $(\tilde{x},\tilde{y})$ feasible for $\mathsf{LPP}(l^k,u^k)$ is *integer feasible* if it is also feasible for $\mathsf{MILPP}$. A branch-and-bound node $k$ is defined by some $(l^k,u^k,\mathrm{UB}^k)\in\mathbb{Z}^{2n}\times(\mathbb{R}\cup\{+\infty\})$ where $(l^k,u^k)$ are the bounds defining the node and $\mathrm{UB}^k$ is an upper bound on $z_{\mathsf{MILPP}(l^k,u^k)}$. Finally, we denote by LB the global lower bound on $z_{\mathsf{MILPP}}$ and by $\mathcal{H}$ the set of active branch-and-bound nodes. With these definitions the basic branch-and-bound algorithm is given by the pseudocode in Figure 1

```
1  Set global lower bound LB := −∞.
2  Set l_i^0 := −∞, u_i^0 := +∞ for all i ∈ {1,...,n}.
3  Set UB^0 = +∞.
4  Set node list H := {(l^0, u^0, UB^0)}.
5  while H ≠ ∅ do
6  |  Select and remove a node (l^k, u^k, UB^k) ∈ H.
7  |  Solve LPP(l^k, u^k).
8  |  if LPP(l^k, u^k) is feasible and z_LPP(l^k,u^k) > LB then
9  |  |  Let (x̂^k, ŷ^k) be the optimal solution to LPP(l^k, u^k).
10 |  |  if x̂^k ∈ Z^n then
11 |  |  |  LB := z_LPP(l^k,u^k).
12 |  |  else                                   /* Branch on x̂^k */
13 |  |  |  Pick i_0 in {i ∈ {1,...,n} : x̂_i^k ∉ Z}.
14 |  |  |  Let l_i = l_i^k, u_i = u_i^k for all i ∈ {1,...,n} \ {i_o}.
15 |  |  |  Let u_{i_0} = ⌊x̂_{i_0}^k⌋, l_{i_0} = ⌊x̂_{i_0}^k⌋ + 1.
16 |  |  |  H := H ∪ {(l^k, u, z_LPP(l^k,u^k)), (l, u^k, z_LPP(l^k,u^k))}
17 |  |  end
18 |  end
19 |  Remove every node (l^k, u^k, UB^k) ∈ H such that UB^k ≤ LB.
20 end
```

**Figure 1:** Basic LP Based Branch-and-Bound Algorithm.

Lines 8 and 19 of the algorithm eliminate all nodes that cannot have any integer feasible solutions with an objective value larger than that of the best integer feasible solution found so far. If we omit these lines the algorithm will essentially enumerate every $\tilde{x}\in\mathbb{Z}^n$ that can be completed to a solution $(\tilde{x},\tilde{y})$ feasible for $\mathsf{MILPP}$ and chose the one for which $z_{\mathsf{LPP}(\tilde{x},\tilde{x})}$ is largest. Hence, these steps are crucial for the performance of the algorithm. One of the reasons for effectiveness of modern LP based solvers for MILP is that they use a wide

array of techniques to improve bounds $z_{\mathsf{LPP}(l^k,u^k)}$ in line 8 and $\mathrm{UB}^k$ in line 19 to a number closer to $z_{\mathsf{MILPP}(l^k,u^k)}$. This is also the reason that $z_{\mathsf{LPP}}$ being close to $z_{\mathsf{MILPP}}$ is a desirable property for MILPs.

Another reason for the effectiveness of modern solvers is the use of the warm start capabilities of the simplex algorithm for solving a series of very similar LP problems. For example, $\mathsf{LPP}(l^k, u)$ constructed in line 16 of the algorithm in Figure 1 is almost identical to $\mathsf{LPP}(l^k, u^k)$ so we can hope that a few dual simplex iterations starting from the optimal solution to $\mathsf{LPP}(l^k, u^k)$ would suffice to solve $\mathsf{LPP}(l^k, u)$. Usually, this is significantly faster than solving $\mathsf{LPP}(l^k, u)$ from scratch.

### 1.1.2 Modeling with MILP

MILP can clearly be used to model problems where the decision variables are discrete or indivisible such as the number of cars sold or the number of workers assigned to a task. However, MILP can also be used to model additional types of constraints such as implications or other logical conditions. A study of the types of constraints that can be modeled as MILPs began with Meyer [98, 99, 100, 101] and was continued by Jeroslow and Lowe [64, 66, 67, 68, 87]. They showed that the constraints that can be modeled as MILPs are essentially those of the form $x \in \bigcup_{i\in I} P_i \subset \mathbb{R}^n$, where $\{P_i\}_{i\in I}$ is a finite family of polyhedra with a special property that is satisfied if, for instance, all the polyhedra are bounded.

For example, if we wish to model the constraint $x \in \mathcal{Q}_4$ for

$$\mathcal{Q}_4 := \{x \in \mathbb{R}^4 : |x_1| + |x_2| \le 1,\, x_3 = x_4 = 0\} \cup \{x \in \mathbb{R}^4 : |x_2| + |x_3| \le 1,\, x_1 = x_4 = 0\} \cup \{x \in \mathbb{R}^4 : |x_3| + |x_4| \le 1,\, x_1 = x_2 = 0\} \quad (2)$$

we can use the MILP formulation given by

$$
\begin{aligned}
rx_1 + sx_2 &\leq 1 && \forall r \in \{-1,1\},\ s \in \{-1,1\} && \text{(3a)}\\
rx_2 + sx_3 &\leq 1 && \forall r \in \{-1,1\},\ s \in \{-1,1\} && \text{(3b)}\\
rx_3 + sx_4 &\leq 1 && \forall r \in \{-1,1\},\ s \in \{-1,1\} && \text{(3c)}\\
rx_1 &\leq z_1 && \forall r \in \{-1,1\} && \text{(3d)}\\
rx_2 &\leq z_1 + z_2 && \forall r \in \{-1,1\} && \text{(3e)}\\
rx_3 &\leq z_2 + z_3 && \forall r \in \{-1,1\} && \text{(3f)}\\
rx_4 &\leq z_3 && \forall r \in \{-1,1\} && \text{(3g)}\\
z_1 + z_2 + z_3 &= 1 && && \text{(3h)}\\
0 \leq z_j &\leq 1 && \forall j \in \{1,2,3\} && \text{(3i)}\\
z &\in \mathbb{Z}^3 && . && \text{(3j)}
\end{aligned}
$$

Of course, there are alternative MILP formulations for this constraint, which raises the question of what is a good. One desirable property of the formulation is for its size to be small. However, having the LP relaxation of a MILP be "similar" to the MILP is also a very desirable property. We explore these issues in the next two sections and then give a more practical example of modeling with MILP which is related to two chapters of this thesis.

#### *1.1.2.1 Quality of MILP Formulations*

If we want to maximize $\sum_{j=1}^{4} c_j x_j$ over all $x \in \mathcal{Q}_4$ for some $c \in \mathbb{R}^4$ we can solve the MILP given by $z_{\mathsf{MILPP}} := \max\{\sum_{j=1}^{4} c_j x_j \ :\ \text{(3a)–(3j)}\}$. As noted in Section 1.1.1, the performance of an LP based algorithm when solving this MILP will be highly dependent on how close $z_{\mathsf{LPP}} := \max\{\sum_{j=1}^{4} c_j x_j \ :\ \text{(3a)–(3i)}\}$ is to $z_{\mathsf{MILPP}}$. Ideally we would like $z_{\mathsf{MILPP}} = z_{\mathsf{LPP}}$ or at least $\delta := (z_{\mathsf{LPP}} - z_{\mathsf{MILPP}})/z_{\mathsf{MILPP}} \ll 1$ for all $c \in \mathbb{R}^4$. Unfortunately, we have that $\delta \geq 1$ for any $c \in \{-1,1\}^4$. In effect, any $(x,z)$ feasible for (3a)–(3j) has $\sum_{j=1}^{4} |x_j| \leq 1$ which implies $z_{\mathsf{MILPP}} = 1$ and, for $c \in \{-1,1\}^4$, $(x,z)$ given by $x_i = c_i/2$ for $i \in \{1,\ldots,4\}$ $z_1 = z_3 = 1/2$ and $z_2 = 0$ is feasible for (3a)–(3i) which implies $z_{\mathsf{LPP}} \geq 2$. However, we can

achieve $z_{\mathsf{MILPP}} = z_{\mathsf{LPP}}$ for all $c \in \mathbb{R}^4$ by adding to (3a)–(3j) the 16 inequalities given by

$$\sum_{i=1}^{4} r_i x_i \leq 1 \quad \forall r \in \{-1, 1\}^4. \tag{4}$$

The condition $z_{\mathsf{MILPP}} = z_{\mathsf{LPP}}$ for all $c \in \mathbb{R}^4$ is equivalent to asking for the projection of (3a)–(3i) onto the $x$ variables to be equal to the convex hull of $\mathcal{Q}_4$. A MILP formulation with this property is referred to as *sharp* by Jeroslow and Lowe, who also showed that it is the best we can ask from a MILP formulation if we only consider the original $x$ variables [67, 87].

But if we consider the integrality requirements on the $z$ variables of MILPP, there is an even stronger property. We can ask that every optimal solution to LPP should also be feasible for MILPP. This property is equivalent to requiring the extreme points of LPP to naturally comply with the integrality requirements of MILPP. When $x \in \bigcup_{i \in I} P_i$ is included in a larger problem that includes additional constraints this property is usually required to hold in the absence of these additional constraint and in this case the formulation is referred to as *locally ideal* [105, 106].

A locally ideal formulation is always sharp, but not vice versa. For example, the formulation of $x \in \mathcal{Q}_4$ given by (3a)–(3j) and (4) is sharp, but its LP relaxation has $x_1 = x_2 = z_1 = z_3 = 1/2$, $x_3 = x_4 = z_2 = 0$ as an extreme point and hence is not locally ideal.

#### *1.1.2.2 Extended Formulations*

Constructing good formulations of $x \in \bigcup_{i \in I} P_i$ using only the original $x$ variables and binary variables $z \in \{0, 1\}^{|I|}$ can sometimes require a large number of constraints. For example, let $\mathcal{Q}_n$ be the generalization of $\mathcal{Q}_4$ given by

$$\mathcal{Q}_n := \bigcup_{i=1}^{n-1} \left\{ x \in \mathbb{R}^n \,:\, |x_i| + |x_{i+1}| \leq 1,\, x_j = 0 \,\forall j \notin \{i, i+1\} \right\}. \tag{5}$$

A sharp MILP formulation of $x \in \mathcal{Q}_n$ is given by

$$\sum_{j=0}^{n} r_j x_j \leq 1 \qquad \forall s \in \{-1,1\}^n \tag{6a}$$

$$rx_1 \leq z_1 \qquad \forall r \in \{-1,1\} \tag{6b}$$

$$rx_j \leq z_j + z_{j-1} \qquad \forall i \in \{2,\ldots,n-1\},\ r \in \{-1,1\} \tag{6c}$$

$$rx_n \leq z_{n-1} \qquad \forall r \in \{-1,1\} \tag{6d}$$

$$\sum_{j=1}^{n-1} z_j = 1 \tag{6e}$$

$$z \in \{0,1\}^{n-1}. \tag{6f}$$

This formulation has $2^n + 2n + 1$ constraints besides the integrality requirements on $z$ and this number cannot be significantly reduced while preserving the sharpness property if we remain in the $(x, z)$ space. In effect, it is easy to show that constraints (6a) are facet defining for both $\text{conv}(\mathcal{Q}_n)$ and $\text{conv}\left(\left\{(x,z) \in \mathbb{R}^{2n} : \text{(6a)–(6f)}\right\}\right)$.

In contrast, if we allow the use of auxiliary variables we can construct formulations with a significantly smaller number of constraints. For example, by using a well known formulation trick we can construct a sharp MILP formulation of $x \in \mathcal{Q}_n$ with $4n + 2$ constraints given by

$$rx_j \leq y_j \qquad \forall j \in \{1,\ldots,n\},\ r \in \{-1,1\} \tag{7a}$$

$$\sum_{j=0}^{n} y_j \leq 1 \tag{7b}$$

$$rx_1 \leq z_1 \qquad \forall r \in \{-1,1\} \tag{7c}$$

$$rx_j \leq z_j + z_{j-1} \qquad \forall j \in \{2,\ldots,n-1\},\ r \in \{-1,1\} \tag{7d}$$

$$rx_n \leq z_{n-1} \qquad \forall r \in \{-1,1\} \tag{7e}$$

$$\sum_{j=1}^{n-1} z_j = 1 \tag{7f}$$

$$z \in \{0,1\}^{n-1}. \tag{7g}$$

However, this formulation is still not locally ideal for $n = 4$ since its LP relaxation has $x_1 = x_2 = y_1 = y_2 = z_1 = z_3 = 1/2$, $x_3 = x_4 = z_2 = 0$ as an extreme point. Fortunately,

the following theorem by Balas shows us how to construct a locally ideal formulation for a disjunctive set such as $\mathcal{Q}_n$.

**Theorem 1.1** (Balas [9])**.** *Let*

$$\mathcal{D} := \bigcup_{i=1}^{r} \{x \in \mathbb{R}^n \,:\, A^i x \leq b^i\} \tag{8}$$

*for* $A^i \in \mathbb{R}^{m_i \times n}$, $b^i \in \mathbb{R}^{m_i}$ *such that*

$$\{x \in \mathbb{R}^n \,:\, A^i x = 0\} = \{x \in \mathbb{R}^n \,:\, A^j x = 0\} \quad \forall i, j \in \{1, \ldots, r\}. \tag{9}$$

*A locally ideal formulation with* $r(n+1)$ *variables and* $r + n + 1 + \sum_{i=1}^{r} m_i$ *constraints for* $x \in \mathcal{D}$ *is given by*

$$A^i x^i \leq z_i b^i \qquad \forall i \in \{1, \ldots, r\} \tag{10a}$$

$$\sum_{i=1}^{r} x^i = x \tag{10b}$$

$$\sum_{i=1}^{r} z_i = 1 \tag{10c}$$

$$z \in \{0,1\}^r. \tag{10d}$$

The auxiliary variables used in formulation (10) correspond to a copy of variables $x$ for each polyhedron on the right hand side of (8) and binary variables that indicate which one of these polyhedrons is selected. Condition (9) simply require that all polyhedrons on the right hand side of (8) should have the same directions of unboundedness.

Using Theorem 1.1 we obtain the locally ideal formulation of $x \in \mathcal{Q}_n$ with $4n-3$ variables and $2n$ constraints given by

$$r x_j^j + s x_{j+1}^j \leq z_j \qquad \forall j \in \{1, \ldots, n-1\},\, r \in \{-1, 1\},\, s \in \{-1, 1\} \tag{11a}$$

$$x_1 = x_1^1 \tag{11b}$$

$$x_j = x_j^j + x_j^{j-1} \quad \forall j \in \{2, \ldots, n-1\},\, r \in \{-1, 1\} \tag{11c}$$

$$x_n = x_n^{n-1} \tag{11d}$$

$$\sum_{j=1}^{n-1} z_j = 1 \tag{11e}$$

$$z \in \{0,1\}^{n-1}. \tag{11f}$$

MILP formulations that use additional auxiliary variables are usually referred to as *extended* formulations and have been used to construct strong compact formulations for many problems (for example, see [34] and the references within). The idea of using auxiliary variables to exploit the favorable properties of projection [10] have also been used to construct compact polyhedral approximations of convex sets [15].

#### *1.1.2.3 Example: MILP Models for Piecewise Linear Interpolations*

We now show how MILP can be used to model an approximation of a nonlinear function. We first detail the construction for univariate functions and then sketch it for bivariate functions.

A common way to approximate a univariate non-linear function $g : [l, u] \to \mathbb{R}$ is to subdivide $[l, u]$ into subintervals of the form $[d_{k-1}, d_k]$ for breakpoints $l = d_0 < d_1 < \ldots < d_{K-1} < d_K = u$ and construct the piecewise linear interpolation given by

$$f(x) := \Big\{ g(d_{k-1}) + \tfrac{g(d_k)-g(d_{k-1})}{d_k-d_{k-1}}(x - d_{k-1}) \quad x \in [d_{k-1}, d_k]. \tag{12}$$

The resulting function $f$ is also referred to as the piecewise Lagrange polynomial of degree 1 or the linear spline interpolation of $g$ (e.g. [102, 110]) and is the only function that is continuous on $[l, u]$, affine on each subinterval $[d_{k-1}, d_k]$ and agrees with $g$ on all of the break points. This procedure is illustrated in Figure 2, where $g$ is given by the dashed curve and $f$ by the three thick line segments.

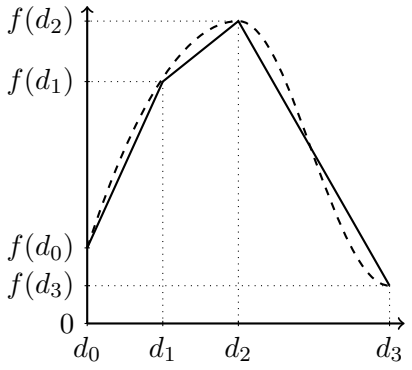



**Figure 2:** Univariate Interpolated Piecewise Linear Function.

For the resulting piecewise linear function $f : [l, u] \to \mathbb{R}$ we have that $f(x) = z$ is

equivalent to $(x, z) \in \mathcal{Q}_k$ for

$$\mathcal{Q}_k := \bigcup_{k=1}^{K} \left\{ (x,z) \in \mathbb{R}^2 \,:\, z = f(d_{k-1}) + \frac{f(d_k) - f(d_{k-1})}{d_k - d_{k-1}} \left(x - d_{k-1}\right),\, d_{k-1} \le x \le d_k \right\},$$

which we can model as the MILP (see for example section I.1.4 of [104]) given by

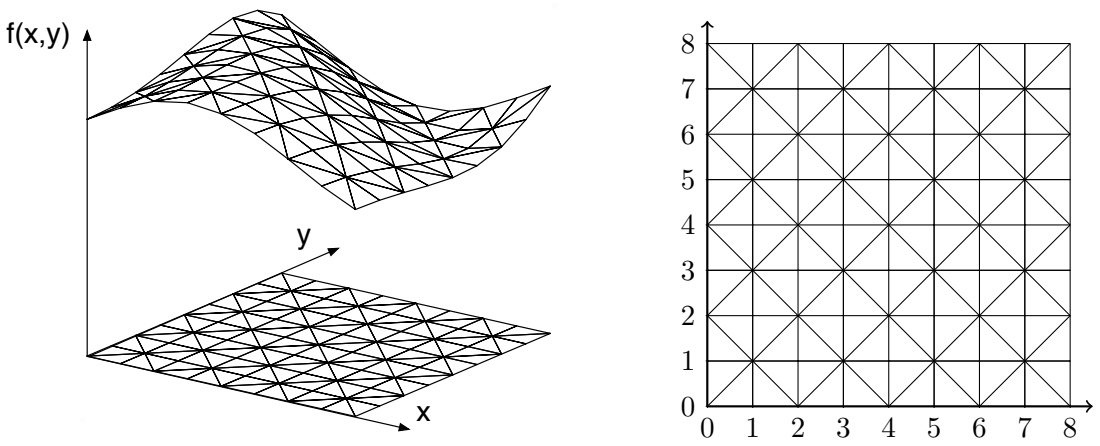
$$\sum_{k=0}^{K} d_k \lambda_k = x, \quad \sum_{k=0}^{K} f(d_k)\lambda_k = z, \quad \sum_{k=0}^{K} \lambda_k = 1, \quad \lambda_k \ge 0 \quad \forall k \in \{0, \dots, K\} \tag{13a}$$

$$\lambda_0 \le y_1, \quad \lambda_K \le y_K, \quad \lambda_k \le (y_k + y_{k+1}) \quad \forall k \in \{1, \dots, K-1\} \tag{13b}$$

$$\sum_{k=1}^{K} y_k = 1, \quad y_k \in \{0,1\} \quad \forall k \in \{1, \dots, K\}. \tag{13c}$$

Constraints (13a) describe $(x, z) = (x, f(x))$ as the convex combination of points $(d_k, f(d_k))_{k=0}^{K}$. Constraints (13b)–(13c) assures the validity of this convex combination by enforcing the combinatorial requirement on $(\lambda_k)_{k=0}^{K}$ that at most two $\lambda_k$'s can be non-zero and that if two $\lambda_k$'s are non-zero their indices must be adjacent (i.e. if $\lambda_i > 0$ and $\lambda_j > 0$ then $j = i+1$). This combinatorial requirement is known as *SOS2 constraints* [12].

Formulation (13) is sharp, but it not locally ideal [36, 72, 105]. However, we can again use Corollary 2.1.2 of [9] to get an extended formulation of $(x, z) \in \mathcal{Q}_k$ that is locally ideal. This formulation is described in Section 4.3.3 of Chapter 4.




(a) Bivariate Piecewise Linear Function. (b) Triangulation

**Figure 3:** Interpolating Bivariate Functions.

A similar procedure can be used for multivariate functions. For example, for a bivariate function $g : [l, u]^2 \to \mathbb{R}$ we can triangulate (e.g. [126, 142]) $[l, u]^2$ by subdividing it into a

finite number of simplices and construct a linear spline interpolation $f$ that is continuous, affine on each of the simplices and agrees with $g$ on all the vertices of the triangulation (e.g. [75]). This procedure is illustrated in Figure 3 where Figure 3(a) shows the piecewise linear function resulting from the interpolation of $g(x, y) := \sin\left(x/2 + (y/5)^2\right)$ over the triangulation of $[0, 8]^2$ given in Figure 3(b). The vertices of this triangulation are the points in $\{0, 1, \ldots, 8\}^2$.

For the resulting piecewise linear function $f : [l, u]^2 \to \mathbb{R}$ we can use an extension of formulation (13) for multivariate piecewise linear functions given in [82, 96, 127]. This extension describes $(x, y, z) = (x, y, f(x, y))$ as the convex combination of points $(v_1, v_2, f(v_1, v_2))$ for each vertex $(v_1, v_2)$ of the triangulation used to define $f$. Validity of this convex combination is assured by a combinatorial constraint that is the analog of SOS2 constraints for triangulations. This extension is described in detail in Section 3.4 of Chapter 3 and Section 4.3.2 of Chapter 4. This formulation is again sharp and not locally ideal, but we can also use Corollary 2.1.2 of [9] to construct a locally ideal formulation that is described in Section 4.3.3 of Chapter 4.

## *1.2 Dissertation Overview*

In this thesis we study how to solve some nonconvex optimization problems by using methods that capitalize on the success of LP based solvers for MILP. A common aspect of our solution approaches is the use, development and analysis of small but strong extended LP/MILP formulations and approximations.

In Chapter 2 work we develop a LP based branch-and-bound algorithm for Mixed Integer Conic Quadratic programs (MICQP). The algorithm is based on an extended or lifted polyhedral relaxation of conic quadratic constraints introduced by Ben-Tal and Nemirovski [15]. The algorithm is different from other LP based branch-and-bound algorithms for mixed integer nonlinear programs in that, it is not based on cuts from gradient inequalities and it sometimes branches on integer feasible solutions. The algorithm is conceptually valid for any mixed integer nonlinear programming problem whose continuous relaxation is a convex optimization problem, but we only test it on MICQP problems as we are only aware of the

existence of an efficient lifted polyhedral relaxation for this case. We test the algorithm on a series of portfolio optimization problems and show that it significantly outperforms commercial and open source solvers based on both linear and nonlinear relaxations.

In Chapter 3 we study the modeling of a class of disjunctive constraints with a logarithmic number of binary variables and constraints. Many combinatorial constraints over continuous variables such as SOS1 and SOS2 constraints [12] can be interpreted as disjunctive constraints that restrict the variables to lie in the union of a finite number of specially structured polyhedra. Known mixed integer binary formulations for these constraints have a number of binary variables and extra constraints linear in the number of polyhedra. We give sufficient conditions for constructing formulations for these constraints with a number of binary variables and extra constraints logarithmic in the number of polyhedra. Using these conditions we introduce mixed integer binary formulations for SOS1 and SOS2 constraints that have a number of binary variables and extra constraints logarithmic in the number of continuous variables. We also introduce the first mixed integer binary formulations for piecewise linear functions of one and two variables that use a number of binary variables and extra constraints logarithmic in the number of linear pieces of the functions. We prove that the new formulations for piecewise linear functions have favorable tightness properties and present computational results showing that they can significantly outperform other mixed integer binary formulations.

In Chapter 4 we study the modeling of non-convex piecewise linear functions as MILPs. We review several new and existing MILP formulations for continuous piecewise linear functions including the one introduced in Chapter 3. We pay special attention to the modeling of multivariate non-separable functions such as the one depicted in Figure 3(a). We compare the formulations with respect to their theoretical properties and their relative computational performance. In particular, we study which formulations are sharp and which are locally ideal. In addition, we study the extension of the formulations to lower semicontinuous piecewise linear functions through a general technique and ad-hoc approaches.

Finally, in Chapter 5 we study the strength of MILP formulations for LPs with Probabilistic Constraints. We first study the strength of existing MILP formulations that only

considers one row of the probabilistic constraint at a time. We then introduce an extended formulation that considers more than one row of the constraint at a time and use it to computationally compare the relative strength between formulations that consider one and two rows at a time.

# CHAPTER II

# A LIFTED LINEAR PROGRAMMING BRANCH-AND-BOUND ALGORITHM FOR MIXED INTEGER CONIC QUADRATIC PROGRAMS

## *2.1 Introduction*

This chapter deals with the development of an algorithm for the class of mixed integer nonlinear programming (MINLP) problems known as mixed integer conic quadratic programming problems. This class of problems arises from adding integrality requirements to conic quadratic programming problems [86], and is used to model several applications from engineering and finance [3, 13, 31, 86, 85, 23, 32, 94, 21]. Conic quadratic programming problems are also known as second order cone programming problems, and together with semidefinite and linear programming (LP) problems are special cases of the more general conic programming problems [14]. For ease of exposition, we will refer to conic quadratic and mixed integer conic quadratic programming problems simply as conic programming (CP) and mixed integer conic programming (MICP) problems respectively.

We are interested in solving MICP problems of the form

$$z_{\mathsf{MICPP}} := \max_{x,y} \quad cx + dy \tag{14}$$

$$s.t.$$

$$Dx + Ey \leq f \tag{15}$$

$$(x, y) \in \mathcal{CC}_i \qquad i \in \mathcal{I} \tag{16}$$

$$(x, y) \in \mathbb{R}^{n+p} \tag{17}$$

$$x \in \mathbb{Z}^n \tag{18}$$

where $c \in \mathbb{R}^n$, $d \in \mathbb{R}^p$, $D \in \mathbb{R}^{m \times n}$, $E \in \mathbb{R}^{m \times p}$, $f \in \mathbb{R}^m$, $\mathcal{I} \subset \mathbb{Z}_+$, $|\mathcal{I}| < \infty$ and for each $i \in \mathcal{I}$ $(x, y) \in \mathcal{CC}_i$ is a conic constraint of the form

$$(x, y) \in \mathcal{CC} := \{(x, y) \in \mathbb{R}^{n+p} \, : \, ||Ax + By + \delta||_2 \leq ax + by + \delta_0\} \tag{19}$$

for some $r \in \mathbb{Z}_+$, $A \in \mathbb{R}^{r\times n}$, $B \in \mathbb{R}^{r\times p}$, $\delta \in \mathbb{R}^r$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}^p$, $\delta_0 \in \mathbb{R}$ and where $||\cdot||_2$ is the Euclidean norm and for two vectors $u, v \in \mathbb{R}^k$ of the same dimension $uv$ denotes the inner product $\sum_{i=1}^k u_i v_i$. We denote the MICP problem given by (14)–(18) as MICPP and its CP relaxation given by (14)–(17) as CPP.

MICPP includes many portfolio optimization problems (see for example [13], [31], [86] and [85]). A specific example is the portfolio optimization problem with cardinality constraints (see for example [23], [32], [94] and [21]) which can be formulated as

$$\max_{x,y} \quad \bar{a}y \tag{20}$$

$$s.t.$$

$$||Q^{1/2}y||_2 \leq \sigma \tag{21}$$

$$\sum_{j=1}^{n} y_j = 1 \tag{22}$$

$$y_j \leq x_j \qquad \forall j \in \{1,\ldots,n\} \tag{23}$$

$$\sum_{j=1}^{n} x_j \leq K \tag{24}$$

$$x \in \{0,1\}^n \tag{25}$$

$$y \in \mathbb{R}^n_+, \tag{26}$$

where $n$ is the number of assets available, $y$ indicates the fraction of the portfolio invested in each asset, $\bar{a} \in \mathbb{R}^n$ is the vector of expected returns of the stocks, $Q^{1/2}$ is the positive semidefinite square root of the covariance matrix of the returns of the stocks, $\sigma$ is the maximum allowed risk and $K < n$ is the maximum number of stocks that can be held in the portfolio. Objective (20) is to maximize the expected return of the portfolio, constraint (21) limits the risk of the portfolio, and constraints (23)–(25) limit the number of stocks that can be held in the portfolio to $K$. Finally, constraints (22) and (26) force the investment of the entire budget in the portfolio.

Most algorithms for solving MICP problems (and in general for solving MINLP problems whose continuous relaxations are convex optimization problems) can be classified into two major groups depending on what type of continuous relaxations they use (see for example

[28] and [56]).

The first group only uses the nonlinear relaxation CPP in a branch-and-bound procedure [29, 59, 84, 123]. This procedure is the direct analog of the LP based branch-and-bound procedure for mixed integer linear programming (MILP) problems and is the basis for the MICP solver in CPLEX 9.0 and 10.0 [62] and the I-BB solver in Bonmin [28]. We refer to these algorithms as *NLP based branch-and-bound algorithms*.

The second group is related to domain decomposition techniques in global optimization (see for example Section 7 of [60] and [124]) and uses polyhedral relaxations of the nonlinear constraints of MICPP, possibly together with the nonlinear relaxation CPP. These polyhedral relaxations are usually updated after solving an associated MILP problem or inside a branch-and-bound procedure. Additionally the nonlinear relaxation of MICPP is sporadically solved to obtain integer feasible solutions, to improve the polyhedral relaxations, to fathom nodes in a branch-and-bound procedure or as a local search procedure. Some of the algorithms in this group include outer approximation [49, 50], generalized Benders decomposition [53], LP/NLP-based branch-and-bound [111] and the extended cutting plane method [136, 137]. This approach is the basis for the I-OA, I-QG and I-Hyb solvers in Bonmin [28] and the MINLP solver FilMINT [1]. We refer to these algorithms as *polyhedral relaxation based algorithms*.

For algorithms in the second group to perform efficiently, it is essential to have polyhedral relaxations of the nonlinear constraints that are both tight and have few constraints. To the best of our knowledge, the polyhedral relaxations used by all the algorithms proposed so far are based on gradient inequalities for the nonlinear constraints. This approach yields a polyhedral relaxation which is constructed in the space of the original variables of the problem. The difficulty with these types of polyhedral relaxations is that they can require an unmanageable number of inequalities to yield tight approximations of the nonlinear constraints. In particular, it is known that obtaining a tight polyhedral approximation of the Euclidean ball without using extra variables requires an exponential number of inequalities [11]. To try to resolve this issue, current polyhedral based algorithms generate the relaxations dynamically.

In the context of CP problems, an alternative polyhedral relaxation that is not based on gradient inequalities was introduced in 1999 by Ben-Tal and Nemirovski [15]. This approach uses the projection of a higher dimensional or *lifted* polyhedral set to generate a polyhedral relaxation of a conic quadratic constraint of the form $\mathcal{CC}$. By exploiting the fact that projection can significantly multiply the number of facets of a polyhedron, this approach constructs a relaxation that is "efficient" in the sense that it is very tight and yet it is defined using a relatively small number of constraints and extra variables. The relaxation of Ben-Tal and Nemirovski has been further studied by Glineur [54] who also tested it computationally on continuous CP problems. These tests showed that solving the original CP problem with state of the art interior point solvers was usually much faster than solving the polyhedral relaxation.

Although the polyhedral relaxation of [15] and [54] might not be practical for solving purely continuous CP problems, it could be useful for polyhedral relaxation based algorithms for solving MICP problems. In particular, solving the polyhedral relaxation in a branch-and-bound procedure instead of the original CP relaxations could benefit from the "warm start" capabilities of the simplex algorithm for LP problems and the various integer programming enhancements such as cutting planes and preprocessing that are available in commercial MILP solvers. The objective of this paper is to develop such a polyhedral relaxation based algorithm and to demonstrate that this approach can significantly outperform other methods. The algorithm is conceptually valid for any MINLP problem whose continuous relaxation is a convex optimization problem, but we only test it on MICP problems as we are only aware of the existence of an efficient lifted polyhedral relaxation for this case.

The remainder of the chapter is organized as follows. In Section 2.2 we introduce a branch-and-bound algorithm based on a lifted polyhedral relaxation. In Section 2.3 we describe the polyhedral relaxation of [15] and [54] we use in our test. Then, in Section 2.4 we present computational results which demonstrate that the algorithm significantly outperforms other methods. Finally, in Section 2.5 we give some conclusions and possible future work in this area.

## *2.2 A Branch-and-Bound Algorithm for Convex MINLP*

We describe the algorithm for MINLP problems whose continuous relaxations are convex programs. These problems are usually referred to as convex MINLPs [59, 111, 123, 136]. The algorithm is somewhat similar to other polyhedral relaxation algorithms and in particular to enhanced versions of the LP/NLP-based branch-and-bound algorithm such as Bonmin's I-Hyb solver and FilMINT. The main differences between the proposed algorithm and existing polyhedral relaxation based algorithms for convex MINLPs are that:

(i) it is based on a lifted polyhedral relaxation instead of one constructed using gradient inequalities,

(ii) it does not update the relaxation using gradient inequalities, and

(iii) it will sometimes branch on integer feasible solutions.

The MINLP we solve is of the form

$$z_{\mathsf{MINLPP}} := \max_{x,y} \quad cx + dy \tag{27}$$

$$s.t.$$

$$(x, y) \in \mathcal{C} \subset \mathbb{R}^{n+p} \tag{28}$$

$$x \in \mathbb{Z}^n \tag{29}$$

where $\mathcal{C}$ is a compact convex set. We denote the problem given by (27)–(29) by MINLPP. We also denote by NLPP the continuous relaxation of MINLPP given by (27)–(28) and we assume for simplicity that MINLPP is feasible. Note that MINLPP includes all MINLPs for which their continuous relaxation is a convex optimization problem, as a problem with nonlinear concave (we are maximizing) objective functions can always be converted to one with a linear objective function.

We further assume that we have a lifted polyhedral relaxation of the convex set $\mathcal{C}$. In other words there exists $q \in \mathbb{Z}_+$ and a bounded polyhedron $\mathcal{P} \subset \mathbb{R}^{n+p+q}$ such that

$$\mathcal{C} \subset \{(x, y) \in \mathbb{R}^{n+p} \,:\, \exists\, v \in \mathbb{R}^q \text{ s.t. } (x, y, v) \in \mathcal{P}\}.$$

Thus we have the lifted linear programming relaxation of MINLPP given by

$$z_{\mathsf{LLPP}} := \max_{x,y,v} \quad cx + dy \tag{30}$$

$$s.t.$$

$$(x, y, v) \in \mathcal{P}, \tag{31}$$

which we denote by LLPP.

Note that we could very well choose $q = 0$ in the construction of LLPP, but as we will discuss in Section 2.3, the key idea for the effectiveness of our algorithm is the use of a tight lifted LP relaxation that requires $q > 0$.

The final problem we use in the algorithm is defined for any $\hat{x} \in \mathbb{Z}^n$ as

$$z_{\mathsf{NLPP}(\hat{x})} := \max_{y} \quad c\hat{x} + dy$$

$$s.t.$$

$$(\hat{x}, y) \in \mathcal{C} \subset \mathbb{R}^{n+p}.$$

We denote this problem by $\mathsf{NLPP}(\hat{x})$.

We use these auxiliary problems to construct a branch-and-bound algorithm for solving MINLPP as follows. For any $(l^k, u^k) \in \mathbb{Z}^{2n}$ we denote by $\mathsf{LLPP}(l^k, u^k)$ and $\mathsf{NLPP}(l^k, u^k)$ the problems obtained by adding constraints $l^k \leq x \leq u^k$ to LLPP and NLPP respectively. We also adopt the convention that a node $k$ in a branch-and-bound tree is defined by some $(l^k, u^k, \mathrm{UB}^k) \in \mathbb{Z}^{2n} \times (\mathbb{R} \cup \{+\infty\})$ where $(l^k, u^k)$ are the bounds defining the node and $\mathrm{UB}^k$ is an upper bound on $z_{\mathsf{NLPP}(l^k,u^k)}$. Furthermore, we denote by LB the global lower bound on $z_{\mathsf{MINLPP}}$ and by $\mathcal{H}$ the set of active branch-and-bound nodes. We give in Figure 4 a lifted LP branch-and-bound algorithm for solving MINLPP.

A pure NLP based branch-and-bound algorithm solves $\mathsf{NLPP}(l^k, u^k)$ at each node $k$ of the branch-and-bound tree. The idea of the lifted LP branch-and-bound algorithm of Figure 4 is to replace each call to $\mathsf{NLPP}(l^k, u^k)$ in an NLP based branch-and-bound algorithm by a call to $\mathsf{LLPP}(l^k, u^k)$. After this replacement special care has to be taken when fathoming by integrality as an integer feasible solution to $\mathsf{LLPP}(l^k, u^k)$ is not necessarily an integer feasible

Set global lower bound LB $:= -\infty$.
Set $l_i^0 := -\infty$, $u_i^0 := +\infty$ for all $i \in \{1, \ldots, n\}$.
Set $\mathrm{UB}^0 = +\infty$.
Set node list $\mathcal{H} := \{(l^0, u^0, \mathrm{UB}^0)\}$.
**while** $\mathcal{H} \neq \emptyset$ **do**
    Select and remove a node $(l^k, u^k, \mathrm{UB}^k) \in \mathcal{H}$.
    Solve $\mathsf{LLPP}(l^k, u^k)$.
    **if** $\mathsf{LLPP}(l^k, u^k)$ *is feasible* **and** $z_{\mathsf{LLPP}(l^k,u^k)} > \mathrm{LB}$ **then**
        Let $(\hat{x}^k, \hat{y}^k)$ be the optimal solution to $\mathsf{LLPP}(l^k, u^k)$.
        **if** $\hat{x}^k \in \mathbb{Z}^n$ **then**
            Solve $\mathsf{NLPP}(\hat{x}^k)$.
            **if** $\mathsf{NLPP}(\hat{x}^k)$ *is feasible* **and** $z_{\mathsf{NLPP}(\hat{x}^k)} > \mathrm{LB}$ **then**
                LB $:= z_{\mathsf{NLPP}(\hat{x}^k)}$.
            **end**
            **if** $l^k \neq u^k$ **and** $z_{\mathsf{LLPP}(l^k,u^k)} > \mathrm{LB}$ **then**
                Solve $\mathsf{NLPP}(l^k, u^k)$.
                **if** $\mathsf{NLPP}(l^k, u^k)$ *is feasible* **and** $z_{\mathsf{NLPP}(l^k,u^k)} > \mathrm{LB}$ **then**
                    Let $(\tilde{x}^k, \tilde{y}^k)$ be the optimal solution to $\mathsf{NLPP}(l^k, u^k)$.
                    **if** $\tilde{x}^k \in \mathbb{Z}^n$ **then** `/* Fathom by Integrality */`
                        LB $:= z_{\mathsf{NLPP}(l^k,u^k)}$.
                    **else** `/* Branch on` $\tilde{x}^k$ `*/`
                        Pick $i_0$ in $\{i \in \{1, \ldots, n\} : \tilde{x}_i^k \notin \mathbb{Z}\}$.
                        Let $l_i = l_i^k$, $u_i = u_i^k$ for all $i \in \{1, \ldots, n\} \setminus \{i_o\}$.
                        Let $u_{i_0} = \lfloor \tilde{x}_{i_0}^k \rfloor$, $l_{i_0} = \lfloor \tilde{x}_{i_0}^k \rfloor + 1$.
                        $\mathcal{H} := \mathcal{H} \cup \{(l^k, u, z_{\mathsf{NLPP}(l^k,u^k)}), (l, u^k, z_{\mathsf{NLPP}(l^k,u^k)})\}$
                    **end**
                **end**
            **end**
        **else** `/* Branch on` $\hat{x}^k$ `*/`
            Pick $i_0$ in $\{i \in \{1, \ldots, n\} : \hat{x}_i^k \notin \mathbb{Z}\}$.
            Let $l_i = l_i^k$, $u_i = u_i^k$ for all $i \in \{1, \ldots, n\} \setminus \{i_o\}$.
            Let $u_{i_0} = \lfloor \hat{x}_{i_0}^k \rfloor$, $l_{i_0} = \lfloor \hat{x}_{i_0}^k \rfloor + 1$.
            $\mathcal{H} := \mathcal{H} \cup \{(l^k, u, z_{\mathsf{LLPP}(l^k,u^k)}), (l, u^k, z_{\mathsf{LLPP}(l^k,u^k)})\}$
        **end**
    **end**
    Remove every node $(l^k, u^k, \mathrm{UB}^k) \in \mathcal{H}$ such that $\mathrm{UB}^k \leq \mathrm{LB}$.
**end**

**Figure 4:** A Lifted LP Branch-and-Bound Algorithm.

solution to $\mathsf{NLPP}(l^k, u^k)$. This is handled by the algorithm in lines 11–28. The first step is to solve $\mathsf{NLPP}(\hat{x}^k)$ to attempt to correct an integer feasible solution $(\hat{x}^k, \hat{y}^k)$ to $\mathsf{LLPP}(l^k, u^k)$ into an integer feasible solution to $\mathsf{NLPP}(l^k, u^k)$. If the correction is successful and $z_{\mathsf{NLPP}(\hat{x}^k)} >$ LB we can update LB. This step is carried out in lines 11–14 of the algorithm. Another complication arises when the optimal solution to $\mathsf{LLPP}(l^k, u^k)$ is integer feasible, but $l^k \neq u^k$. The problem in this case is that integer optimal solutions to $\mathsf{LLPP}(l^k, u^k)$ and $\mathsf{NLPP}(\hat{x}^k)$ may not be solutions to $\mathsf{MINLPP}(l^k, u^k)$. In fact, in this case, it is possible for $\mathsf{NLPP}(\hat{x}^k)$ to be infeasible and for $\mathsf{MINLPP}(l^k, u^k)$ to be feasible. To resolve this issue, the algorithm of Figure 4 solves $\mathsf{NLPP}(l^k, u^k)$ to process the node in the same way it would be processed in an NLP based branch-and-bound algorithm for MINLPP. This last step is carried out in lines 15–28.

Note that, in lines 21–26, the algorithm is effectively branching on a variable $x_i$ such that $\hat{x}_i^k$ is integer but for which $l_i^k < u_i^k$. Branching on integer feasible variables is sometimes used in MILP (it can be used for example to find alternative optimal solutions) and global optimization (see for example [124]), but to the best of our knowledge it has never been used in the context of polyhedral relaxation based algorithms for convex MINLPs.

We show the correctness of the algorithm in the following proposition.

**Proposition 2.1.** *For any polyhedral relaxation* LLPP *of* NLPP *using a bounded polyhedron* $\mathcal{P}$*, the lifted LP branch-and-bound algorithm of Figure 4 terminates with* LB *equal to the optimal objective value of* MINLPP*.*

*Proof.* Finiteness of the algorithm is direct from the fact that $\mathcal{P}$ is bounded. However after branching in lines 21–26, solution $(\hat{x}^k, \hat{y}^k)$ could be repeated in one of the newly created nodes, which could cause $(\hat{x}^k, \hat{y}^k)$ to be generated again in several nodes. This can only happen a finite number of times though, as the branching will eventually cause $l^k = u^k$ or $\mathsf{LLPP}(l^k, u^k)$ will become infeasible.

All that remains to prove is that the sub-tree rooted at a fathomed node cannot contain an integer feasible solution to MINLPP which has an objective value strictly larger than the current incumbent integer solution. The algorithm fathoms a node only in lines 8, 15,

17 and 19. In line 8, the node is fathomed if $\mathsf{LLPP}(l^k, u^k)$ is infeasible or if $z_{\mathsf{LLPP}(l^k,u^k)} \leq$ LB. Because $\mathsf{LLPP}(l^k, u^k)$ is a relaxation of $\mathsf{NLPP}(l^k, u^k)$ we have that infeasibility of $\mathsf{LLPP}(l^k, u^k)$ implies infeasibility of $\mathsf{NLPP}(l^k, u^k)$ and $z_{\mathsf{NLPP}(l^k,u^k)} \leq z_{\mathsf{LLPP}(l^k,u^k)}$, hence in both cases we have that the sub-tree rooted at node $(l^k, u^k)$ cannot contain an integer feasible solution strictly better than the incumbent. In line 15, the node is fathomed if $l^k = u^k$ or if $z_{\mathsf{LLPP}(l^k,u^k)} \leq$ LB. In the first case, $\mathsf{NLPP}(l^k, u^k) = \mathsf{NLPP}(\hat{x}^k)$ and hence processing node $k$ is correctly done by lines 12–14. In the second case, the node is correctly fathomed for the same reasons for correctness in line 8. In line 17, the node is fathomed if $\mathsf{NLPP}(l^k, u^k)$ is infeasible or if $z_{\mathsf{NLPP}(l^k,u^k)} \leq$ LB, in either case the sub-tree rooted at the fathomed node cannot contain a integer feasible solution strictly better that the incumbent. Finally, in line 19 the node is fathomed because solution $(\tilde{x}^k, \tilde{y}^k)$ to $\mathsf{NLPP}(l^k, u^k)$ is integer feasible and hence it is the best integer feasible solution that can be found at the sub-tree rooted at the fathomed node. □

We note that, as in other branch-and-bound algorithms, at any point in the execution of the algorithm we have a lower bound of $z_{\mathsf{MINLPP}}$ given by LB and an upper bound given by $\max\{\mathrm{UB}^k : (l^k, u^k, \mathrm{UB}^k) \in \mathcal{H}\}$. This can be used for early termination of the algorithm given a target optimality gap.

### *2.3 Lifted Polyhedral Relaxations*

The key idea for the effectiveness of the lifted LP branch-and-bound algorithm is the use of a lifted polyhedral relaxation ($q > 0$) for the construction of $\mathsf{LLPP}$. For the algorithm to be effective we need $\mathsf{NLPP}(l^k, u^k)$ to be called in as few nodes as possible, so we need $\mathsf{LLPP}$ to be a tight approximation of $\mathsf{NLPP}$. On the other hand we need to solve $\mathsf{LLPP}(l^k, u^k)$ quickly, which requires the polyhedral relaxation to have relatively few constraints and extra variables. The problem is that using a relaxation with $q = 0$, such as those constructed using gradient inequalities, can require a polyhedron $\mathcal{P}$ with an exponential number of facets to approximate the convex set $\mathcal{C}$ tightly. In fact, it is known (see for example [11]) that for any $\varepsilon > 0$ approximating the $d$-dimensional unit euclidean ball $\mathcal{B}^d$ with a polyhedron $\mathcal{P} \subset \mathbb{R}^d$ such that $\mathcal{B}^d \subset \mathcal{P} \subset (1+\varepsilon)\mathcal{B}^d$ requires $\mathcal{P}$ to have at least $\exp(d/(2(1+\varepsilon)^2))$ facets.

However, in many instances, only a few inequalities are needed to optimize over a convex set to a given accuracy. Therefore, current polyhedral relaxation based algorithms do not use a fixed polyhedral relaxation of $\mathcal{C}$ and instead dynamically refine the relaxation as needed.

On the other hand, when we allow for a polyhedron $\mathcal{P}$ in a higher dimensional space we can take advantage of the fact that a lifted polyhedron with a polynomial number of constraints and extra variables can have the same effect as a polyhedron in the original space with an exponential number of facets. Exploiting this property, it is sometimes possible to have a tight lifted polyhedral relaxation of $\mathcal{C}$ that can be described by a reasonable number of inequalities and extra variables. [15] introduced such a lifted polyhedral relaxation for MICP problems. We now give a compact description of the version of the lifted polyhedral relaxation of [15] and [54] we use in this study.

We start by noting that a set $\mathcal{CC}$ given by (19) can be written as

$$\begin{aligned}\mathcal{CC} = \{(x,y) \in \mathbb{R}^{n+p} \,:\, \exists (z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \quad \text{s.t.} \quad & Ax + By + \delta = z, \\ & ax + by + \delta_0 = z_0, \\ & (z_0, z) \in \mathcal{L}^r\}\end{aligned}$$

where $\mathcal{L}^r$ is the $(r+1)$-dimensional Lorentz cone given by

$$\mathcal{L}^r := \{(z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \,:\, \sum_{k=1}^{r} z_k^2 \leq z_0^2\}.$$

Hence a polyhedral relaxation of $\mathcal{L}^r$ induces a polyhedral relaxation of $\mathcal{CC}$. Then, for a given tightness parameter $\varepsilon > 0$ we want to construct a polyhedron $\mathcal{L}^r_\varepsilon$ such that

$$\mathcal{L}^r \subsetneq \mathcal{L}^r_\varepsilon \subsetneq \{(z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \,:\, ||z||_2 \leq (1+\varepsilon) z_0\}. \tag{32}$$

To describe this polyhedral relaxation of $\mathcal{L}^r$ we assume at first that $r = 2^p$ for some $p \in \mathbb{Z}_+$. We then begin by grouping variables $z$ in $\mathcal{L}^r$ into $k = r/2$ pairs and associate a new variable $\rho_k$ for the $k$th pair. We can then rewrite $\mathcal{L}^r$ as

$$\begin{aligned}\mathcal{L}^r = \{(z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \,:\, & \exists \rho \in \mathbb{R}^{r/2} \quad \text{s.t.} \\ & \sum_{k=1}^{r/2} \rho_k^2 \leq z_0^2 \\ & z_{2k-1}^2 + z_{2k}^2 \leq \rho_k^2 \text{ for } k \in \{1, \ldots, r/2\}\}.\end{aligned}$$

In other words, we can rewrite $\mathcal{L}^r$ using $(r/2)$ 3-dimensional Lorentz cones and one $(r/2+1)$-dimensional Lorentz cone as

$$\begin{aligned}\mathcal{L}^r = \{(z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \ : \exists \rho \in \mathbb{R}^{r/2} \quad \text{s.t.} \\ (\rho, z_0) \in \mathcal{L}^{r/2} \\ (z_{2k-1}, z_{2k}, \rho_k) \in \mathcal{L}^2 \text{ for } k \in \{1, \ldots, r/2\}\}.\end{aligned}$$

By recursively applying this procedure to the $(r/2+1)$-dimensional Lorentz cone we can rewrite $\mathcal{L}^r$ using only $(r-2)$ 3-dimensional Lorentz cones. We can then replace each of these 3-dimensional Lorentz cones with the polyhedral relaxation of $\mathcal{L}^2$ given by

$$\begin{aligned}\mathcal{W}_s := \{(z_0, z_1, z_2) \in \mathbb{R}_+ \times \mathbb{R}^2 \ : \ \exists (\alpha, \beta) \in \mathbb{R}^{2s} \quad \text{s.t.} \\ z_0 = \alpha_s \cos\left(\frac{\pi}{2^s}\right) + \beta_s \sin\left(\frac{\pi}{2^s}\right) \\ \alpha_1 = z_1 \cos(\pi) + z_2 \sin(\pi) \\ \beta_1 \geq |z_2 \cos(\pi) - z_1 \sin(\pi)| \\ \alpha_{i+1} = \alpha_i \cos\left(\frac{\pi}{2^i}\right) + \beta_i \sin\left(\frac{\pi}{2^i}\right) \\ \beta_{i+1} \geq \left|\beta_i \cos\left(\frac{\pi}{2^i}\right) - \alpha_i \sin\left(\frac{\pi}{2^i}\right)\right| \\ \text{for } i \in \{1, \ldots, s-1\}\},\end{aligned}$$

for some $s \in \mathbb{Z}$.

For a general $r$, not necessarily a power of two, these ideas and some careful selection of the parameter $s$ in $\mathcal{W}_s$ yield the polyhedral relaxation of $\mathcal{L}^r$ given by

$$\begin{aligned}\mathcal{L}^r_\varepsilon := \{(z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \ : \exists (\zeta^k)_{k=0}^K \in \mathbb{R}^{T(r)} \quad \text{s.t.} \\ z_0 = \zeta_1^K \\ \zeta_i^0 = z_i & \quad \text{for } i \in \{1, \ldots, r\}, \\ (\zeta_{2i-1}^k, \zeta_{2i}^k, \zeta_i^{k+1}) \in \mathcal{W}_{s_k(\varepsilon)} & \quad \text{for } i \in \{1, \ldots, \lfloor t_k/2 \rfloor\}, \\ & \quad k \in \{0, \ldots, K-1\}, \\ \zeta_{t_k}^k = \zeta_{\lceil t_k/2 \rceil}^{k+1} & \quad \text{for } k \in \{0, \ldots, K-1\} \quad \text{s.t.} \\ & \quad t_k \text{ is odd}\}\end{aligned}$$

where $K = \lceil \log_2(r) \rceil$, $\{t_k\}_{k=0}^K$ is defined by the recursion $t_0 = r$, $t_{k+1} = \lceil t_k/2 \rceil$ for $k \in \{0, \ldots, K-1\}$, $T(r) = \sum_{k=0}^K t_k$ and

$$s_k(\varepsilon) = \left\lceil \frac{k+1}{2} \right\rceil - \left\lceil \log_4 \left( \frac{16}{9} \pi^{-2} \log(1+\varepsilon) \right) \right\rceil. \tag{33}$$

From [15] and [54] we have that $\mathcal{L}_\varepsilon^r$ complies with (32) for any $\varepsilon > 0$ and has $O(n \log(1/\varepsilon))$ variables and constraints for any $0 < \varepsilon < 1/2$.

We can then use $\mathcal{L}_\varepsilon^r$ to define the relaxation of $\mathcal{CC}$ given by

$$\begin{aligned} \mathcal{P}(\mathcal{CC}, \varepsilon) = \{(x,y) \in \mathbb{R}^{n+p} : \exists (z_0, z) \in \mathbb{R}_+ \times \mathbb{R}^r \quad \text{s.t.} \quad & Ax + By + \delta = z, \\ & ax + by + \delta_0 = z_0, \\ & (z_0, z) \in \mathcal{L}_\varepsilon^r\}, \end{aligned}$$

which complies with

$$\mathcal{CC} \subsetneq \mathcal{P}(\mathcal{CC}, \varepsilon) \subsetneq \{(x,y) \in \mathbb{R}^{n+p} : ||Ax + By + \delta||_2 \leq (1+\varepsilon)(ax + by + \delta_0)\}.$$

Using this relaxation we can construct the lifted polyhedral relaxation of CPP given by

$$z_{\mathsf{LP}(\varepsilon)} := \max_{x,y,v} \quad cx + dy \tag{34}$$

$$s.t.$$

$$Dx + Ey \leq f \tag{35}$$

$$(x,y,v) \in \overline{\mathcal{P}}(\mathcal{CC}_i, \varepsilon) \quad i \in \mathcal{I} \tag{36}$$

$$(x,y,v) \in \mathbb{R}^{n+p+q} \tag{37}$$

where $v \in \mathbb{R}^q$ are the auxiliary variables used to construct all $\mathcal{P}(\mathcal{CC}_i, \varepsilon)$'s and $\overline{\mathcal{P}}(\mathcal{CC}_i, \varepsilon)$ is the polyhedron in $\mathbb{R}^{n+p+q}$ whose projection to $\mathbb{R}^{n+p}$ is $\mathcal{P}(\mathcal{CC}_i, \varepsilon)$. We denote the problem given by (34)–(37) as $\mathsf{LP}(\varepsilon)$ and the problem given by (34)–(37) and (18) as $\mathsf{MILP}(\varepsilon)$.

### *2.4 Computational Results*

In this Section we present the results of computational tests showing the effectiveness of the lifted LP branch-and-bound algorithm based on $\mathsf{LP}(\varepsilon)$. We begin by describing how the algorithm was implemented, then describe the problem instances we used in the tests and finally we present the computational results.

### 2.4.1 Implementation

We implemented the lifted LP branch-and-bound algorithm of Figure 4 for $\mathsf{LLPP} = \mathsf{LP}(\varepsilon)$ and $\mathsf{NLPP} = \mathsf{CPP}$ by modifying CPLEX 10.0's MILP solver. We used the branch callback feature to implement branching on integer feasible solutions when necessary and we used the incumbent and heuristic callback features to implement the solve of $\mathsf{NLPP}(\hat{x}^k)$. All coding was done in C++ using Ilog Concert Technology. We used CPLEX's barrier solver to solve $\mathsf{CPP}(l^k, u^k)$ and $\mathsf{CPP}(\hat{x})$. In all cases we used CPLEX's default settings. We denote this implementation as $\mathsf{LP}(\varepsilon)$-BB .

There are some technical differences between this implementation and the lifted LP branch-and-bound algorithm of Figure 4. First, in the CPLEX based implementation, $\mathsf{NLPP}(\hat{x}^k)$ is solved for all integer feasible solutions found. This is a difference because the algorithm of Figure 4 only finds integer solutions when $\mathsf{LLPP}(l^k, u^k)$ is integer feasible, but CPLEX also finds integer feasible solutions by using primal heuristics. Finally, the implementation benefits from other advanced CPLEX features such as preprocessing, cutting planes and sophisticated branching and node selection schemes. In particular, the addition of cutting planes conceptually modifies the algorithm as adding these cuts updates the polyhedral relaxation defining $\mathsf{LLPP}$. This updating does not use any information from the nonlinear constraints though, as CPLEX's cutting planes are only derived using the linear constraints of $\mathsf{LLPP}$ and the integrality of the $x$ variables.

### 2.4.2 Test Instances

Our test set consists of three different portfolio optimization problems with cardinality constraints from the literature [31, 86, 85]. For most portfolio optimization problems only the continuous variables are present in the nonlinear constraints and hence the convex hull of integer feasible solutions to these problems is almost never a polyhedron. Furthermore, polyhedral relaxation based algorithms for the purely continuous versions of these problems are known to converge slowly. For these reasons we believe that portfolio optimization problems are a good set of problems to test the effectiveness of the lifted LP branch-and-bound algorithm based on $\mathsf{LP}(\varepsilon)$.

For all three problems we let $a_i$ be the random return on stock $i$ and let the expected value and covariance matrix of the joint distribution of $a = (a_1, \ldots, a_n)$ be $\bar{a} \in \mathbb{R}^n_+$ and $Q$ respectively. Also, let $y_i$ be the fraction of the portfolio invested in stock $i$ and $Q^{1/2}$ be the positive semidefinite square root of $Q$.

The first problem is obtained by adding a cardinality constraint to the classical mean-variance portfolio optimization model to obtain the MICP problem already explained in (20)–(26). We refer to the set of instances of this problem as the *classical* instances.

The second problem is constructed by replacing the variance risk constraint (21) of the classical mean-variance model with two shortfall risk constraints of the form $\text{Prob}(\bar{a}y \geq W^{low}) \geq \eta$. Following [86] and [85] we formulate this model as a conic quadratic programming problem obtained by replacing constraint (21) in the classical mean-variance problem with

$$\Phi^{-1}(\eta_i)||Q^{1/2}y||_2 \leq \bar{a}y - W_i^{low} \qquad i \in \{1, 2\}$$

where $\Phi(\cdot)$ is the cumulative distribution function of a zero mean, unit variance Gaussian random variable. We refer to the set of instances of this problem as the *shortfall* instances.

The final problem is a robust portfolio optimization problem studied in [31]. This model assumes that there is some uncertainty in the expected returns $\bar{a}$ and that the true expected return vector is normally distributed with mean $\bar{a}$ and covariance matrix $R$. The model is similar to one introduced in [13] and can be formulated as the conic quadratic programming problem obtained by replacing the objective function (20) of the classical mean-variance with $\max_{x,y,r} r$ and adding the constraint $\bar{a}y - \alpha||R^{1/2}y||_2 \geq r$ where $R^{1/2}$ is the positive semidefinite square root of $R$. The effect of this change is the maximization of $\bar{a}y - \alpha||R^{1/2}y||_2$ which is a robust version of the maximization of the expected return $\bar{a}y$. We refer to the set of instances of this problem as the *robust* instances.

We generated the data for the classical instances in a manner similar to the test instances of [85]. We first estimated $\bar{a}$ and $Q$ from 251 daily closing prices of S&P 500 stocks starting with the 22nd of August 2005 and we scaled the distributions for a portfolio holding period of 20 days. Then, for each $n$ we generated an instance by randomly selecting $n$ stocks out of the 462 stocks for which we had closing price data. We also arbitrarily selected $\sigma = 0.2$

and $K = 10$.

For the shortfall instances we used the same data generated for the classical mean-variance instances, but we additionally included a risk-less asset with unit return to make these instances differ even more from the classical mean-variance instances. Also, in a manner similar to the test sets of [85] we arbitrarily selected $\eta_1 = 80\%$, $W_1^{low} = 0.9$, $\eta_2 = 97\%$, $W_2^{low} = 0.7$.

Finally, we generated the data for the robust instances in a manner similar to the test instances of [31]. We used the same daily closing prices used for the classical mean-variance and shortfall risk constraints instances, but we randomly selected different groups of $n$ stocks and we generated the data in a slightly different way. For stock $i$ we begin by calculating $\mu_i$ as the mean daily return from the first 120 days available. We then let $\bar{a}_i = 0.1\mu_i + 0.9r$ where $r$ is the daily return for the 121st day. Finally $Q$ is estimated from the same first 120 days and following [31] we let $R = (0.9/120)Q$. We also arbitrarily selected $\alpha = 3$ and we again selected $\sigma = 0.2$ and $K = 10$.

For the three sets of instances we generated 100 instances for each $n$ in $\{20, 30, 40, 50\}$ and 10 instanced for each $n$ in $\{100, 200\}$. All data sets are available at `http://www2.isye.gatech.edu/~jvielma/portfolio`.

### 2.4.3 Results

All computational tests where done on a dual 2.4GHz Xeon workstation with 2GB of RAM running Linux Kernel 2.4. The first set of experiments show calibration results for different values of $\varepsilon$. We then study how $\mathsf{LP}(\varepsilon)$-BB compares to other algorithms. Finally, we study some factors that might affect the effectiveness of $\mathsf{LP}(\varepsilon)$.

#### *2.4.3.1 Selection of $\varepsilon$*

Note that as $\varepsilon$ gets smaller the size of $\mathsf{LP}(\varepsilon)$ grows as $O(n \log(1/\varepsilon))$, on the other hand the relaxation gets tighter. To select the value of $\varepsilon$ for subsequent runs we first studied the sizes of $\mathsf{LP}(\varepsilon)$ for $n$ in $\{20, 30\}$ and for values of $\varepsilon$ in $\{1, 0.1, 0.01, 0.001, 0.0001\}$. Table 1 presents the number of columns, rows and non-zero coefficients for the different values of $n$ and $\varepsilon$.

**Table 1:** Problem Sizes for Different Values of $\varepsilon$

| | | classical | | shortfall | | robust | |
|---|---|---|---|---|---|---|---|
| $n$ | $\varepsilon$ | cols+rows | nz | cols+rows | nz | cols+rows | nz |
| 20 | 1 | 484 | 1172 | 908 | 2310 | 956 | 2368 |
| | 0.1 | 579 | 1343 | 1098 | 2652 | 1156 | 2728 |
| | 0.01 | 769 | 1685 | 1478 | 3336 | 1556 | 3448 |
| | 0.001 | 959 | 2027 | 1858 | 4020 | 1956 | 4168 |
| | 0.0001 | 1054 | 2198 | 2048 | 4362 | 2156 | 4528 |
| | CPP | 105 | 501 | 150 | 968 | 154 | 948 |
| 30 | 1 | 734 | 2076 | 1378 | 4098 | 1426 | 4146 |
| | 0.1 | 879 | 2337 | 1668 | 4620 | 1726 | 4686 |
| | 0.01 | 1169 | 2859 | 2248 | 5664 | 2326 | 5766 |
| | 0.001 | 1459 | 3381 | 2828 | 6708 | 2926 | 6846 |
| | 0.0001 | 1604 | 3642 | 3118 | 7230 | 3226 | 7386 |
| | CPP | 155 | 1051 | 220 | 2048 | 224 | 2018 |

The table also includes the same information for CPP.

We see that the sizes of LP($\varepsilon$) are considerable larger that the sizes of CPP. On the other hand we confirm that sizes only grow logarithmically with $\varepsilon$.

We additionally devised the following simple computational experiment for selecting the appropriate value of $\varepsilon$. For $n$ equal to 20 and 30 we selected the first 10 instances of each instance class and tried to solve them with values of $\varepsilon$ in $\{1, 0.1, 0.01, 0.001, 0.0001\}$. A time limit of 100 seconds was set. Note that, although Proposition 2.1 shows that LP($\varepsilon$)-BB solves the problem exactly (up to the precision of the continuous relaxation solvers) for any $\varepsilon$, for efficiency reasons we would probably never select $\varepsilon = 1$ as the resulting relaxation is too weak. However, we decided to test it anyway to illustrate that the procedure works even for this extreme choice of $\varepsilon$.

Table 2 shows the minimum, average, maximum and standard deviation of the number of nodes needed by LP($\varepsilon$)-BB to solve the instances. Tables 3 and 4 show the same statistics for solve times in seconds and the number of branch-and-bound nodes in which nonlinear relaxation CPP$(l^k, u^k)$ is solved.

Figure 5 shows the performance profile (see [48]) for all the instances using solve time as

**Table 2:** Number of Nodes for Different Values of $\varepsilon$

| stat | $\varepsilon = 1$ | $\varepsilon = 0.1$ | $\varepsilon = 0.01$ | $\varepsilon = 0.001$ | $\varepsilon = 0.0001$ |
|---|---|---|---|---|---|
| min | 0 | 0 | 0 | 0 | 0 |
| **avg** | **7760** | **1497** | **166** | **193** | **239** |
| max | 36443 | 14281 | 2390 | 2228 | 3995 |
| std | 3087 | 1342 | 196 | 303 | 289 |

**Table 3:** Solve Time for Different Values of $\varepsilon$ [s]

| stat | $\varepsilon = 1$ | $\varepsilon = 0.1$ | $\varepsilon = 0.01$ | $\varepsilon = 0.001$ | $\varepsilon = 0.0001$ |
|---|---|---|---|---|---|
| min | 0.14 | 0.07 | 0.09 | 0.12 | 0.18 |
| **avg** | **37.18** | **3.71** | **1.10** | **3.19** | **5.79** |
| max | 100.31 | 21.28 | 8.64 | 35.38 | 71.16 |
| std | 21.61 | 3.33 | 1.16 | 5.53 | 10.53 |

**Table 4:** Number of Nodes that Solve $\mathsf{CPP}(l^k, u^k)$

| stat | $\varepsilon = 1$ | $\varepsilon = 0.1$ | $\varepsilon = 0.01$ | $\varepsilon = 0.001$ | $\varepsilon = 0.0001$ |
|---|---|---|---|---|---|
| min | 1 | 1 | 1 | 0 | 0 |
| **avg** | **1700** | **74** | **4** | **3** | **3** |
| max | 6178 | 367 | 18 | 22 | 23 |
| std | 436 | 25 | 2 | 3 | 1 |

a performance metric. For a given value $m$ on the horizontal axis and a given method, the value $f$ plotted in the performance profile indicates the fraction $f$ of the instances that were solved by that method within $m$ times the length of time required by the fastest method for each instance. In particular, the intercepts of the plot (if any) with the vertical axis to the left and right indicate the fraction of the instances for which the method was the fastest and the fraction of the instances which the method could solve in the alloted time, respectively. For example, the method for $\varepsilon = 1$ was the fastest in about 5% of the instances, could solve almost 80% of the instances and could solve under 30% of the instances within 10 times the length of time required by the fastest solver. Note that these profiles are step functions. In Figure 5, and all subsequent performance figures, we mark a small subset of the data points to distinguish the individual profiles. We refer the reader to [48] for more details about the construction and interpretation of performance profiles.

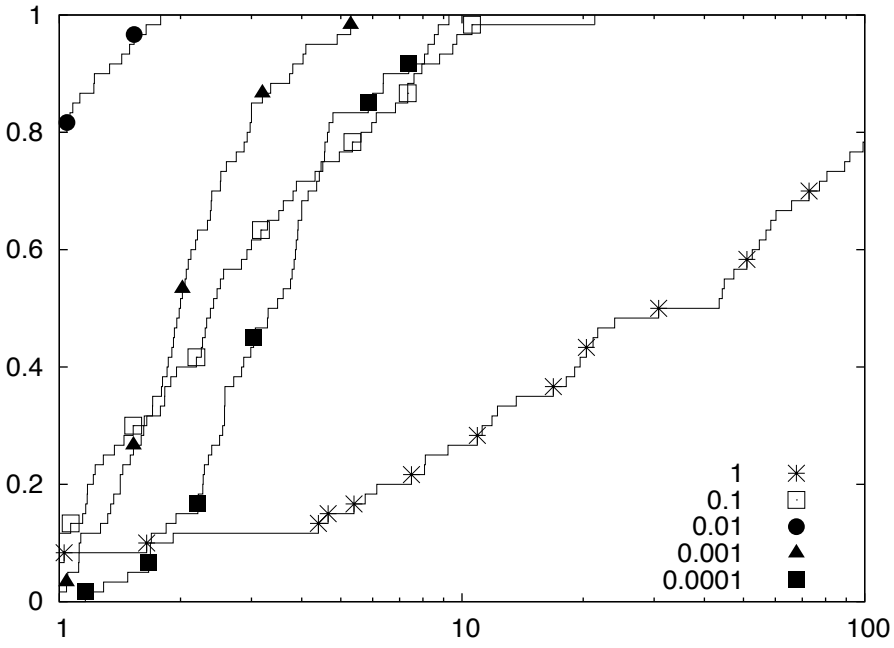



**Figure 5:** Performance Profile for Different Values of $\varepsilon$

We see that $\varepsilon = 0.01$ is the best choice on average. It also has the best performance profile and it yields the fastest method for 80% of the instances. Furthermore, for $\varepsilon = 0.01$ we have very few nodes solving $\mathsf{CPP}(l^k, u^k)$.

It is also interesting to note that the procedure still works fairly well for values of $\varepsilon$ as big as 0.1 and even for the extreme case of $\varepsilon = 1$ the procedure was still able to solve almost 80% of the instances in the alloted time of 100 seconds. For this last case though, the high number of nodes that solve $\mathsf{CPP}(l^k, u^k)$ makes the algorithm behave almost like an NLP based branch-and-bound algorithm.

Finally, we note that the result that $\varepsilon = 0.01$ requires the smallest number of branch-and-bound nodes on average is somewhat unexpected. This contradicts the belief that the algorithm of Figure 4 should require fewer branch-and-bound nodes if tighter relaxations are used. An explanation of this apparent contradiction comes from the difference between the algorithm of Figure 4 and the implementation of $\mathsf{LP}(\varepsilon)$-BB as discussed in Section 2.4.1, since the use of CPLEX's advanced features in $\mathsf{LP}(\varepsilon)$-BB makes it hard to predict the behavior of the algorithm.

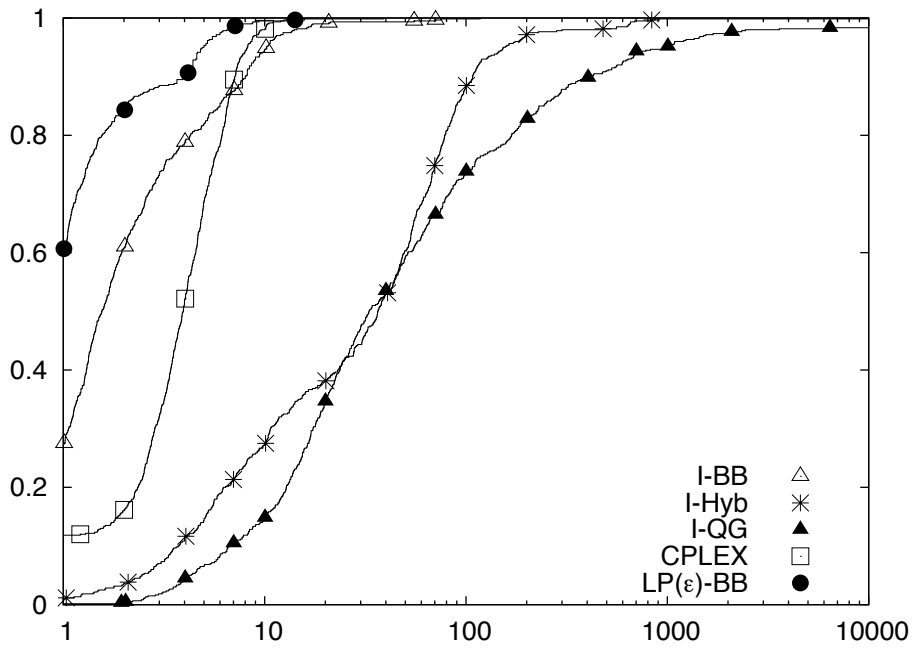



**Figure 6:** Performance Profile for Small Instances

#### *2.4.3.2 Performance of* LP($\varepsilon$)*-BB against other methods*

In this Section we compare LP($\varepsilon$)-BB with $\varepsilon = 0.01$ against other solvers. The solvers we choose for the comparison are the NLP based branch-and-bound algorithms CPLEX 10 MICP solver and Bonmin's I-BB and the polyhedral relaxation based algorithms I-QG and I-Hyb from Bonmin. We did not include Bonmin's I-OA algorithm as it performed very badly in preliminary tests.

For CPLEX we used its default settings and for the Bonmin solvers we set parameters `allowable_gap` and `allowable_fraction_gap` to $10^{-6}$ and $10^{-4}$ respectively to have the same target gaps as CPLEX. All tests were run with a time limit of 10000 seconds.

We first tested all solvers for all instances for $n$ in $\{20, 30\}$. We denote this set of instances as the *small instances*. Table 5 shows the minimum, average, maximum and standard deviations of the solve times. Figure 6 shows the performance profile for all the instances using solve time as a performance metric.

From Table 5 we see that LP($\varepsilon$)-BB is the fastest algorithm on average for all but one

**Table 5:** Solve Times for Small Instances [s]

| instance(n) | stat | LP($\varepsilon$)-BB | I-QG | I-Hyb | I-BB | CPLEX |
|---|---|---|---|---|---|---|
| classical(20) | min | 0.08 | 0.38 | 0.22 | 0.28 | 0.02 |
| | **avg** | **0.29** | **26.41** | **24.84** | **1.28** | **1.31** |
| | max | 1.06 | 222.19 | 164.71 | 13.33 | 7.95 |
| | std | 0.18 | 42.92 | 26.37 | 2.31 | 1.17 |
| classical(30) | min | 0.25 | 1.62 | 0.33 | 0.38 | 0.73 |
| | **avg** | **1.65** | **1434.86** | **217.25** | **13.19** | **9.68** |
| | max | 27.0 | 10005.2 | 10003.3 | 573.97 | 324.63 |
| | std | 3.21 | 2768.34 | 1016.68 | 59.17 | 33.68 |
| shortfall(20) | min | 0.19 | 0.18 | 0.26 | 0.34 | 0.03 |
| | **avg** | **0.48** | **17.42** | **16.78** | **0.63** | **1.68** |
| | max | 1.65 | 174.62 | 58.45 | 3.52 | 5.19 |
| | std | 0.21 | 30.77 | 17.96 | 0.52 | 0.89 |
| shortfall(30) | min | 0.4 | 1.25 | 0.57 | 0.47 | 1.26 |
| | **avg** | **2.20** | **847.63** | **136.39** | **5.00** | **9.26** |
| | max | 29.34 | 10003.1 | 5907.32 | 73.81 | 80.36 |
| | std | 3.21 | 1992.86 | 588.61 | 10.00 | 12.20 |
| robust(20) | min | 0.19 | 0.39 | 0.12 | 0.37 | 0.03 |
| | **avg** | **0.39** | **4.99** | **15.51** | **2.57** | **1.03** |
| | max | 1.05 | 33.85 | 599.46 | 57.22 | 3.5 |
| | std | 0.20 | 5.60 | 60.37 | 10.25 | 0.90 |
| robust(30) | min | 0.43 | 0.59 | 0.29 | 0.48 | 0.07 |
| | **avg** | **1.20** | **75.07** | **23.43** | **1.02** | **3.54** |
| | max | 4.72 | 2071.47 | 134.08 | 4.92 | 10.76 |
| | std | 0.81 | 284.39 | 25.49 | 0.87 | 2.45 |

set of instances and that this average can be up to five times better than the average for its closest competitor. Furthermore, as the standard deviation and maximum numbers show, LP($\varepsilon$)-BB is far more consistent in providing good solve times than the other methods. From Figure 6 we can also see that LP($\varepsilon$)-BB has the best performance profile, that it is the fastest solver in 60% of the instances and that it is almost never an order of magnitude slower than the best solver.

Our second set of tests include all instances for $n$ in $\{40, 50\}$. We denote this set of instances as the *medium instances*. We did not include I-QG in these tests as it performed very poorly on the instances for $n = 30$ and reached the time limit in several instances. Although I-Hyb performed close to I-QC we included it in these tests as it had only reached the time limit in one instance and we wanted to have at least one of the original LP/NLP

based branch-and-bound solvers in our tests. Table 6 shows the minimum, average, maximum and standard deviation of the solve times. Figure 7 shows the performance profile for all the instances using solve time as a performance metric.

**Table 6:** Solve Times for Medium Instances [s]

| instance(n) | stat | LP($\varepsilon$)-BB | I-Hyb | I-BB | CPLEX |
|---|---|---|---|---|---|
| classical(40) | min | 0.56 | 35.04 | 0.61 | 1.55 |
| | **avg** | **14.84** | **1412.23** | **144.17** | **63.41** |
| | max | 554.52 | 10006.0 | 8518.95 | 2033.65 |
| | std | 56.64 | 2631.92 | 848.84 | 208.86 |
| classical(50) | min | 0.76 | 35.17 | 0.75 | 4.12 |
| | **avg** | **102.88** | **4139.92** | **894.00** | **636.83** |
| | max | 1950.81 | 12577.8 | 10030.1 | 10000.0 |
| | std | 270.96 | 4343.71 | 2048.96 | 1626.37 |
| shortfall(40) | min | 1.17 | 34.72 | 0.7 | 4.93 |
| | **avg** | **16.60** | **956.98** | **92.85** | **111.97** |
| | max | 389.57 | 10004.6 | 4888.26 | 4259.5 |
| | std | 43.85 | 2133.56 | 489.98 | 430.95 |
| shortfall(50) | min | 1.58 | 33.22 | 0.96 | 5.69 |
| | **avg** | **163.10** | **3143.84** | **452.05** | **567.74** |
| | max | 7674.86 | 10006.0 | 10034.1 | 10000.0 |
| | std | 771.98 | 3803.14 | 1285.52 | 1319.39 |
| robust(40) | min | 0.51 | 0.43 | 0.69 | 0.14 |
| | **avg** | **3.82** | **59.10** | **4.31** | **11.17** |
| | max | 42.57 | 1141.91 | 129.82 | 160.71 |
| | std | 6.04 | 130.37 | 14.64 | 18.58 |
| robust(50) | min | 0.92 | 0.65 | 0.93 | 0.25 |
| | **avg** | **20.44** | **435.43** | **23.67** | **41.71** |
| | max | 443.29 | 10002.1 | 746.37 | 876.31 |
| | std | 63.47 | 1702.15 | 95.68 | 120.24 |

We see from Table 6 that LP($\varepsilon$)-BB is now the fastest algorithm on average for all instances and this average can be up to six times better than the average for its closest competitor. Again, as the standard deviation and maximum numbers show, LP($\varepsilon$)-BB is far more consistent in providing good solve times than the other methods. From Figure 7 we again see that LP($\varepsilon$)-BB has the best performance profile, that it is the fastest solver in over 60% of the instances and that it is almost never an order of magnitude slower than the best solver. Moreover, it is the only solver with this last property.

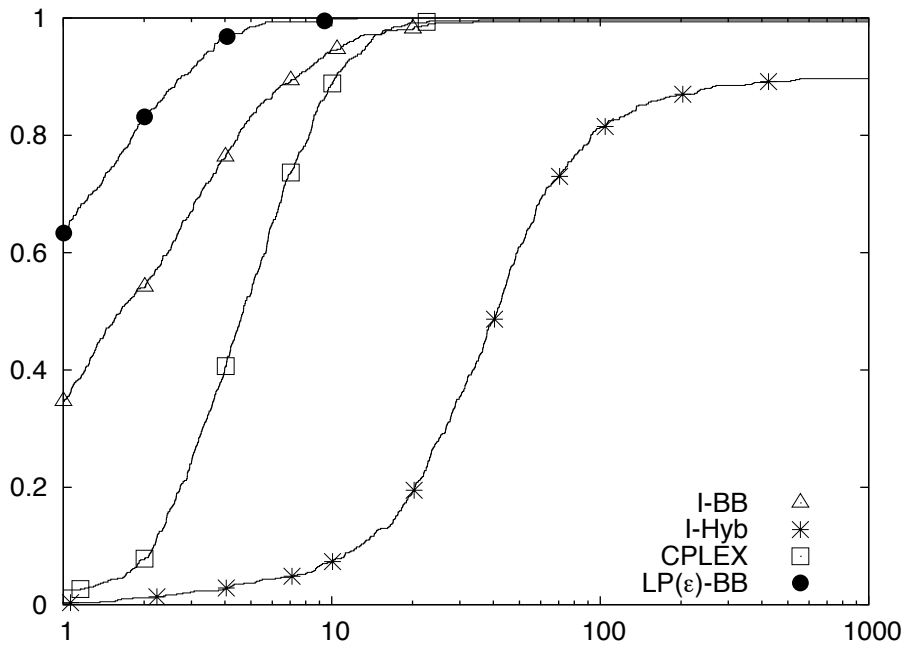



**Figure 7:** Performance Profile for Medium Instances

Our last set of tests include instances for $n$ in $\{100, 200\}$. We denote this set of instances as the *large instances*. We do not include the results for I-Hyb as it was not able to solve any of the instances in this group. Neither did we include results for the classical or shortfall instances for $n = 200$ as none of the methods could solve a single instance in the alloted time. Table 7 shows the minimum, average and maximum of the solve times. We did not include standard deviations as time limits were reached for too many instances. We instead include the number of instances (out of a total of 10 per instance class) that each method could solve in the alloted time. Figure 8 shows the performance profile for all the instances using solve time as a performance metric.

From Table 7 we see that LP($\varepsilon$)-BB is the fastest algorithm on average for all but one set of instances in this group. Furthermore, for all instance classes it is the method that solves the largest number of instances. From Figure 8 we can also see that LP($\varepsilon$)-BB has the best performance profile, that it is the fastest solver in about 40% of the instances and that it is the method that is able to solve the greatest number of instances in the alloted

**Table 7:** Solve Times for Large Instances [s]

| instance(n) | stat | LP($\varepsilon$)-BB | I-BB | CPLEX |
|---|---|---|---|---|
| classical(100) | min | 1653 | 3497 | 4503 |
| | **avg** | **7443** | **8605** | **8767** |
| | max | 10012 | 10035 | 10000 |
| | solved | 4 | 3 | 3 |
| shortfall(100) | min | 2014 | 4105 | 8733 |
| | **avg** | **6660** | **8497** | **9818** |
| | max | 10003 | 10163 | 10000 |
| | solved | 6 | 4 | 2 |
| robust(100) | min | 30 | 4 | 85 |
| | **avg** | **956** | **612** | **1395** |
| | max | 4943 | 2684 | 5294 |
| | solved | 10 | 10 | 10 |
| robust(200) | min | 1458 | 1775 | 9789 |
| | **avg** | **6207** | **7346** | **9979** |
| | max | 10138 | 10016 | 10000 |
| | solved | 6 | 5 | 1 |

time.

#### *2.4.3.3 Factors that affect the effectiveness of* LP($\varepsilon$)*-BB*

In this subsection we study some factors that might affect the effectiveness of LP($\varepsilon$) including solve times, accuracy and size of the polyhedral relaxation LP($\varepsilon$), number of branch-and-bound nodes processed and number of calls to the nonlinear relaxations.

We begin by confirming the results from [54] that solving the polyhedral relaxation LP($\varepsilon$) is slower than solving CPP directly still hold for our tests instances. To confirm this we solved CPP with CPLEX 10's barrier solver and LP(0.01) with CPLEX 10's primal simplex, dual simplex and LP barrier solvers. We did this for all instances for $n$ equal to 100 and 200. Table 8 shows the minimum, average, maximum and standard deviation of the solve times.

We see that solving LP(0.01) is slower than solving CPP. In fact, solving LP(0.01) with dual simplex is more than twice as slow as solving CPP with barrier. Hence, it is not the solve times of a single relaxation that gives the LP($\varepsilon$)-BB algorithm the advantage over the NLP based branch-and-bound algorithms. Note that the "warm start" capabilities of an

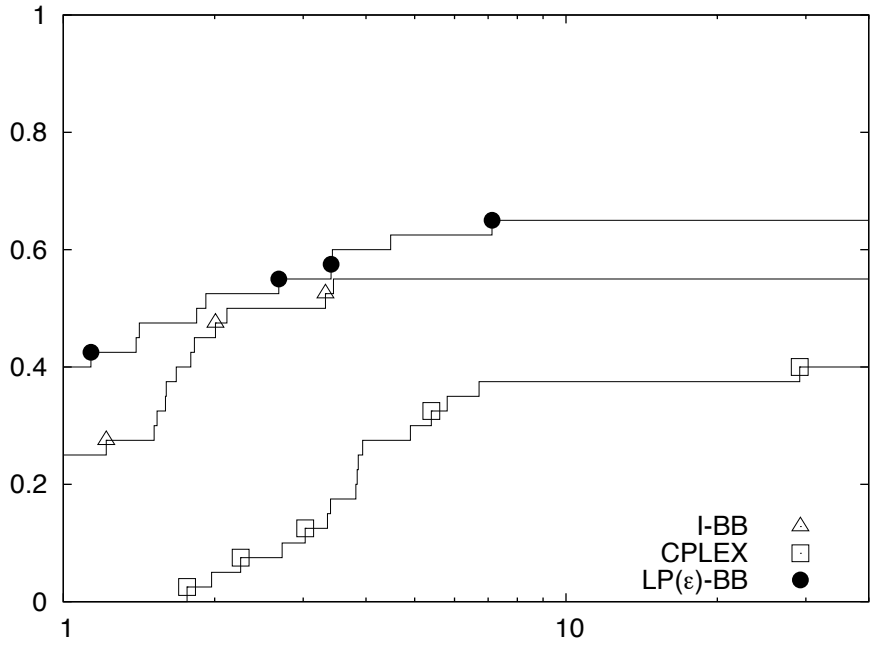



**Figure 8:** Performance Profile for Large Instances

**Table 8:** Solve Time for Root Relaxation [s]

| | CPP | LP(0.01) | | |
|---|---|---|---|---|
| stat | Barrier | P. Simplex | D. Simplex | Barrier |
| min | 0.09 | 0.72 | 0.58 | 0.19 |
| **avg** | **4.88** | **34.94** | **11.66** | **6.94** |
| max | 20.39 | 174.45 | 49.69 | 29.7 |
| std | 1.50 | 10.45 | 2.86 | 2.05 |

LP solver might still make solving a series of similar LP(0.01) problems faster than solving a series of similar CPP problems.

It is noted in [54] that although the relaxation used to construct LP($\varepsilon$) gives accuracy guarantees on $\mathcal{L}^r_\varepsilon$, it is in general not possible to give *a priori* guarantees on the accuracy of LP($\varepsilon$) or its optimal objective value $z_{\mathsf{LP}(\varepsilon)}$. For this reason we studied empirically the accuracy of $z_{\mathsf{LP}(\varepsilon)}$ on our set of test instances. To do this we calculated the value of the accuracy measure $100 \times (z_{\mathsf{LP}(\varepsilon)} - z_{\mathsf{CPP}})/z_{\mathsf{CPP}}$ for all of our test instances and for values of $\varepsilon$ in $\{1, 0.1, 0.01, 0.001, 0.0001\}$. Table 9 shows the minimum, average, maximum and standard

**Table 9:** Accuracy of Relaxation $z_{\mathsf{LP}(\varepsilon)}$ [%]

| stat | $\varepsilon = 1$ | $\varepsilon = 0.1$ | $\varepsilon = 0.01$ | $\varepsilon = 0.001$ | $\varepsilon = 0.0001$ |
|---|---|---|---|---|---|
| min | 0.25 | 0.07 | 0.00 | 0.00 | 0.00 |
| **avg** | **4.30** | **1.14** | **0.07** | **0.00** | **0.00** |
| max | 13.94 | 5.16 | 0.29 | 0.02 | 0.01 |
| std | 0.80 | 0.21 | 0.01 | 0.00 | 0.00 |

deviation of this accuracy measure.

We see that even for the extreme case $\varepsilon = 1$ the accuracy of $z_{\mathsf{LP}(\varepsilon)}$ is fairly good and for our chosen value of $\varepsilon = 0.01$ the accuracy is extremely good. This accuracy is likely one of the reasons for the effectiveness of the $\mathsf{LP}(\varepsilon)$-BB algorithm. We note that Proposition 2.1 implies that $\mathsf{LP}(\varepsilon)$-BB would still be exact (up to the precision of the continuous relaxation solvers) even if the accuracy of $z_{\mathsf{LP}(\varepsilon)}$ was not good, but in this case its performance could be significantly reduced by the need to solve more nonlinear relaxations.

We next study the sizes of $\mathsf{LP}(0.01)$ as $n$ varies. Table 10 presents the number of columns, rows and non-zero coefficients for different values of $n$. Again, we also include the sizes of CPP. We see that $\mathsf{LP}(0.01)$ can be up to about an order of magnitude larger than CPP. However, we also confirm that the size of $\mathsf{LP}(0.01)$ only grows linearly with $n$.

**Table 10:** Problem Sizes for Different Values of $n$

| | | classical | | shortfall | | robust | |
|---|---|---|---|---|---|---|---|
| $n$ | Relaxation | cols+rows | nz | cols+rows | nz | cols+rows | nz |
| 20 | CPP | 105 | 501 | 150 | 968 | 154 | 948 |
| | $\mathsf{LP}(0.01)$ | 769 | 1685 | 1478 | 3336 | 1556 | 3448 |
| 30 | CPP | 155 | 1051 | 220 | 2048 | 224 | 2018 |
| | $\mathsf{LP}(0.01)$ | 1169 | 2859 | 2248 | 5664 | 2326 | 5766 |
| 40 | CPP | 205 | 1801 | 290 | 3528 | 294 | 3488 |
| | $\mathsf{LP}(0.01)$ | 1574 | 4242 | 3028 | 8410 | 3116 | 8520 |
| 50 | CPP | 255 | 2751 | 360 | 5408 | 364 | 5358 |
| | $\mathsf{LP}(0.01)$ | 1979 | 5825 | 3808 | 11556 | 3886 | 11638 |
| 100 | CPP | 505 | 10501 | 710 | 20808 | 714 | 20708 |
| | $\mathsf{LP}(0.01)$ | 3994 | 16722 | 7688 | 33250 | 7766 | 33282 |
| 200 | CPP | 1005 | 41001 | 1410 | 81608 | 1414 | 81408 |
| | $\mathsf{LP}(0.01)$ | 8024 | 53516 | 15448 | 106638 | 15536 | 106588 |

Finally, we study the number of branch-and-bound nodes processed and the number of calls to the nonlinear relaxations. We begin by comparing $\mathsf{LP}(\varepsilon)$-BB to the two NLP based branch-and-bound solvers. To do this, we selected all instances that solved within the time limit by I-BB, CPLEX and $\mathsf{LP}(\varepsilon)$-BB . For these instances we present in Table 11 the total number of branch-and-bound nodes processed by each method and the total number of calls to nonlinear relaxations $\mathsf{CPP}(l^k, u^k)$ and $\mathsf{CPP}(\hat{x})$ made by $\mathsf{LP}(\varepsilon)$-BB .

**Table 11:** Total Number of Nodes and Calls to Relaxations for All Instances

| | |
|---|---|
| B-and-b nodes I-BB | 3007029 |
| B-and-b nodes CPLEX | 3224115 |
| B-and-b nodes $\mathsf{LP}(\varepsilon)$-BB | 2027332 |
| $\mathsf{LP}(\varepsilon)$-BB calls to $\mathsf{CPP}(l^k, u^k)$ | 5818 |
| $\mathsf{LP}(\varepsilon)$-BB calls to $\mathsf{CPP}(\hat{x})$ | 17784 |

We see that the total number of calls to the nonlinear relaxations made by $\mathsf{LP}(\varepsilon)$-BB is only around 1% of the total number of branch-and-bound nodes processed. This is another reason for the effectiveness of $\mathsf{LP}(\varepsilon)$-BB as it has to solve very few expensive nonlinear relaxations. An interesting observation is that $\mathsf{CPP}(\hat{x})$ is solved more times than $\mathsf{CPP}(l^k, u^k)$. This is expected as CPLEX usually finds most of the integer feasible solutions with its primal heuristic than at integer feasible nodes. On the other hand, the fact that $\mathsf{LP}(\varepsilon)$-BB processed fewer branch-and-bound nodes than the two NLP based branch-and-bound methods is somewhat unexpected. Because $\mathsf{LP}(\varepsilon)$ is a relaxation of $\mathsf{CPP}$ we would expect that a pure branch-and-bound algorithm based on $\mathsf{LP}(\varepsilon)$ should process at least the same number of nodes as an algorithm based on $\mathsf{CPP}$. We believe that the reason for this unexpected behavior is that CPLEX is not a pure branch-and-bound solver. $\mathsf{LP}(\varepsilon)$-BB benefits from CPLEX being able to use some features which are currently available for MILP problems, but not for MICP problems, such as advanced preprocessing, branching rules, cutting planes and heuristics.

Finally, we compared $\mathsf{LP}(\varepsilon)$-BB to the polyhedral relaxation based solvers by selecting all instances that were solved within the time limit by I-QG, I-Hyb and $\mathsf{LP}(\varepsilon)$-BB . For

these instances we present in Table 12 the total number of branch-and-bound nodes processed by each method. Because the instances used to generate Table 12 are not the same as the ones used to generate Table 11 we also include, as a reference, the total number of branch-and-bound nodes processed by I-BB and CPLEX and the total number of calls to nonlinear relaxations $\mathsf{CPP}(l^k, u^k)$ and $\mathsf{CPP}(\hat{x})$ made by $\mathsf{LP}(\varepsilon)$-BB .

**Table 12:** Total Number of Nodes and Calls to Relaxations for Small Instances

| | |
|---|---|
| B-and-b nodes I-QG | 3580051 |
| B-and-b nodes I-Hyb | 328316 |
| B-and-b nodes I-BB | 68915 |
| B-and-b nodes CPLEX | 85957 |
| B-and-b nodes $\mathsf{LP}(\varepsilon)$-BB | 57933 |
| $\mathsf{LP}(\varepsilon)$-BB calls to $\mathsf{CPP}(l^k, u^k)$ | 2305 |
| $\mathsf{LP}(\varepsilon)$-BB calls to $\mathsf{CPP}(\hat{x})$ | 7810 |

We see that I-Hyb needed almost four times the number of nodes needed by CPLEX and I-QG needed over 40 times as many nodes as CPLEX. In contrast, $\mathsf{LP}(\varepsilon)$-BB was the algorithm that needed the fewest number of nodes. This confirms that the relaxation $\mathsf{LP}(\varepsilon)$ is extremely good for our set of instances.

### *2.5 Conclusions and Further Work*

We have introduced a branch-and-bound algorithm for convex MINLP problems that is based on a lifted polyhedral relaxation, does not update the relaxation using gradient inequalities and sometimes branches on integer feasible variables. We have also demonstrated how this lifted LP branch-and-bound algorithm can be very effective when a good lifted polyhedral relaxation is available. More specifically, we have shown that the lifted LP branch-and-bound algorithm based on $\mathsf{LP}(\varepsilon)$ can significantly outperform other methods for solving a series of portfolio optimization problems with cardinality constraints. One reason for this good performance is that, for these problems, high accuracy of $\mathcal{L}_\varepsilon^r$ translates into high accuracy of $\mathsf{LP}(\varepsilon)$ which results in the construction of a tight but small polyhedral relaxation. Another factor is that by using a polyhedral relaxation of the nonlinear constraints we can benefit from "warm start" capabilities of the simplex LP algorithm and the

many advanced features of CPLEX's MILP solver. It is curious to note that a statement similar to this last one can also be made for the other polyhedral relaxation based algorithms we tested and these were the worst performers in our tests. It seems then that using $\mathsf{LP}(\varepsilon)$ provides a middle point between NLP based branch-and-bound solvers and polyhedral relaxation based solvers which only use gradient inequalities by inheriting most of the good properties of this last class without suffering from slow convergence of the relaxations.

Although the lifted LP branch-and-bound algorithm based on $\mathsf{LP}(\varepsilon)$ we have presented is already very efficient there are many improvements that can be made to it. While the version of $\mathsf{LP}(\varepsilon)$ that we used achieves the best possible asymptotic order of magnitude of variables and constraints (see [15] and [54]), it is shown in [54] that for a fixed $r$ and $\varepsilon$ it can be improved further. Using a slightly smaller version of $\mathsf{LP}(\varepsilon)$ would probably not increase significantly the performance of the algorithm for our test instances, but it could provide an advantages for problems with many conic constraints of the form $\mathcal{CC}$.

The choice of value $\varepsilon$ for $\mathsf{LP}(\varepsilon)$ is another aspect that can be studied further. The dependence of $\mathsf{LP}(\varepsilon)$ on $\varepsilon$ is through the function $\left\lceil \log_4 \left( \frac{16}{9} \pi^{-2} \log(1+\varepsilon) \right) \right\rceil$ in (33). Hence, there is only a discrete set of possible choices of $\varepsilon$ in a certain interval that yield different relaxations $\mathsf{LP}(\varepsilon)$. This allows for a refinement of the calibration experiments of Section 2.4.3.1. For example, in our calibration experiment the only different relaxations $\mathsf{LP}(\varepsilon)$ for values of $\varepsilon$ in $[0.001, 0.1]$ are the ones corresponding to values of $\varepsilon$ in $\{0.1, 0.03, 0.01, 0.004, 0.001\}$. By re-running our calibration experiments for all of these values of $\varepsilon$ we discovered that $\varepsilon = 0.01$ was still the best choice on average. This suggests the existence of, in some sense, an optimal choice of $\varepsilon$. The choice of this $\varepsilon$ could become more complicated though when the more elaborate constructions of [54] are used. An alternative to choosing $\varepsilon$ *a priori* is to choose a moderate initial value and refine the relaxation inside the branch-and-bound procedure. It is not clear how to do this efficiently though.

We are currently studying some of these issues and the possibility of extending this work to other classes of convex MINLP problems.

# CHAPTER III

# MODELING DISJUNCTIVE CONSTRAINTS WITH A LOGARITHMIC NUMBER OF BINARY VARIABLES AND CONSTRAINTS

## *3.1 Introduction*

Since the 1957 paper by Dantzig [39], the issue of modeling problems as mixed integer programs (MIPs) has been extensively studied. A study of the problems that can be modeled as MIPs began with Meyer [98, 99, 100, 101] and was continued by Jeroslow and Lowe [64, 66, 67, 68, 87].

An important question in the area of mixed integer programming (MIP) is characterizing when a disjunctive constraint of the form

$$z \in \bigcup_{i \in I} P_i \subset \mathbb{R}^n, \tag{38}$$

where $P_i = \{z \in \mathbb{R}^n \, : \, A^i z \leq b^i\}$ and $I$ is a finite index set, can be modeled as a binary integer program. Jeroslow and Lowe [64, 67, 87] showed that a necessary and sufficient condition is for $\{P_i\}_{i \in I}$ to be a finite family of polyhedra with a common recession cone. That is, the directions of unboundedness of the polyheda given by $\{z \in \mathbb{R}^n \, : \, A^i z \leq 0\}$ for $i \in I$ are all equal. Using results from disjunctive programming [6, 7, 9, 26, 63, 119] they showed that, in this case, constraint (38) can be simply modeled as

$$A^i z^i \leq x_i b^i \quad \forall i \in I, \quad z = \sum_{i \in I} z^i, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0, 1\} \quad \forall i \in I. \tag{39}$$

The possibility of reducing the number of continuous variables in these models has been studied in [8, 27, 65], but the number of binary variables and extra constraints needed to model (38) has received little attention. However, it has been observed that a careful construction can yield a much smaller model than a naive approach. Perhaps the simplest example comes from the equivalence between general integer and binary integer programming. The requirement $x \in [0, u] \cap \mathbb{Z}$ can be written in the form (38) by letting $P_i := \{i\}$

for all $i$ in $I := [0, u] \cap \mathbb{Z}$ which, after some algebraic simplifications, yields a representation of the form (39) given by

$$z = \sum_{i \in I} i\, x_i, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0, 1\} \quad \forall i \in I. \tag{40}$$

This formulation has a number of binary variables that is linear in $|I|$ and can be replaced by

$$z = \sum_{i=0}^{\lfloor \log_2 u \rfloor} 2^i\, x_i, \quad z \leq u, \quad x_i \in \{0, 1\} \quad \forall i \in \{0, \ldots, \lfloor \log_2 u \rfloor\}. \tag{41}$$

In contrast to (40), (41) has a number of binary variables that is logarithmic in $|I|$. Another example of a model with a logarithmic number of variables is the work in [79], which also considers polytopes of the form $P_i := \{i\}$ to model different choices from an abstract set $I$. This work is used in [81] to model edge coloring problems by using $I = \{\text{possible colors}\}$.

Although (41) appears in the mathematical programming literature as early as [134], and the possibility of modeling with a logarithmic number of binary variables and a linear number of constraints is studied in the theory of disjunctive programming [7] and in [61], we are not aware of any formulation with a logarithmic number of binary variables and extra constraints for the case in which each polyhedron $P_i$ contains more than one point.

The main objective of this chapter is to show that some well known classes of constraints of the form (38) can be modeled with a logarithmic number of binary variables and extra constraints. Although modeling with fewer binary variables and constraints might seem advantageous, a smaller formulation is not necessarily a better formulation. More constraints might provide a tighter LP relaxation and more variables might do the same by exploiting the favorable properties of projection [10]. For this reason, we will also show that under some conditions our new formulations are as tight as any other mixed integer formulation, and we empirically show that they can provide a significant computational advantage.

The chapter is organized as follows. In Section 3.2 we study the modeling of a class of hard combinatorial constraints. In particular we introduce the first formulations for SOS1 and SOS2 constraints that use only a logarithmic number of binary variables and extra constraints. In Section 3.3 we relate the modeling with a logarithmic number of binary variables to branching and we introduce sufficient conditions for these models to exist. We

then show that for a broad class of problems the new formulations are as tight as any other mixed integer programming formulation. In Section 3.4 we use the sufficient conditions to present a new formulation for non-separable piecewise linear functions of one and two variables that uses only a logarithmic number of binary variables and extra constraints. In Section 3.5 we study the extension of the formulations from Sections 3.2 and 3.3 to a slightly different class of constraints and study the strength of these formulations. In Section 3.6 we show that the new models for piecewise linear functions of one and two variables can perform significantly better than the standard binary models. Section 3.7 gives some conclusions.

## *3.2 Modeling a Class of Hard Combinatorial Constraints*

In this section we study a class of constraints of the form (38) in which the polyhedra $P_i$ have the simple structure of only allowing some subsets of variables to be non-zero. Specifically, we study constraints over a vector of continuous variables $\lambda$ indexed by a finite set $J$ that are of the form

$$\lambda \in \bigcup_{i \in I} Q(S_i) \subset \Delta^J, \tag{42}$$

where $I$ is a finite set such that $|I|$ is a power of two, $\Delta^J := \{\lambda \in \mathbb{R}_+^{|J|} : \sum_{j \in J} \lambda_j \leq 1\}$ is the $|J|$-dimensional simplex in $\mathbb{R}^{|J|}$, $S_i \subset J$ for each $i \in I$ and

$$Q(S_i) = \left\{\lambda \in \Delta^J : \lambda_j = 0 \quad \forall\, j \notin S_i\right\}. \tag{43}$$

Furthermore, without loss of generality we assume that $\bigcup_{i \in I} S_i = J$. Since $Q(S_i)$ is a face of $\Delta^J$ we call $\Delta^J$ the *ground set* of the constraint. Except for Theorem 3.8, our results easily extend to the case in which the simplex is replaced by a box in $\mathbb{R}_+^{|J|}$, but the restriction to $\Delta^J$ greatly simplifies the presentation. We will study this extension in Section 3.5. We finally note that the requirement of $|I|$ being a power of two is without loss of generality as we can allways add $2^{\lceil \log_2 |I| \rceil} - |I|$ polyhedra $Q(S_i)$ with $S_i = \emptyset$ to (42). We study the implications of this completion on formulation sizes in Section 3.3.

Disjunctive constraint (42) includes SOS1 and SOS2 constraints [12] over continuous variables in $\Delta^J$. SOS1 constraints on $\lambda \in \mathbb{R}_+^n$ allow at most one of the $\lambda$ variables to be

non-zero which can be modeled by letting $I = J = \{1, \ldots, n\}$ and $S_i = \{i\}$ for each $i \in I$. SOS2 constraints on $(\lambda_j)_{j=0}^n \in \mathbb{R}_+^{n+1}$ allow at most two $\lambda$ variables to be non-zero and have the extra requirement that if two variables are non-zero their indices must be adjacent. This can be modeled by letting $I = \{1, \ldots, n\}$, $J = \{0, \ldots, n\}$ and $S_i = \{i-1, i\}$ for each $i \in I$.

Mixed integer binary models for SOS1 and SOS2 constraints have been known for many years [40, 95], and some recent research has focused on branch-and-cut algorithms that do not use binary variables [42, 72, 73, 96]. However, the incentive of being able to use state of the art MIP solvers (see for example the discussion in section 5 of [131]) makes binary models for these constraints very attractive [36, 92, 105, 118].

We first review a formulation for (42) with a linear number of binary variables and a formulation with a logarithmic number of binary variables and a linear number of extra constraints. We then study how to obtain a formulation with a logarithmic number of variables and a logarithmic number of extra constraints and show that this can be achieved for SOS1 and SOS2 constraints.

The most direct way of formulating (42) as an integer programming problem is by assigning a binary variable for each set $Q(S_i)$ and using formulation (39). After some algebraic simplifications this yields the formulation of (42) given by

$$\lambda \in \Delta^J, \quad \lambda_j \leq \sum_{i \in I(j)} x_i \quad \forall j \in J, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I \tag{44}$$

where $I(j) = \{i \in I \,:\, j \in S_i\}$. This gives a formulation with $|I|$ binary variables and $|J| + 1$ extra constraints and yields standard formulations for SOS1 and SOS2 constraints. (We consider the inequalities of ground set $\Delta^J$ as the original constraints and disregard the bounds on $x$.)

The following theorem shows that by using techniques from [61] we can obtain a formulation with $\log_2 |I|$ binary variables and $|I|$ extra constraints.

**Theorem 3.1.** *Let $L(r) := \{1, \ldots, \log_2 r\}$, $B : I \to \{0,1\}^{\log_2 |I|}$ be any injective function and $\sigma(B)$ be the support of vector $B$. Then*

$$\sum_{j \notin S_i} \lambda_j \leq \sum_{l \notin \sigma(B(i))} x_l + \sum_{l \in \sigma(B(i))} (1 - x_l) \quad \forall i \in I, \quad \lambda \in \Delta^J, \quad x_l \in \{0,1\} \quad \forall l \in L(|I|) \tag{45}$$

*is a valid formulation for* (42).

*Proof.* The formulation simply fixes $\lambda_j$ to zero for all $j \notin S_i$ when $x$ takes the value $B(i)$. □

The following example illustrates formulation (45) for SOS1 and SOS2 constraints.

**Example 3.1.** *Let* $J = \{1, \ldots, 4\}$, $(\lambda_j)_{j=1}^4 \in \Delta^J$ *be SOS1 constrained and let* $B^*(1) = (1,1)^T$, $B^*(2) = (1,0)^T$, $B^*(3) = (0,1)^T$ *and* $B^*(4) = (0,0)^T$. *Formulation* (45) *for this case with* $B = B^*$ *is*

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_2 + \lambda_3 + \lambda_4 \leq 2 - x_1 - x_2, \quad \lambda_1 + \lambda_3 + \lambda_4 \leq 1 - x_1 + x_2,$$
$$\lambda_1 + \lambda_2 + \lambda_4 \leq 1 + x_1 - x_2, \quad \lambda_1 + \lambda_2 + \lambda_3 \leq x_1 + x_2.$$

*Let* $J = \{0, \ldots, 4\}$ *and* $(\lambda_j)_{j=0}^4 \in \Delta^J$ *be SOS2 constrained. Formulation* (45) *for this case with* $B = B^*$ *is*

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_2 + \lambda_3 + \lambda_4 \leq 2 - x_1 - x_2, \quad \lambda_0 + \lambda_3 + \lambda_4 \leq 1 - x_1 + x_2,$$
$$\lambda_0 + \lambda_1 + \lambda_4 \leq 1 + x_1 - x_2, \quad \lambda_0 + \lambda_1 + \lambda_2 \leq x_1 + x_2.$$

For SOS1 constraints, for which $|I(j)| = 1$ for all $j \in J$, we obtain the following alternative formulation of (42) which has $\log_2 |I|$ binary variables and $2 \log_2 |I|$ extra constraints.

**Theorem 3.2.** *Let* $B : I \to \{0,1\}^{\log_2 |I|}$ *be any injective function. Then*

$$\lambda \in \Delta^J, \sum_{j \in J^+(l,B)} \lambda_j \leq x_l, \sum_{j \in J^0(l,B)} \lambda_j \leq (1 - x_l) \quad \forall j \in J, \quad x_l \in \{0,1\} \quad \forall l \in L(|I|), \quad (46)$$

*where* $J^+(l,B) = \{j \in J : \forall i \in I(j) \quad l \in \sigma(B(i))\}$ *and* $J^0(l,B) = \{j \in J : \forall i \in I(j)$ $l \notin \sigma(B(i))\}$, *is a valid formulation for SOS1 constraints.*

*Proof.* For SOS1 constraints we have $I = J = \{1, \ldots, n\}$ and $S_j = \{j\}$ for each $i \in I$. This implies that $I(j) = \{j\}$ and hence $J^+(l,B) = \{j \in J : l \in \sigma(B(j))\}$ and $J^0(l,B) = \{j \in J : l \notin \sigma(B(j))\}$. Then, in formulation (46), we have that $\lambda_j = 0$ for all $x \neq B(j)$. □

The following example illustrates formulation (46) for SOS1 constraints.

**Example 3.2.** *Let* $J = \{1, \ldots, 4\}$, $(\lambda_j)_{j=1}^4 \in \Delta^J$ *be SOS1 constrained. Formulation* (46) *for this case with* $B = B^*$ *from Example 1 is*

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_1+\lambda_2 \leq x_1, \quad \lambda_3+\lambda_4 \leq 1-x_1, \quad \lambda_1+\lambda_3 \leq x_2, \quad \lambda_2+\lambda_4 \leq 1-x_2.$$

We don't know how to give meaning to the binary variables in formulation (45) because fixing them individually has little effect on the $\lambda$ variables. For example fixing $x_1 = 1$ and letting $x_2$ be free in either of the formulations of Example 1 has no effect on the $\lambda$ variables. In contrast fixing $x_l = 1$ individually in (46) has the very precise effect fixing to zero all $\lambda_j$'s for which $B(i)_l = 0$ for all $i$ such that $j \in S_i$. Analogously, fixing $x_l = 0$ individually in (46) fixes to zero all $\lambda_j$'s for which $B(i)_l = 1$ for all $i$ such that $j \in S_i$. Fixing the binary variables then gives a way of enforcing $\lambda \in Q(S_i)$ by systematically fixing certain $\lambda$ variables to zero.

Formulation (46) is valid for SOS1 constraints independent of the choice of $B$. In contrast, for SOS2 constraints, where $|I(j)| = 2$ for some $j \in J$, formulation (46) can be invalid for some choices of $B$. This is illustrated by the following example.

**Example 3.3.** *Let* $J = \{0, \ldots, 4\}$ *and* $(\lambda_j)_{j=0}^4 \in \Delta^J$ *be SOS2 constrained. Formulation* (46) *for this case with* $B = B^*$ *is*

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_0 + \lambda_1 \leq x_1, \quad \lambda_3 + \lambda_4 \leq 1 - x_1, \quad \lambda_0 \leq x_2, \quad \lambda_4 \leq 1 - x_2$$

*which has the feasible solution* $\lambda_0 = 1/2$, $\lambda_2 = 1/2$, $\lambda_1 = \lambda_3 = \lambda_4 = 0$, $x_1 = x_2 = 1$ *that does not comply with SOS2 constraints. However, the formulation can be made valid by adding constraints*

$$\lambda_2 \leq x_1 + x_2, \quad \lambda_2 \leq 2 - x_1 - x_2. \tag{47}$$

For any $B$ we can always correct formulation (46) for SOS2 constraints by adding a

number of extra linear inequalities, but with a careful selection of $B$ the validity of the model can be preserved without the need for additional constraints.

**Definition 3.3** (SOS2 Compatible Function)**.** *A function* $B : \{1, \dots, n\} \to \{0,1\}^{\log_2(n)}$ *is* compatible *with an SOS2 constraint on* $(\lambda_j)_{j=0}^n \in \mathbb{R}_+^{n+1}$ *if it is injective and for all* $i \in \{1, \dots, n-1\}$ *the vectors* $B(i)$ *and* $B(i+1)$ *differ in at most one component.*

**Theorem 3.4.** *If* $B$ *is an SOS2 compatible function then* (46) *is valid for SOS2 constraints.*

*Proof.* For SOS2 constraints we have that $I = \{1, \dots, n\}$, $J = \{0, \dots, n\}$ and $S_i = \{i-1, i\}$ for each $i \in I$. This implies that $I(0) = \{1\}$ and $I(n) = \{n\}$. Then, in a similar way to the proof of Theorem 3.2 for SOS1 constraints, we have that for $j \in \{0, n\}$ formulation (46) imposes $\lambda_j = 0$ for all $x \neq B(j)$.

In contrast, for $j \in J \setminus \{0, n\}$ we have $I(j) = \{j, j+1\}$ and hence $J^+(l, B) = \{j \in J : l \in \sigma(B(j)) \cap \sigma(B(j+1))\}$ and $J^0(l, B) = \{j \in J : l \notin \sigma(B(j)) \text{ and } l \notin \sigma(B(j+1))\}$. Using the fact that $B$ is SOS2 compatible we have that, in formulation (46), $\lambda_j = 0$ for all $x \notin \{B(j), B(j+1)\}$. □

The following example illustrates how an SOS2 compatible function yields a valid formulation.

**Example 3.3.** continued *Let* $B^0(1) = (1,0)^T$, $B^0(2) = (1,1)^T$, $B^0(3) = (0,1)^T$ *and* $B^0(4) = (0,0)^T$. *Formulation* (46) *with* $B = B^0$ *for the same SOS2 constraints is*

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}$$

$$\lambda_0 + \lambda_1 \leq x_1, \quad \lambda_3 + \lambda_4 \leq (1 - x_1) \tag{48}$$

$$\lambda_2 \leq x_2, \quad \lambda_0 + \lambda_4 \leq (1 - x_2). \tag{49}$$

Finally, the following lemma shows that an SOS2 compatible function can always be constructed.

**Lemma 3.5.** *For any $n \in \mathbb{Z}_+$ there exists a compatible function for SOS2 constraints on $(\lambda)_{j=0}^n$.*

*Proof.* We construct an SOS2 compatible function $\bar{B} : \{1, \dots, 2^r\} \to \{0,1\}^r$ inductively on $r$. The case $r = 1$ follows immediately. Now assume that we have an SOS2 compatible function $\bar{B} : \{1, \dots, 2^r\} \to \{0,1\}^r$. We define $\tilde{B} : \{1, \dots, 2^{r+1}\} \to \{0,1\}^{r+1}$ as

$$\tilde{B}(i)_l := \begin{cases} \bar{B}(i)_l & \text{if } i \leq 2^r \\ \bar{B}(2^{r+1} - i + 1)_l & \text{o.w.} \end{cases} \quad \forall l \in \{1, \dots, r\}, \quad \tilde{B}(i)_{r+1} := \begin{cases} 1 & \text{if } i \leq 2^r \\ 0 & \text{o.w.} \end{cases},$$

which is also SOS2 compatible. □

The function from the proof of Lemma 3.5 is not the only possible SOS2 compatible function. In fact, Definition 3.3 is equivalent to requiring $(B(i))_{i=1}^n$ to be a *reflected binary* or *Gray code* [138] and the construction from Lemma 3.5 corresponds to a version of this code that is usually called the *standard reflected Gray code*. Definition 3.3 is also equivalent to requiring $(B(i))_{i=1}^n$ to be a Hamiltonian path on the hypercube.

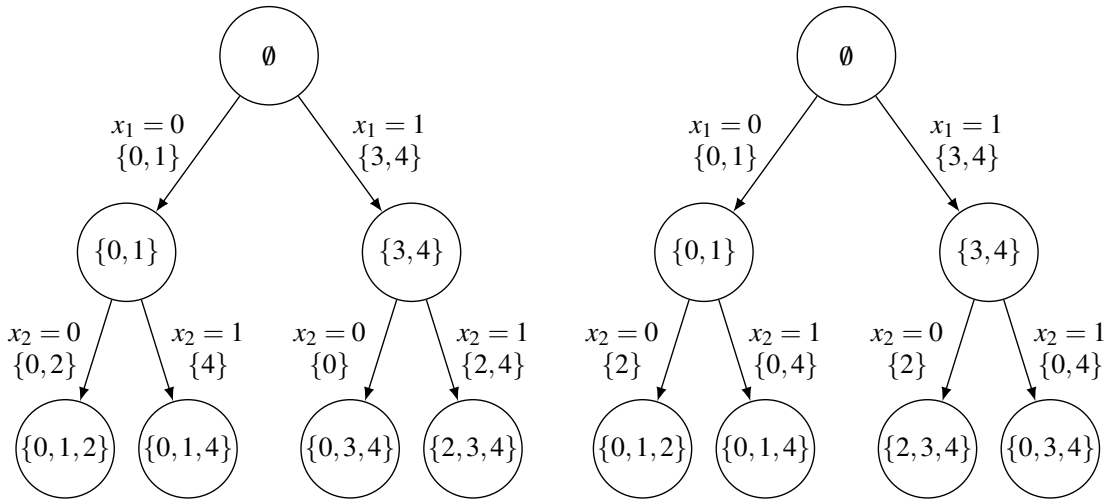
### *3.3 Branching and Logarithmic Size Formulations*

We have seen that fixing the binary variables of (46) provides a systematic procedure for enforcing $\lambda \in Q(S_i)$. In this section we exploit the relation between this procedure and specialized branching schemes to extend the formulation to a more general framework.

We can identify each vector in $\{0,1\}^{\log_2 |I|}$ with a leaf in a binary tree with $\log_2 |I|$ levels such that each component corresponds to a level and the value of that component indicates the selected branch in that level. Then, using function $B$ we can identify each set $Q(S_i)$ with a leaf in the binary tree and we can interpret each of the $\log_2 |I|$ variables as the execution of a branching scheme on sets $Q(S_i)$. The formulations in Example 3.3 illustrate this idea.

In formulation (46) with $B = B^0$ the branching scheme associated with $x_1$ sets $\lambda_0 = \lambda_1 = 0$ when $x_1 = 0$ and $\lambda_3 = \lambda_4 = 0$ when $x_1 = 1$, which is equivalent to the traditional SOS2 constraint branching of [12] whose dichotomy is fixing to zero variables to the "left of" (smaller than) a certain index in one branch and to the "right" (greater) in the other.

In contrast, the scheme associated with $x_2$ sets $\lambda_2 = 0$ when $x_2 = 0$ and $\lambda_0 = \lambda_4 = 0$ when $x_2 = 1$, which is different from the traditional branching as its dichotomy can be interpreted as fixing variables in the "center" and on the "sides" respectively. If we use function $B^*$ instead we recover the traditional branching. The drawback of the $B^*$ scheme is that the second level branching cannot be implemented independently of the first level branching using linear inequalities. For $B^0$ the branch alternatives associated with $x_2$ are implemented by (49), which only include binary variable $x_2$. In contrast, for $B^*$ one of the branching alternatives requires additional constraints (47) which involve both $x_1$ and $x_2$. The binary tree associated with the model for $B^*$ and $B^0$ are shown in Figure 9, where the arc labels indicate the values taken by the binary variables and the indices of the $\lambda$ variables which are fixed to zero because of this and the node labels indicate the indices of the $\lambda$ variables that are set to zero because of the cumulative effect of the binary variable fixing. The main difference in the trees is that for $B = B^*$ the effect on the $\lambda$ variables of fixing $x_2$ to a particular value depends on the value previously assigned to $x_1$ while for $B = B^0$ this effect is independent of the previous assignment to $x_1$.




(a) $B = B^*$. (b) $B = B^0$.

**Figure 9:** Two level binary trees for example 3.3.

This example illustrates that a sufficient condition for modeling (42) with a logarithmic number of binary variables and extra constraints is to have a binary branching scheme for

$\lambda \in \bigcup_{i \in I} Q(S_i)$ with a logarithmic number of dichotomies and for which each dichotomy can be implemented independently. This condition is formalized in the following definition.

**Definition 3.6.** (Independent Branching Scheme) $\{L_k, R_k\}_{k=1}^{d}$ *with* $L_k, R_k \subset J$ *is an independent branching scheme of depth* $d$ *for disjunctive constraint* (42) *if*

$$\bigcup_{i \in I} Q(S_i) = \bigcap_{k=1}^{d} \big(Q(L_k) \cup Q(R_k)\big). \tag{50}$$

This definition can then be used in the following theorem and immediately gives a sufficient condition for modeling with a logarithmic number of variables and constraints.

**Theorem 3.7.** *Let* $\{Q(S_i)\}_{i \in I}$ *be a finite family of polyhedra of the form* (43) *and* $\{L_k, R_k\}_{k=1}^{\log_2 |I|}$ *be an independent branching scheme for* $\lambda \in \bigcup_{i \in I} Q(S_i)$. *Then*

$$\lambda \in \Delta^J, \quad \sum_{j \notin L_k} \lambda_j \leq x_k, \quad \sum_{j \notin R_k} \lambda_j \leq (1 - x_k), \quad x_k \in \{0,1\} \quad \forall k \in L(|I|) \tag{51}$$

*is a valid formulation for* (42) *with* $\log_2 |I|$ *binary variables and* $2 \log_2 |I|$ *extra constraints.*

Formulation (46) with $B = B^0$ in Example 3.3 illustrates how an SOS2 compatible function induces an independent branching scheme for SOS2 constraints. In general, given an SOS2 compatible function $B : \{1, \ldots, n\} \to \{0,1\}^{\log_2(n)}$ the induced independent branching is given by $L_k = J \setminus J^+(k, B)$, $R_k = J \setminus J^0(l, B)$ for all $k \in \{1, \ldots, n\}$.

Formulation (51) in Theorem 3.7 can be interpreted as a way of implementing a specialized branching scheme using binary variables. Similar techniques for implementing specialized branching schemes have been given in [4] and [120], but the resulting models require at least a linear number of binary variables. To the best of our knowledge the first independent branching schemes of logarithmic depth for the case in which polytopes $Q(S_i)$ contain more than one point are the ones for SOS1 constraints from Theorem 3.2 and for SOS2 constraints induced by an SOS2 compatible function.

Formulation (51) can be obtained by algebraic simplifications from formulation (39) of (42) rewritten as the conjunction of two-term polyhedral disjunctions. Both the simplifications and the rewrite can result in a significant reduction in the tightness of the linear programming relaxation of (51) [7, 8, 27, 65]. Fortunately, as the following theorem shows, the restriction to $\Delta^J$ makes (51) as tight as any other mixed integer formulation for (42).

**Theorem 3.8.** *Let $P_\lambda$ and $Q_\lambda$ be the projection onto the $\lambda$ variables of the LP relaxation of formulation* (51) *and of any other mixed integer programming formulation of* (42) *respectively. Then $P_\lambda = \operatorname{conv}\left(\bigcup_{i\in I} Q(S_i)\right)$ and hence $P_\lambda \subseteq Q_\lambda$.*

*Proof.* Without loss of generality $\bigcup_{i\in I} S_i = J$ and hence for every $j \in J$ there is a $i \in I$ such that $j \in S_i$. Using this, it follows that $P_\lambda = \Delta^J = \operatorname{conv}\left(\bigcup_{i\in I} Q(S_i)\right)$. The relation with other mixed integer programming formulations follows directly from Theorem 3.1 of [67]. □

Theorem 3.8 might not be true if we do not use ground set $\Delta^J$, but this restriction is not too severe as it includes a popular way of modeling piecewise linear functions. We explore this modeling in Section 3.4 and the potential loss of Theorem 3.8 when using a different ground set in Section 3.5.

We finally study the effect on formulation (51) of dropping the assumption that $|I|$ is a power of two. As mentioned in Section 3.2, if $|I|$ is not a power of two we can complete $I$ to an index set of size $2^{\lceil \log_2 |I| \rceil}$ without changing (42). If we now construct a formulation that is of logarithmic size with respect to the completed index set we obtain a formulation that is still of logarithmic order with respect to the original index set. For instance, if $I$ is not a power of two we can complete it and apply Theorem 3.1 to obtain a formulation with $\lceil \log_2 |I| \rceil < \log_2 |I| + 1$ binary variables and $2^{\lceil \log_2 |I| \rceil} < 2|I|$ extra constraints with respect to the original index set $I$. This is illustrated in the following example.

**Example 3.4.** *Let $J = \{1, \ldots, 3\}$, $(\lambda_j)_{j=1}^3 \in \Delta^J$ be SOS1 constrained. In this case $I = \{1, \ldots, 3\}$ and $S_i = \{i\}$ for all $i \in I$. We can complete $I$ so that $|I|$ is a power of two by letting $I = \{1, \ldots, 4\}$ and $S_4 = \emptyset$. Using $B = B^*$ from example 1 formulation* (45) *for the completed constraint is*

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_2 + \lambda_3 \leq 2 - x_1 - x_2, \quad \lambda_1 + \lambda_3 \leq 1 - x_1 + x_2,$$
$$\lambda_1 + \lambda_2 \leq 1 + x_1 - x_2, \quad \lambda_1 + \lambda_2 + \lambda_3 \leq x_1 + x_2.$$

Formulation (51) deals with the requirement that $|I|$ is a power of two somewhat differently. It is clear that (51) does not have this requirement explicitly as it only needs the

existence of an independent branching scheme. Fortunately, if a family of constraints has an independent branching scheme when $|I|$ is a power of two we can easily construct an independent branching scheme for the cases in which $|I|$ is not a power of two. This is illustrated in the following example.

**Example 3.5.** *Let $\{\overline{L}_k, \overline{R}_k\}_{k=1}^{\lceil \log_2 n \rceil}$ be an independent branching scheme for an SOS2 constraint on $(\lambda_j)_{j=0}^{\overline{n}} \in \Delta^{\overline{J}}$ for $\overline{n} := 2^{\lceil \log_2 n \rceil}$ and $\overline{J} = \{0, \ldots, \overline{n}\}$. Then $\{L_k, R_k\}_{k=1}^{\lceil \log_2 n \rceil}$ defined by*

$$L_k := \overline{L}_k \cap \{0, \ldots, n\}, \quad R_k := \overline{R}_k \cap \{0, \ldots, n\} \quad \forall k \in \{1, \ldots, \lceil \log_2 n \rceil\} \tag{52}$$

*is an independent branching scheme for an SOS2 constraint on $(\lambda_j)_{j=0}^{n} \in \Delta^J$ for $J = \{0, \ldots, n\}$.*

*For example, for $n = 3$ and $\overline{n} = 4$, SOS2 compatible function $B^0$ from example 3 yields the independent branching scheme for SOS2 on $(\lambda_j)_{j=0}^{4} \in \Delta^{\overline{J}}$ given by $\overline{L}_1 := \{2, 3, 4\}$, $\overline{R}_1 := \{0, 1, 2\}$, $\overline{L}_2 := \{0, 1, 3, 4\}$ and $\overline{R}_2 := \{1, 2, 3\}$. By restricting this scheme to $\{0, \ldots, 3\}$ we get the independent branching scheme for SOS2 on $(\lambda_j)_{j=0}^{3} \in \Delta^J$ given by $L_1 := \{2, 3\}$, $R_1 := \{0, 1, 2\}$, $L_2 := \{0, 1, 3\}$ and $R_2 := \{1, 2, 3\}$. This scheme yields the following formulation of SOS2 on $(\lambda_j)_{j=0}^{3} \in \Delta^J$.*

$$\begin{aligned} &\lambda \in \Delta^J, \quad x_1, x_2 \in \{0, 1\} \\ &\lambda_0 + \lambda_1 \leq x_1, \quad \lambda_3 \leq (1 - x_1) \\ &\lambda_2 \leq x_2, \quad \lambda_0 \leq (1 - x_2). \end{aligned}$$

Note that this formulation can also be obtained by completing the constraint to $I = \{1, \ldots, 4\}$ by adding $S_4 = \emptyset$ and using formulation (46) for $B = B^0$ from example 3. We could show the validity of this procedure without referring to independent branching schemes by proving an analog to Theorem 3.4 for the case in which $|I|$ is not a power of two.

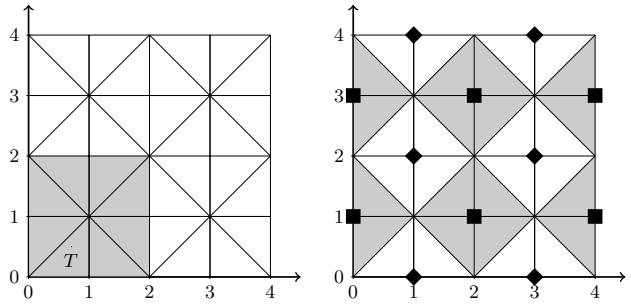
### *3.4 Modeling Nonseparable Piecewise Linear Functions*

In this section we use Theorem 3.7 to construct a model for non-separable piecewise linear functions of two variables that use a number of binary variables and extra constraints

logarithmic in the number of linear pieces of the functions. We also extend this formulation to functions of $n$ variables, in which case the formulation is slightly larger, but still asymptotically logarithmic for fixed $n$.

As described in Section 1.1.2.3 of Chapter 1, imposing SOS2 constraints on $(\lambda_j)_{j=0}^n \in \Delta^J$ with $J = \{0, \ldots, n\}$ is a popular way of modeling a one variable piecewise-linear function which is linear in $n$ different intervals [72, 73, 82, 96, 127]. This approach has been extended to non-separable piecewise linear functions in [82, 96, 127, 140]. For functions of two variables this approach can be described as follows.

We assume that for an even integer $w$ we have a continuous function $f : [0, w]^2 \to \mathbb{R}$ which we want to approximate by a piecewise linear function. A common approach is to partition $[0, w]^2$ into a number of triangles and approximate $f$ with a piecewise linear function that is linear in each triangle. One possible triangulation of $[0, w]^2$ is the $\mathbf{J}_1$ or "Union Jack" triangulation [125] which is depicted in Figure 10(a) for $w = 4$. The $\mathbf{J}_1$ triangulation of $[0, w]^2$ for any even $w$ is obtained by adding copies of the 8 triangles shaded gray in Figure 10(a). This yields a triangulation with $2w^2$ triangles.




(a) Example of "Union Jack" Triangulation (b) Triangle selecting branching

**Figure 10:** Triangulations

We use this triangulation to approximate $f$ with a piecewise linear function that we denote by $g$. Let $I$ be the set of all the triangles of the $\mathbf{J}_1$ triangulation of $[0, w]^2$ and let $S_i$ be the vertices of triangle $i$. For example, in Figure 10(a), the vertices of the triangle

labeled $T$ are $S_T := \{(0,0),(1,0),(1,1)\}$. A valid model for $g(y)$ [82, 96, 127] is

$$\sum_{j\in J}\lambda_j = 1, \quad y = \sum_{j\in J} v_j\lambda_j, \quad g(y) = \sum_{j\in J} f(v_j)\lambda_j \tag{53a}$$

$$\lambda \in \bigcup_{i\in I} Q(S_i) \subset \Delta^J, \tag{53b}$$

where $J := \{0,\dots,w\}^2$, $v_j = j$ for $j \in J$. This model becomes a traditional model for one variable piecewise linear functions when we restrict it to one coordinate of $[0,w]^2$ by setting $y_2 = 0$ and $\lambda_{(s,t)} = 0$ for all $0 \le s \le w$, $1 \le t \le w$.

To obtain a mixed integer formulation of (53) with a logarithmic number of binary variables and extra constraints it suffices to construct an independent binary branching scheme of logarithmic depth for (53b) and use formulation (51). Binary branching schemes for (53b) with a similar triangulation have been developed in [127] and [96], but they are either not independent or have too many dichotomies. We adapt some of the ideas of these branching schemes to develop an independent branching scheme for the two-dimensional $\mathbf{J}_1$ triangulation. Our independent branching scheme will basically select a triangle by forbidding the use of vertices in $J$. We divide this selection into two phases. We first select the square in the grid induced by the triangulation and we then select one of the two triangles inside this square.

To implement the first branching phase we use the observation made in [96, 127] that selecting a square can be achieved by applying SOS2 branching to each component. To make this type of branching independent it then suffices to use the independent SOS2 branching induced by an SOS2 compatible function. This results in the set of constraints

$$\sum_{v_2=0}^{w}\sum_{v_1\in J_2^+(l,B,w)}\lambda_{(v_1,v_2)} \le x_l^1, \quad \sum_{v_2=0}^{w}\sum_{v_1\in J_2^0(l,B,w)}\lambda_{(v_1,v_2)} \le 1 - x_l^1,$$

$$x_l^1 \in \{0,1\} \quad \forall l \in L(w), \tag{54a}$$

$$\sum_{v_1=0}^{w}\sum_{v_2\in J_2^+(l,B,w)}\lambda_{(v_1,v_2)} \le x_l^2, \quad \sum_{v_1=0}^{w}\sum_{v_2\in J_2^0(l,B,w)}\lambda_{(v_1,v_2)} \le 1 - x_l^2,$$

$$x_l^2 \in \{0,1\} \quad \forall l \in L(w), \tag{54b}$$

where $B$ is an SOS2 compatible function and $J_2^+(l,B,w)$, $J_2^0(l,B,w)$ are the specializations of $J^+(l,B)$, $J^0(l,B)$ for SOS2 constraints on $(\lambda_j)_{j=0}^w$. Constraints (54a) and binary variables

$x_l^1$ implement the independent SOS2 branching for the first coordinate and (54b) and binary variables $x_l^2$ do the same for the second one.

To implement the second phase we use the branching scheme depicted in Figure 10(b) for the case $w = 4$. The dichotomy of this scheme is to select the triangles colored white in one branch and the ones colored gray in the other. For general $w$, this translates to forbidding the vertices $(v_1, v_2)$ with $v_1$ even and $v_2$ odd in one branch (square vertices in the figure) and forbidding the vertices $(v_1, v_2)$ with $v_1$ odd and $v_2$ even in the other (diamond vertices in the figure). This branching scheme selects exactly one triangle of every square in each branch and induces the set of constraints

$$\sum_{(v_1,v_2)\in L} \lambda_{(v_1,v_2)} \leq y_0, \qquad \sum_{(v_1,v_2)\in R} \lambda_{(v_1,v_2)} \leq 1 - y_0, \quad y_0 \in \{0,1\}, \tag{55}$$

where $L = \{(v_1, v_2) \in J : \ v_1 \text{ is even and } v_2 \text{ is odd}\}$ and $R = \{(v_1, v_2) \in J : \ v_1 \text{ is odd and } v_2 \text{ is even}\}$. When $w$ is a power of two the resulting formulation has exactly $\log_2 \mathcal{T}$ binary variables and $2 \log_2 \mathcal{T}$ extra constraints where $\mathcal{T}$ is the number of triangles in the triangulation. We illustrate the formulation with the following example.

**Example 3.6.** *Constraints* (54)–(55) *for* $w = 2$ *are*

$$\begin{aligned} \lambda_{(0,0)} + \lambda_{(0,1)} + \lambda_{(0,2)} \leq x_{(1,1)}, \quad & \lambda_{(2,0)} + \lambda_{(2,1)} + \lambda_{(2,2)} \leq 1 - x_{(1,1)} \\ \lambda_{(0,0)} + \lambda_{(1,0)} + \lambda_{(2,0)} \leq x_{(2,1)}, \quad & \lambda_{(0,2)} + \lambda_{(1,2)} + \lambda_{(2,2)} \leq 1 - x_{(2,1)} \\ \lambda_{(0,1)} + \lambda_{(2,1)} \leq x_0, \quad & \lambda_{(1,0)} + \lambda_{(1,2)} \leq 1 - x_0. \end{aligned}$$

A portion of the associated branching scheme is shown in Figure 11. The shaded triangles inside the nodes indicates the triangles forbidden by the corresponding assignment of the binary variables.

The restriction to the first coordinate of $[0, w]^2$ yields a logarithmic formulation for piecewise linear functions of one variable that only uses one of the SOS2 branchings and does not use the triangle selecting branching. Furthermore, under some mild assumptions,

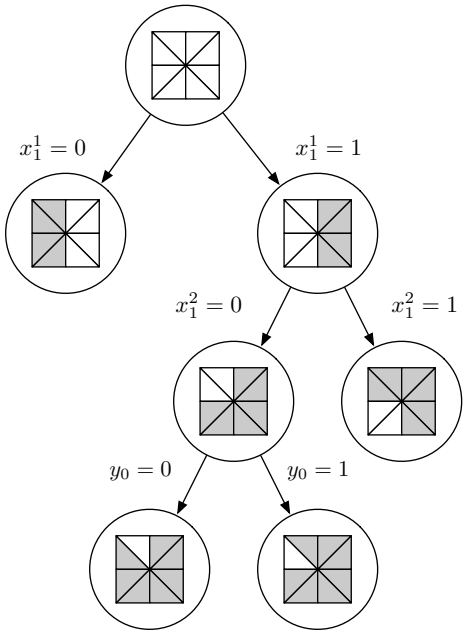



**Figure 11:** Partial B&B tree for Example 3.6

the model can be extended to non-uniform grids by selecting different values of $v_j$. This last extension is described in Section 4.3.2.2 of Chapter 4.

The extension of the formulation to functions of $n$ variables is direct from the definition of the $n$-dimensional $\mathbf{J}_1$ triangulation [125]. For $D = [0, w]^n$ with $w$ an even integer the vertex set of the triangulation is defined to be $\{0, \ldots, w\}^n$ and the triangulation is composed by the finite family of simplices defined as follows. Let $N = \{1, \ldots, n\}$, $\mathcal{V}^0 = \{v \in \{0, \ldots, w\}^n : v_i \text{ is odd}, \forall i \in N\}$, $\text{Sym}(N)$ be the group of all permutations on $N$ and $e^i$ be the $i$-th unit vector of $\mathbb{R}^n$. For each $(v^0, \pi, s) \in \mathcal{V}^0 \times \text{Sym}(N) \times \{-1, 1\}^n$ we define $j_1(v^0, \pi, s)$ to be the simplex whose extreme points are $\{y^i\}_{i=0}^n$ where $y^i = y^{i-1} + s_{\pi(i)} e^{\pi(i)}$ for each $i \in N$. The $\mathbf{J}_1$ triangulation of $D = [0, w]^n$ is given by all the simplices $j_1(v^0, \pi, s)$. By letting $J = \{0, \ldots, w\}^n$ and $I$ be the set of triangles of the $\mathbf{J}_1$ of $D = [0, w]^n$ we have that (53) is a model for the piecewise linear approximation $g$ of function $f : [0, w]^n \to \mathbb{R}$. For this case, to implement the independent branching scheme for (53b) we can use the fact that indices $v^0$ and $s$ of the simplices determines the hypercube in which the simplex is contained and index $\pi$ determines the selection of one of the $n!$ simplices contained in a given hypercube (For example for the triangulation in Figure 1(a), the simplices for $v^0 = (1, 1)$ and $s = (-1, -1)$

are the two triangles contained in box $[0,1]^2$ and the triangle labeled $T$ corresponds to the permutation $\pi(1) = 2$, $\pi(2) = 1$). Then to select the hypercube we can again apply independent SOS2 branching for each component which yields the constraints given by

$$\sum_{v \in \tilde{J}_2^+(l,B,w,k)} \lambda_v \leq x_l^k, \qquad \sum_{v \in \tilde{J}_2^0(l,B,w,k)} \lambda_v \leq 1 - x_l^k, \quad x_l^k \in \{0,1\} \quad \forall l \in L(w), \forall k \in N \tag{56}$$

where $\tilde{J}_2^+(l,B,w,k) = \{v \in J : v_k \in J_2^+(l,B,w)\}$ and $\tilde{J}_2^0(l,B,w,k) = \{v \in J : v_k \in J_2^0(l,B,w)\}$. To select a permutation $\pi$ it suffices to select between $\pi^{-1}(r) < \pi^{-1}(s)$ or $\pi^{-1}(r) > \pi^{-1}(s)$ for each $r, s \in N$, $r < s$. If we select a permutation with $\pi^{-1}(r) < \pi^{-1}(s)$ we have that no vertex $v$ of the resulting triangulation will have an odd $v_r$ component and even $v_s$ component. In contrast, if we select a permutation with $\pi^{-1}(s) < \pi^{-1}(r)$ we have that no vertex $v$ of the resulting triangulation will have an even $v_r$ component and odd $v_s$ component. Hence to select a simplex if suffices to apply the triangle selection branching depicted in Figure 10(b) to each pair of indices $r, s \in N$, $r < s$ which yields the constraints given by

$$\sum_{v \in L(r,s)} \lambda_v \leq y_{(r,s)}, \qquad \sum_{v \in R(r,s)} \lambda_v \leq 1 - y_{(r,s)}, \quad y_{(r,s)} \in \{0,1\} \quad \forall r, s \in N,\, r < s \tag{57}$$

where $L(r,s) = \{v \in J : v_r \text{ is even and } v_s \text{ is odd}\}$ and $R = \{v \in J : v_r \text{ is odd and } v_s \text{ is even}\}$. The resulting formulation has $L := n\lceil \log_2 w \rceil + n(n-1)/2$ binary variables (and twice as many extra constraints) and the $\mathbf{J}_1$ triangulation has $\mathcal{T} := w^n n!$ simplices. In contrast to the two dimensional case, it is not clear how to explicitly relate these two numbers even for the case when $w$ is a power of two. However we can see that $L$ grows asymptotically as $\log_2 \mathcal{T}$ only when $n$ is fixed. More specifically, for fixed $n$ we have $L \sim \log_2 \mathcal{T}$ (i.e. $\lim_{w\to\infty} L/\log_2 \mathcal{T} = 1$), but for fixed $w$ we have $\log_2 \mathcal{T} \in o(L)$ (i.e. $\lim_{n\to\infty} \log_2 \mathcal{T}/L = 0$).

### *3.5 Extension of the Model to Ground Set* $[0,1]^J$

We replace $\lambda \in \Delta^J$ in definition (43) of $Q(S_i)$ with the box constraint $\lambda \in [0,1]^J$ to obtain $\overline{Q}(S_i) = \left\{\lambda \in [0,1]^J : \lambda_j = 0 \,\forall\, j \notin S_i\right\}$. We have that an independent branching $\{L_k, R_k\}_{k=1}^d$ for (42) is also an independent branching for

$$\lambda \in \bigcup_{i \in I} \overline{Q}(S_i) \tag{58}$$

since

$$\bigcup_{i\in I} \overline{Q}(S_i) = \bigcap_{k=1}^{d} \left(\overline{Q}(L_k) \cup \overline{Q}(R_k)\right). \tag{59}$$

However, to preserve validity formulation (51) needs to be modified to

$$\lambda \in [0,1]^J \tag{60a}$$

$$\sum_{j\notin L_k} \lambda_j \leq |J \setminus L_k|\, x_k, \quad \sum_{j\notin R_k} \lambda_j \leq |J \setminus R_k|\,(1-x_k), \quad x_k \in \{0,1\} \quad \forall k \in \{1,\ldots,d\}. \tag{60b}$$

This formulation still has $d$ binary variables and $2d$ extra constraints, but Theorem 3.8 is no longer true for this formulation.

To understand the potential sources of weakness of formulation (60) we study how this formulation can be constructed from the standard disjunctive programming formulation of (58) in three steps, two of which have the potential for weakening the formulation. The first step is to use identity (59) to reduce the formulation of (58) to the formulation of

$$\lambda \in \overline{Q}(L_k) \cup \overline{Q}(R_k) \tag{61}$$

for each $k \in \{1,\ldots,d\}$. The second step is to eliminate the duplicated continuous variables of formulation (39) for (61) in the following way. Formulation (39) for (61) is given by

$$\lambda_j^{1,k}, \lambda^{2,k} \in \mathbb{R}_+^{|J|}, \;\; x_k \in \{0,1\} \tag{62a}$$

$$\lambda_j^{1,k} \leq (1-x_k) \quad \forall j \in L_k, \quad \lambda_j^{1,k} \leq 0 \quad \forall j \notin L_k \tag{62b}$$

$$\lambda_j^{2,k} \leq x_k \qquad \forall j \in R_k, \quad \lambda_j^{2,k} \leq 0 \quad \forall j \notin R_k \tag{62c}$$

$$\lambda = \lambda^{1,k} + \lambda^{2,k}. \tag{62d}$$

Using (62d) we can eliminate variables $\lambda^{1,k}, \lambda^{2,k}$ to obtain the formulation of (61) given by

$$\lambda \in [0,1]^J, \quad x_k \in \{0,1\} \tag{63a}$$

$$\lambda_j \leq x_k \qquad \forall j \notin L_k \tag{63b}$$

$$\lambda_j \leq (1-x_k) \qquad \forall j \notin R_k. \tag{63c}$$

The third and final step is to aggregate constraints (63b)–(63c) and combine the resulting formulation of (61) for all $k \in \{1,\ldots,d\}$ to obtain (60).

With regard to the first step, we have that (59) shows how an independent branching scheme rewrites disjunctive constraint (42) from its *disjunctive normal form* (DNF) as the union of polyhedra (left hand side) to a conjunction of two-term polyhedral disjunctions (right hand side). It is well known that this rewrite can significantly reduce the tightness of mixed integer programming formulations [7]. More specifically, Theorem 3.1 of [67] tells us that if we directly formulate constraint (58) the best we can hope is for the projection onto the original $\lambda$ variables of the LP relaxation of our formulation to be equal to $\text{conv}(\bigcup_{i\in I} \overline{Q}(S_i))$. In contrast, if we construct a formulation for constraints (61) for each $k \in \{1,\dots,d\}$ and then combine them, the best we can hope is for the projection onto the original $\lambda$ variables of the LP relaxation of our formulation to be equal to $\bigcap_{k=1}^{d} \text{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k))$. Because the convex hull and intersection operations usually do not commute we only have

$$\text{conv}\left(\bigcup_{i\in I} \overline{Q}(S_i)\right) \subset \bigcap_{k=1}^{d} \text{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k)) \tag{64}$$

and we can expect strict containment resulting in the first formulation being stronger. This is illustrated in the following example.

**Example 3.7.** *Let $J = \{0,\dots,4\}$ and $(\lambda_j)_{j=0}^4 \in \Delta^J$ be SOS2 constrained. We then have $S_1 = \{0,1\}$, $S_2 = \{1,2\}$, $S_3 = \{2,3\}$, $S_4 = \{3,4\}$ and using PORTA [33] we get that*

$$\text{conv}\left(\bigcup_{i=1}^{4} \overline{Q}(S_i)\right) = \Big\{(\lambda_j)_{j=0}^4 \in [0,1]^5 : \lambda_1 + \lambda_4 \le 1, \quad \lambda_1 + \lambda_3 \le 1, \quad \lambda_0 + \lambda_3 \le 1, \\ \lambda_0 + \lambda_2 + \lambda_4 \le 1\Big\}. \tag{65}$$

*If we let $d = 1$, $L_1 = \{2,3,4\}$, $R_1 = \{0,1,2\}$, $L_2 = \{0,1,3,4\}$ and $R_2 = \{1,2,3\}$ we have $\bigcup_{i=1}^4 \overline{Q}(S_i) = \big(\overline{Q}(L_1) \cup \overline{Q}(R_1)\big) \cap \big(\overline{Q}(L_2) \cup \overline{Q}(R_2)\big)$. Again using PORTA we get that*

$$\text{conv}\left(\overline{Q}(L_1) \cup \overline{Q}(R_1)\right) = \Big\{(\lambda_j)_{j=0}^4 \in [0,1]^5 : \lambda_2 \le 1, \quad \lambda_1 + \lambda_4 \le 1, \quad \lambda_1 + \lambda_3 \le 1, \\ \lambda_0 + \lambda_4 \le 1, \quad \lambda_0 + \lambda_3 \le 1\}$$

*and*

$$\operatorname{conv}\left(\overline{Q}(L_2) \cup \overline{Q}(R_2)\right) = \Big\{(\lambda_j)_{j=0}^4 \in [0,1]^5 \,:\, \lambda_3 \leq 1, \quad \lambda_1 \leq 1, \quad \lambda_2 + \lambda_4 \leq 1,$$
$$\lambda_0 + \lambda_2 \leq 1, \quad \lambda_3 + \lambda_4 - \lambda_0 - \lambda_1 \leq 1\}.$$

*Clearly* $(1/2, 1/2, 1/2, 1/2, 1/2) \in \operatorname{conv}\left(Q(L_1) \cup Q(R_1)\right) \cap \operatorname{conv}\left(Q(L_2) \cup Q(R_2)\right)$*, but from* (65) *we have* $(1/2, 1/2, 1/2, 1/2, 1/2) \notin \operatorname{conv}(\bigcup_{i=1}^4 Q(S_i))$*. Hence*

$$\operatorname{conv}\left(\bigcup_{i=0}^{4} \overline{Q}(S_i)\right) \subsetneq \bigcap_{k=1}^{2} \operatorname{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k)).$$

This source of weakness could be avoided by applying techniques from [7] at the expense of increasing the number of continuous variables.

With respect to the second step, it is well known that eliminating the multiple copies of the continuous variables in formulation (39) can result in a weaker formulation [8, 27, 65]. Fortunately, as the following theorem shows, for constraints of the form (42) or (58) eliminating the multiple copies of the continuous variables does not make the formulations weaker.

**Theorem 3.9.** *Let* $P_\lambda$ *be the projection onto the* $\lambda$ *variables of the LP relaxation of formulation* (44) *for* (42) *and let* $\overline{P}_\lambda$ *the projection onto the* $\lambda$ *variables of the LP relaxation of the formulation of* (58) *given by*

$$\lambda \in [0,1]^J, \quad \lambda_j \leq \sum_{i \in I(j)} x_i \quad \forall j \in J, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I. \tag{66}$$

*Then* $P_\lambda = \operatorname{conv}\left(\bigcup_{i \in I} Q(S_i)\right)$ *and* $\overline{P}_\lambda = \operatorname{conv}\left(\bigcup_{i \in I} \overline{Q}(S_i)\right)$*. In particular the projections onto the* $\lambda$ *variables of the LP relaxations of formulations* (62) *and* (63) *are equal to* $\operatorname{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k))$*.*

*Proof.* For $P_\lambda$ the result follows directly from Theorem 3.8. For $\overline{P}_\lambda$ the result follows directly from Section 3.1 of [65] because $\bigcup_{i \in I} \overline{Q}(S_i)$ is the union of multidimensional intervals as defined in that section. □ □

Theorem 3.9 shows that the traditional formulations for SOS1 and SOS2 constraints are as tight as possible, which could explain their success. In addition, Theorem 3.9 shows that the second step does not weaken the formulation as we get the following corollary.

**Corollary 3.10.** *The projection onto the $\lambda$ variables of the LP relaxation of the formulation given by* (63) *for all $k \in \{1, \dots, d\}$ is $\bigcap_{k=1}^{d} \operatorname{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k))$.*

Finally, with respect to the third step, it is well known that a weaker integer programming formulations can result from aggregating constraints. As expected it is also easy to construct examples where formulation (63) is stronger than formulation (60) (the example for the strict containment in (64) also works here). Of course, this source of weakness can be avoided by simply choosing formulation (63) instead of (60) at the expense of increasing the number of constraints from $2d$ to at most $|J|d$.

### *3.6 Computational Results*

In this section we computationally test the logarithmic models for piecewise linear functions of one and two variables against some other existing models. For a set of transportation problems with piecewise linear cost functions, the logarithmic models provide a significant advantage in almost all of our experiments.

We denote the model for piecewise linear functions of one and two variables from Section 3.4 by Log. From the traditional models we selected the so called *incremental* and *multiple choice* models. The incremental model for one variable functions appears as early as [40, 41, 95], was extended to functions of several variables in [140] and it has been recently shown to have favorable integrality and tightness properties [36, 105, 118]. This model is described in detail in Section 4.3.4 of Chapter 4 and we denote it by Inc. The multiple choice model appears in [5, 36, 87] and also has favorable integrality and tightness properties. This model is described in detail in Section 4.3.3 of Chapter 4 and we denote it by MC. We also include two models that are based on independent branching schemes of linear depth. The first model is based on the independent branching scheme for SOS2 constraints on $(\lambda_j)_{j=0}^{n}$ given by $L_k = \{k, \dots, n\}$, $R_k = \{0, \dots, k\}$ for every $k \in \{1, \dots, n-1\}$. This formulation has been independently developed in [120] and is currently defined only for

functions of one variable. We denote this model by CC:Lin1. The second model is based on an independent branching defined in [96, p. 573]. This branching scheme is defined for any triangulation and its depth is equal to the number of vertices in the triangulation. In particular for piecewise linear functions of one variable with $k$ intervals or segments its depth is $k+1$ and for piecewise linear functions on a $k \times k$ grid it is $(k+1)^2$. We denote the model by CC:Lin2. We also tested some other piecewise linear models, but do not report results for them since they did not significantly improve the worst results reported here. We refer the reader to [130] for a more detailed study and evaluation of mixed integer formulations for piecewise linear functions. In addition to the mixed integer programming formulations we tested the traditional SOS2 formulation of univariate piecewise linear functions which does not include binary variables. We implemented this formulation using CPLEX's built in support for SOS2 constraints and we denote it by SOS2. All models were generated using Ilog Concert Technology and solved using CPLEX 11 on a dual 2.4GHz Xeon Linux workstation with 2GB of RAM. Furthermore, all tests were run with a time limit of 10000 seconds.

We note that Log, Inc, MC, CC:Lin1 and CC:Lin2 are mixed integer programming problems that do not include SOS2 constraints such as the ones supported by CPLEX. Hence, when CPLEX solves these formulations the only type of branching that occurs is due to the fixing of binary variables to zero or one. For Log, CC:Lin1 and CC:Lin2 this binary branching induces a specialized branching schemes that fixes some $\lambda$ variables to zero, but CPLEX does not directly fix $\lambda$ variables to zero. In contrast, formulation SOS2 does not contain any binary variables and to solve it CPLEX executes the traditional SOS2 branching of [12] by directly fixing $\lambda$ variables to zero.

The first set of experiments correspond to piecewise linear functions of one variable for which we used the transportation models from [131]. We selected the instances with 10 supply and 10 demand nodes and for each of the 5 available instances we generated several randomly generated objective functions. We generated a separable piecewise linear objective function given by the sum of concave non-decreasing piecewise linear functions of the flow in each arc. We use concave functions because they are widely used in practice

and because using them results in NP-hard problems [72] that are challenging for our experiments. For each instance and number of segments we generated 20 objective functions to obtain a total of 100 instances for each number of segments. We excluded LB2 as LB1 performed consistently better. Table 13 shows the minimum, average, maximum and standard deviation of the solve times in seconds for 4, 8, 16 and 32 segments. The tables also shows the number of times the solves failed because the time limit was reached and the number of times each formulation had the fastest solve time (win or tie). MC is the best model for 4 and 8 segments and Log is clearly the best model for 16 and 32 segments.

**Table 13:** Solve times for one variable functions [s].

(a) 4 segments.

| stat | Log | LB1 | MC | Inc | SOS2 |
|---|---|---|---|---|---|
| min | 0 | 0 | 0 | 0 | 0 |
| **avg** | **2** | **3** | **1** | **3** | **2** |
| max | 12 | 16 | 8 | 15 | 8 |
| std | 2 | 3 | 2 | 3 | 1 |
| wins | 25 | 1 | 46 | 2 | 27 |
| fail | 0 | 0 | 0 | 0 | 0 |

(b) 8 segments.

| stat | Log | LB1 | MC | Inc | SOS2 |
|---|---|---|---|---|---|
| min | 1 | 3 | 1 | 5 | 1 |
| **avg** | **12** | **26** | **10** | **47** | **16** |
| max | 84 | 116 | 39 | 160 | 202 |
| std | 11 | 17 | 7 | 31 | 23 |
| wins | 34 | 0 | 43 | 0 | 23 |
| fail | 0 | 0 | 0 | 0 | 0 |

(c) 16 segments.

| stat | Log | LB1 | MC | Inc | SOS2 |
|---|---|---|---|---|---|
| min | 0 | 7 | 2 | 23 | 2 |
| **avg** | **24** | **124** | **97** | **284** | **109** |
| max | 96 | 376 | 730 | 1250 | 1030 |
| std | 18 | 78 | 122 | 201 | 167 |
| wins | 95 | 0 | 3 | 0 | 2 |
| fail | 0 | 0 | 0 | 0 | 0 |

(d) 32 segments.

| stat | Log | LB1 | MC | Inc | SOS2 |
|---|---|---|---|---|---|
| min | 2 | 117 | 23 | 214 | 10 |
| **avg** | **43** | **569** | **2246** | **889** | **925** |
| max | 194 | 2665 | 10000 | 3943 | 10000 |
| std | 39 | 476 | 3208 | 662 | 1900 |
| wins | 98 | 0 | 0 | 0 | 2 |
| fail | 0 | 0 | 9 | 0 | 2 |

The next set of experiments correspond to piecewise linear functions of two variables for which we selected a series of two commodity transportation problems with 5 supply nodes and 2 demand nodes. These instances were constructed by combining two $5 \times 2$ transportation problems generated in a manner similar to the instances used in [131]. The supplies, demands and individual commodity arc capacities for each commodity were obtained from two different transportation problems and the joint arc capacities were set to 3/4 of the sum of the corresponding individual arc capacities. We considered an objective function of the form $\sum_{e \in E} f_e(x_e^1, x_e^2)$ where $E$ is the common set of 10 arcs of the transportation

problems and $f_e(x_e^1, x_e^2)$ is a piecewise linear function of the flows $x_e^i$ in arc $e$ of commodity $i$ for $i = 1, 2$. Each component $f_e(x_e^1, x_e^2)$ for arc $e$ with individual arc capacities $u_e^i$ for commodity $i = 1, 2$ was constructed as follows. We begin by triangulating $[0, u_e^1] \times [0, u_e^2]$ as described in Section 3.4 with a $K \times K$ segment grid. Using this triangulation we then obtained $f_e(x_e^1, x_e^2)$ by interpolating $g\left(\left\|\left(x_e^1, x_e^2\right)\right\|\right)$ where $\|\cdot\|$ is the euclidean norm and $g : \left[0, \left\|\left(u_e^1, u_e^2\right)\right\|\right] \to \mathbb{R}$ is a continuous concave piecewise linear function which was randomly generated independently for each arc in a similar way to the one variable functions of the previous set of experiments. The idea of this function is to use the sub-linearity of the euclidean norm to consider discounts for sending the two commodities in the same arc and concave function $g$ to consider economies of scale. We note that although $g$ is concave its interpolation is not always concave due to the known fact that multivariate interpolation on a predefined triangulation is not always shape preserving [30]. We selected 5 combinations of different pairs of the original transportation problems and for each one of these we generated 20 objective functions for a total of 100 instances for each $K$. For these instances we excluded SOS2 and LB1 as they are only defined for univariate functions. Table 14 shows the statistics for this set of instances. In the two variable case, Log is best for all sizes and the advantage becomes overwhelming for the largest instances.

**Table 14:** Solve times for two variable functions on a $4 \times 4$, $8 \times 8$ and $16 \times 16$ grids [s].

(a) $4 \times 4$ grid.

| stat | Log | LB2 | MC | Inc |
|---|---|---|---|---|
| min | 0 | 1 | 1 | 3 |
| **avg** | **3** | **6** | **6** | **32** |
| max | 9 | 22 | 17 | 127 |
| std | 2 | 4 | 3 | 26 |
| wins | 87 | 9 | 5 | 0 |
| fail | 0 | 0 | 0 | 0 |

(b) $8 \times 8$ grid.

| stat | Log | LB2 | MC | Inc |
|---|---|---|---|---|
| min | 2 | 37 | 31 | 100 |
| **avg** | **13** | **196** | **398** | **769** |
| max | 33 | 804 | 5328 | 6543 |
| std | 5 | 129 | 584 | 1111 |
| wins | 100 | 0 | 0 | 0 |
| fail | 0 | 0 | 0 | 31 |

(c) $16 \times 16$ grid.

| stat | Log | LB2 | MC | Inc |
|---|---|---|---|---|
| min | 27 | 3116 | 2853 | 772 |
| **avg** | **56** | **9825** | **9266** | **4857** |
| max | 118 | 10000 | 10000 | 10000 |
| std | 19 | 866 | 1678 | 3429 |
| wins | 100 | 0 | 0 | 0 |
| fail | 0 | 94 | 77 | 20 |

It is clear that one of the advantages of Log is that it is smaller than the other formulations while retaining favorable tightness properties. In addition, formulation Log effectively transforms CPLEX's binary variable branching into a specialized branching scheme for piecewise linear functions. This allows formulation Log to combine the favorable properties of specialized branching schemes and the technology in CPLEX's variable branching. Given its computational advantages, we anticipate that Log will become a valuable tool in practice. Results for aditional computational experiments including other formulations are included in Section 4.5 of Chapter 4.

## *3.7 Conclusions*

We have introduced a technique for modeling hard combinatorial problems with a mixed 0-1 integer programing formulation that uses a logarithmic number of binary variable and extra constraints. It is based on the concept of independent branching which is closely related to specialized branching schemes for combinatorial optimization. Using this technique we have introduced the first binary formulations for SOS1 and SOS2 constraints and for one and two variable piecewise linear functions that use a logarithmic number of binary variables and extra constraints. Finally, we have illustrated the usefulness of these new formulations by

showing that for one and two variable piecewise linear functions they provide a significant computational advantage.

There are still a number of unanswered questions concerning necessary and more general sufficient conditions for the existence of formulations with a logarithmic number of binary variables and extra constraints. For example, if we allow the formulation to have a number of binary variables and extra constraints whose asymptotic growth is logarithmic our sufficient conditions do not seem to be necessary. Consider cardinality constraints that restrict at most $K$ components of $\lambda \in [0,1]^n$ to be non-zero. We do not know of an independent branching scheme for this constraint, but it does have a formulation with a number of variables and constraints of logarithmic order. We can write cardinality constraints in the form (42) by letting $J = \{1, \ldots, n\}$, $I = \{1, \ldots, m\}$ for $m = \binom{n}{K}$ and $\{S_j\}_{j=1}^m$ be the family of all subsets of $J$ such that $|S_i| = K$. The traditional formulation for cardinality constraints is [40, 95]

$$\sum_{j=1}^{n} x_j \leq K; \quad \lambda_j \in [0,1], \quad \lambda_j \leq x_j, \quad x_j \in \{0,1\} \quad \forall j \in J. \tag{67}$$

Let $n$ be an even number. By choosing $K = n/2$, which is the non-trivial cardinality constraint with the largest number of sets $S_i$, we can use the fact that for $K = n/2$ we have $n \leq 2\log_2\left(\binom{n}{K}\right)$ to conclude that (67) has $O(\log_2(|I|))$ binary variables and extra constraints.

Another question concerns the case in which $I$ is not a power of two. Theoretical, this does not pose a problem because we can complete $I$ or adapt the independent branching scheme. However, preliminary tests in [130] showed that the computational effectiveness of independent branching schemes can be significantly reduced if $I$ is not a power of two. This is a common problem with binary encoded formulations, that can be mitigated by the use of techniques developed in [35].

# CHAPTER IV

# MIXED-INTEGER MODELS FOR NONSEPARABLE PIECEWISE LINEAR OPTIMIZATION: UNIFYING FRAMEWORK AND EXTENSIONS

## *4.1 Introduction*

We consider optimization problems involving piecewise linear functions modeled as Mixed Integer Programming (MIP) problems. When the functions considered are convex these problems can be modeled as Linear Programming (LP) problems, so we focus on non-convex functions for which the optimization problem is NP-hard even when all the functions are univariate [73].

Non-convex piecewise linear functions are generally used to approximate non-linearities arising from factors such as economies of scale or complex technological processes. They also naturally appear as cost functions of supply chain problems to model discounts for high volume and fixed charges. Applications of optimization problems with non-convex piecewise linear functions include production planning [51], optimization of electronic circuits [55], operation planning of gas networks [96], process engineering [18, 19], wetland restoration [122], merge-in-transit [37] and other network flow problems with non-convex piecewise linear objective functions [38].

Optimization problems involving non-convex piecewise linear functions can be solved with specialized algorithms [43, 73, 127] or they can be modeled as MIPs [87, 118, 36, 5, 72, 40, 140, 82, 68, 105, 132, 92, 95] and solved with a general purpose MIP solver. The advantage of this latter approach is that it capitalizes on the advanced technology available in state of the art MIP solvers [131]. MIP models for non-convex piecewise linear functions have been extensively studied, but existing comparisons [36, 72, 68] only concentrate on the case in which the functions are separable (i.e. can be written as the sum of univariate functions). When a non-separable function is known analytically it can sometimes be converted

into a separable one by algebraic manipulations [127]. However this conversion might be undesirable for numerical reasons [96] and because it can result in weaker formulations [38]. Furthermore, in many applications the functions come from complicated simulation models [78] and are not known analytically.

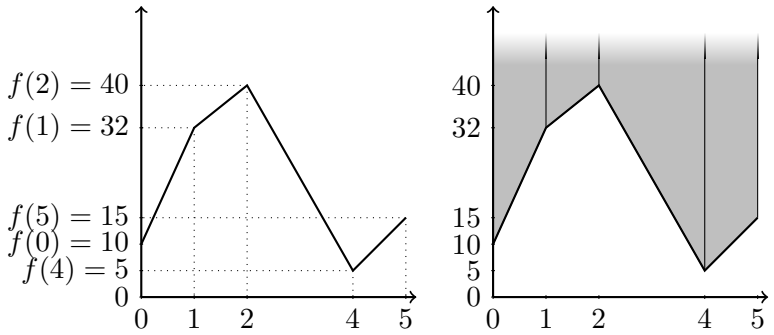
The main objective of this chapter is to unify the numerous MIP models for piecewise linear functions into a common framework which considers the possibility of non-separable functions and discontinuities directly. In addition, we present a theoretical and computational comparison of the models considered. Because models for separable multivariate functions can be obtained directly from models for univariate functions we will assume that multivariate functions are non-separable.

The remainder of chapter is organized as follows. In Section 4.2 we study the MIP modeling of continuous piecewise linear functions and define concepts that will be used throughout the paper. In Section 4.3 we give several MIP models for continuous piecewise linear functions and in Section 4.4 we study some properties of these formulations. In Section 4.5, we present computational results comparing the formulations for continuous functions. In Section 4.6, we study the extension of the formulations to lower semicontinuous functions and in Section 4.7 we present computational results comparing the formulations for this class of functions. In Section 4.8 we present some final remarks.

## *4.2 Modeling Piecewise Linear Functions*

An appropriate way of modeling a piecewise linear function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ is to model its epigraph given by $\text{epi}(f) := \{(x, z) \in D \times \mathbb{R} : f(x) \leq z\}$. For example, the epigraph of the function in Figure 12(a) is depicted in Figure 12(b).

For simplicity, we assume that the function domain $D$ is bounded and $f$ is only used in a constraint of the form $f(x) \leq 0$ or as an objective function that is being minimized. We then need a model of $\text{epi}(f)$ since $f(x) \leq 0$ can be modeled as $(x, z) \in \text{epi}(f)$, $z \leq 0$ and the minimization of $f$ can be achieved by minimizing $z$ subject to $(x, z) \in \text{epi}(f)$. For continuous functions we can also work with its graph, but modeling the epigraph will allow us to extend most of the results to some discontinuous functions and will simplify the




(a) $f$. (b) epi($f$).

**Figure 12:** A continuous piecewise linear function and its epigraph as the union of polyhedra.

analysis of formulation properties.

Following the theory developed by Jeroslow and Lowe [64, 66, 67, 68, 87], we say that a polyhedron $P \subset \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^p \times \mathbb{R}^q$ is a binary mixed-integer programming model for a set $S \subset \mathbb{R}^n \times \mathbb{R}$ if

$$(x, z) \in S \Leftrightarrow \exists(\lambda, y) \in \mathbb{R}^p \times \{0, 1\}^q \text{ s.t. } (x, z, \lambda, y) \in P. \tag{68}$$

Under the bounded domain assumption, Jeroslow and Lowe prove that the epigraph of a function can be modeled as a binary mixed-integer programming model if and only if it is a union of polyhedra with a common recession cone given by $C_n^+ := \{(0, z) \in \mathbb{R}^n \times \mathbb{R} : z \geq 0\}$. This condition is a special case of the results in [66], which also consider unbounded domains and more general uses of $f$ in a mathematical program. Furthermore, this condition implies that for a function $f : D \subset \mathbb{R} \to \mathbb{R}$ we have that epi($f$) can be modeled as a binary mixed-integer programming model if and only if $f$ is piecewise linear and lower semicontinuous. Our definition of a piecewise linear function is motivated by the extension of this characterization to the multivariate case.

A single variable continuous piecewise linear function $f : [0, u] \to \mathbb{R}$ can be described as

$$f(x) := \Big\{ m_i x + c_i \quad x \in [d_{i-1}, d_i] \quad \forall i \in \{1, \ldots, K\} \tag{69}$$

for some $K \in \mathbb{Z}_+$, $\{m_i\}_{i=1}^K \subset \mathbb{R}$, $\{c_i\}_{i=1}^K \subset \mathbb{R}$ and $\{d_k\}_{k=0}^K \subset \mathbb{R}$ such that $0 = d_0 < d_1 <$

$\ldots < d_K = u$. For example, function $f$ depicted Figure 12(a) can be described as

$$f(x) := \begin{cases} 22x + 10 & x \in [0, 1] \\ 8x + 24 & x \in [1, 2] \\ -17.5x + 75 & x \in [2, 4] \\ 10x - 35 & x \in [4, 5]. \end{cases} \tag{70}$$

A natural extension to the multivariate case is given by

**Definition 4.1** (Continuous Piecewise Linear Function)**.** *Let $D \subset \mathbb{R}^n$ be a compact set. A continuous function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ is a piecewise linear function if and only if there exists $\{m_P\}_{P\in\mathcal{P}} \subseteq \mathbb{R}^n$, $\{c_P\}_{P\in\mathcal{P}} \subseteq \mathbb{R}$ and a finite family of polytopes $\mathcal{P}$ such that $D = \bigcup_{P\in\mathcal{P}} P$ and*

$$f(x) := \begin{cases} m_P x + c_P & x \in P \quad \forall P \in \mathcal{P}. \end{cases} \tag{71}$$

Note that $D$ does not need to be convex or connected and that the boundedness assumption is for simplicity. Furthermore, if $x \in P_1 \cap P_2$ for two polytopes $P_1, P_2 \in \mathcal{P}$ the definition implies that $m_{P_1} x + c_{P_1} = m_{P_2} x + c_{P_2}$ which ensures the continuity of $f$ on $D$. In addition, Definition 4.1 does not specify how the polytopes are described as this is formulation dependent. In some formulations the polytopes are given as the convex hull of a finite number of points and in others the polytopes are given as a system of linear inequalities. The finite family of polytopes $\mathcal{P}$ is usually taken to be a triangulation of $D$ [82, 96, 140] and in fact some models will require this. For any family of polytopes $\mathcal{P}$ we denote the set of vertices of the family by $\mathcal{V}(\mathcal{P}) := \bigcup_{P\in\mathcal{P}} V(P)$ where $V(P)$ is the set of vertices of $P$. When $\mathcal{P}$ is a triangulation this coincides with the usual definition of vertices of a triangulation.

Using the approach of modeling epi($f$) as a union of polyhedra, Balas [6] and Jeroslow and Lowe introduce two standard ways of modeling $f$. An advantage of this approach is that it allows for a simple treatment of lower semicontinuous functions. In addition, with this definition the epigraph of a continuous piecewise linear function is the union of polyhedra given by

$$\text{epi}(f) = C_n^+ + \bigcup_{P\in\mathcal{P}} \text{conv}\left(\{(v, f(v))\}_{v\in V(P)}\right) \tag{72}$$

where conv denotes the convex hull operation and + denotes the Minkowski addition of sets. For the function defined in (70) and depicted in Figure 12(a) this characterization is given by

$$\text{epi}(f) = \{(0,r) \,:\, r \geq 0\} + \Big(\text{conv}\big(\{(0,10),(1,32)\}\big) \cup \text{conv}\big(\{(1,32),(2,40)\}\big) \\ \cup \text{conv}\big(\{(2,40),(4,5)\}\big) \cup \text{conv}\big(\{(4,5),(5,15)\}\big)\Big)$$

and illustrated in Figure 12(b). Note that this representation can be simplified by replacing $\text{conv}(\{(2,40),(4,5)\}) \cup \text{conv}(\{(4,5),(5,15)\})$ with $\text{conv}(\{(2,40),(4,5),(5,15)\})$, but this requires detecting that $(\text{conv}(\{(2,40),(4,5)\}) + \{(0,r) \,:\, r \geq 0\}) \cup (\text{conv}(\{(4,5),(5,15)\}) + \{(0,r) \,:\, r \geq 0\})$ is in fact a polyhedron.

## *4.3 Mixed Integer Programming Models for Piecewise Linear Functions*

In this section we review several new and existing formulations for continuous functions. We illustrate the formulations for the function defined in (70) and depicted in Figure 12(a).

### 4.3.1 Disaggregated convex combination models

All formulations in this section represent $(x,z) \in \text{epi}(f)$ as the convex combination of points $(v,f(v))$ for $v \in \mathcal{V}(\mathcal{P})$ plus a ray in cone $C_n^+$. They have one continuous variable for each $v \in V(P)$ and for each $P \in \mathcal{P}$ to represent a point $(x,z) \in \text{epi}(f)$ as $(x,z) = r + \sum_{P\in\mathcal{P}} \sum_{v\in V(P)} \lambda_{P,v}(v,f(v))$, for $r \in C_n^+$ and $\{\lambda_{P,v}\}_{P\in\mathcal{P},\, v\in V(P)} \subset \mathbb{R}_+$ such that $\sum_{P\in\mathcal{P}} \sum_{v\in V(P)} \lambda_{P,v} = 1$.

#### *4.3.1.1 Basic Model*

This formulation has no requirement on the family of polytopes and is given by

$$\sum_{P\in\mathcal{P}} \sum_{v\in V(P)} \lambda_{P,v} v = x, \qquad \sum_{P\in\mathcal{P}} \sum_{v\in V(P)} \lambda_{P,v} \left(m_P v + c_P\right) \leq z \tag{73a}$$

$$\lambda_{P,v} \geq 0 \quad \forall P \in \mathcal{P},\, v \in V(P), \qquad \sum_{v\in V(P)} \lambda_{P,v} = y_P \quad \forall P \in \mathcal{P} \tag{73b}$$

$$\sum_{P\in\mathcal{P}} y_P = 1, \qquad y_P \in \{0,1\} \quad \forall P \in \mathcal{P}. \tag{73c}$$

This formulation has been studied in [36, 64, 67, 87, 100, 103] and [118] and is sometimes referred to as the *convex combination model*. To distinguish it from the formulation in Section 4.3.2 we instead refer to it as the *disaggregated convex combination model* and denote it by DCC. For the function defined in (70) it is given by

$$\begin{aligned}
0\lambda_{[0,1],0} + 1\left(\lambda_{[0,1],1} + \lambda_{[1,2],1}\right) + 2\left(\lambda_{[1,2],2} + \lambda_{[2,4],2}\right) + 4\left(\lambda_{[2,4],4} + \lambda_{[4,5],4}\right) + 5\lambda_{[4,5],5} &= x \\
10\lambda_{[0,1],0} + 32\left(\lambda_{[0,1],1} + \lambda_{[1,2],1}\right) + 40\left(\lambda_{[1,2],2} + \lambda_{[2,4],2}\right) + 5\left(\lambda_{[2,4],4} + \lambda_{[4,5],4}\right) + 15\lambda_{[4,5],5} &\leq z \\
\lambda_{[0,1],0},\, \lambda_{[0,1],1},\, \lambda_{[1,2],1},\, \lambda_{[1,2],2},\, \lambda_{[2,4],2},\, \lambda_{[2,4],4},\, \lambda_{[4,5],4},\, \lambda_{[4,5],5} &\geq 0 \\
\lambda_{[0,1],0} + \lambda_{[0,1],1} = y_{[0,1]}, \quad \lambda_{[1,2],1} + \lambda_{[1,2],2} &= y_{[1,2]}, \\
\lambda_{[2,4],2} + \lambda_{[2,4],4} = y_{[2,4]}, \quad \lambda_{[4,5],4} + \lambda_{[4,5],5} &= y_{[4,5]} \\
y_{[0,1]} + y_{[1,2]} + y_{[2,4]} + y_{[4,5]} = 1, \quad y_{[0,1]},\, y_{[1,2]},\, y_{[2,4]},\, y_{[4,5]} &\in \{0,1\}.
\end{aligned}$$

#### *4.3.1.2 Logarithmic Model*

Using ideas from [61, 132] and [133] we can reduce the number of binary variables and constraints of DCC. To do this we identify each polytope in $\mathcal{P}$ with a binary vector in $\{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}$ through an injective function $B : \mathcal{P} \to \{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}$. We then use $\lceil \log_2 |\mathcal{P}| \rceil$ binary variables $y \in \{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}$ to force $\sum_{v \in V(P)} \lambda_{P,v} = 1$ when $y = B(\mathcal{P})$.

The resulting formulation has no requirement on the family of polytopes and is given by

$$\sum_{P \in \mathcal{P}} \sum_{v \in V(P)} \lambda_{P,v} v = x, \quad \sum_{P \in \mathcal{P}} \sum_{v \in V(P)} \lambda_{P,v} \left(m_P v + c_P\right) \leq z \tag{74a}$$

$$\lambda_{P,v} \geq 0 \quad \forall P \in \mathcal{P},\, v \in V(P), \quad \sum_{P \in \mathcal{P}} \sum_{v \in V(P)} \lambda_{P,v} = 1 \tag{74b}$$

$$\sum_{P \in \mathcal{P}^+(B,l)} \sum_{v \in V(P)} \lambda_{P,v} \leq y_l, \quad \sum_{P \in \mathcal{P}^0(B,l)} \sum_{v \in V(P)} \lambda_{P,v} \leq (1 - y_l), \tag{74c}$$

$$y_l \in \{0,1\} \quad \forall l \in L(\mathcal{P}), \tag{74d}$$

where $B : \mathcal{P} \to \{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}$ is any injective function, $\mathcal{P}^+(B,l) := \{P \in \mathcal{P} : B(P)_l = 1\}$, $\mathcal{P}^0(B,l) := \{P \in \mathcal{P} : B(P)_l = 0\}$ and $L(\mathcal{P}) := \{1, \ldots, \lceil \log_2 |\mathcal{P}| \rceil\}$. We refer to it as the *logarithmic dissagregated convex combination model* and denote it by DLog. For

the function defined in (70) and for $B([0,1]) = (0,0)^T$, $B([1,2]) = (0,1)^T$, $B([2,4]) = (1,1)^T, B([4,5]) = (1,0)^T$ it is given by

$$0\lambda_{[0,1],0} + 1\left(\lambda_{[0,1],1} + \lambda_{[1,2],1}\right) + 2\left(\lambda_{[1,2],2} + \lambda_{[2,4],2}\right) + 4\left(\lambda_{[2,4],4} + \lambda_{[4,5],4}\right) + 5\lambda_{[4,5],5} = x$$

$$10\lambda_{[0,1],0} + 32\left(\lambda_{[0,1],1} + \lambda_{[1,2],1}\right) + 40\left(\lambda_{[1,2],2} + \lambda_{[2,4],2}\right) + 5\left(\lambda_{[2,4],4} + \lambda_{[4,5],4}\right) + 15\lambda_{[4,5],5} \leq z$$

$$\lambda_{[0,1],0},\ \lambda_{[0,1],1},\ \lambda_{[1,2],1},\ \lambda_{[1,2],2},\ \lambda_{[2,4],2},\ \lambda_{[2,4],4},\ \lambda_{[4,5],4},\ \lambda_{[4,5],5} \geq 0$$

$$\lambda_{[2,4],2} + \lambda_{[2,4],4} + \lambda_{[4,5],4} + \lambda_{[4,5],5} \leq y_1, \quad \lambda_{[0,1],0} + \lambda_{[0,1],1} + \lambda_{[1,2],1} + \lambda_{[1,2],2} \leq (1 - y_1)$$

$$\lambda_{[1,2],1} + \lambda_{[1,2],2} + \lambda_{[2,4],2} + \lambda_{[2,4],4} \leq y_2, \quad \lambda_{[0,1],0} + \lambda_{[0,1],1} + \lambda_{[4,5],4} + \lambda_{[4,5],5} \leq (1 - y_2)$$

$$y_1,\ y_2 \in \{0,1\}.$$

### 4.3.2 Convex combination models

The formulations in this section reduce the number of continuous variables of DCC by aggregating variables associated with a point in $\mathcal{V}(\mathcal{P})$ that belongs to more than one polytope in $\mathcal{P}$. The resulting formulations have one continuous variable for each $v \in \mathcal{V}(\mathcal{P})$ and hence represent point $(x,z) \in \text{epi}(f)$ as $(x,z) = r + \sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v(v, f(v))$, for $r \in C_n^+$ and $\lambda \in \mathbb{R}_+^{\mathcal{V}(\mathcal{P})}$ such that $\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v = 1$.

#### *4.3.2.1 Basic Model*

This formulation has no requirement on the family of polytopes and is given by

$$\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v v = x, \qquad \sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v \left(m_P v + c_P\right) \leq z \quad (75a)$$

$$\lambda_v \geq 0 \quad \forall v \in \mathcal{V}(\mathcal{P}), \qquad \sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v = 1 \quad (75b)$$

$$\lambda_v \leq \sum_{P\in\mathcal{P}(v)} y_P \quad \forall v \in \mathcal{V}(\mathcal{P}), \qquad \sum_{P\in\mathcal{P}} y_P = 1, \qquad y_P \in \{0,1\} \quad \forall P \in \mathcal{P}, \quad (75c)$$

where $\mathcal{P}(v) := \{P \in \mathcal{P} : v \in P\}$. This formulation is studied in [41, 40, 52, 68, 72, 82, 87, 103, 104, 105] and [140] and is sometimes referred to as the *lambda method*. We refer to this formulation as the *convex combination model* and denote it by CC. For the function

defined in (70) it is given by

$$0\lambda_0 + 1\lambda_1 + 2\lambda_2 + 4\lambda_4 + 5\lambda_5 = x, \quad 10\lambda_0 + 32\lambda_1 + 40\lambda_2 + 5\lambda_4 + 15\lambda_5 \leq z$$

$$\lambda_0,\, \lambda_1,\, \lambda_2,\, \lambda_4,\, \lambda_5 \geq 0, \quad \lambda_0 + \lambda_1 + \lambda_2 + \lambda_4 + \lambda_5 = 1$$

$$\lambda_0 \leq y_{[0,1]}, \quad \lambda_1 \leq y_{[0,1]} + y_{[1,2]}, \quad \lambda_2 \leq y_{[1,2]} + y_{[2,4]}, \quad \lambda_4 \leq y_{[2,4]} + y_{[4,5]}, \quad \lambda_5 \leq y_{[4,5]}$$

$$y_{[0,1]} + y_{[1,2]} + y_{[2,4]} + y_{[4,5]} = 1, \quad y_{[0,1]},\, y_{[1,2]},\, y_{[2,4]},\, y_{[4,5]} \in \{0,1\}.$$

#### *4.3.2.2 Logarithmic Model*

As in DLog's construction we can reduce the number of binary variables and constraints of CC by identifying each polytope in $\mathcal{P}$ with a binary vector in $\{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}$ through an injective function $B : \mathcal{P} \to \{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}$. However, we now need $B$ to comply with conditions that can be interpreted as the construction of a binary branching scheme for the effect of (75c) on $\lambda \in \mathbb{R}^{\mathcal{V}(\mathcal{P})}$. This constraint requires the non-zero $\lambda$ variables to be associated with the vertices of a polytope in $\mathcal{P}$:

$$\exists P \in \mathcal{P} \text{ s.t. } \{v \in \mathcal{V}(\mathcal{P}) \,:\, \lambda_v > 0\} \subset V(P). \tag{76}$$

A binary branching scheme for (76) imposes it by fixing to zero disjoint sets of $\lambda$ variables in each side of a series of branching dichotomies. For example, for the function depicted in Figure 12(a) we have $\mathcal{P} = \{[0,1],[1,2],[2,4],[4,5]\}$ and we can force (76) by the branching scheme given by the following two dichotomies: ($\lambda_2 = 0$ or $\lambda_0 = \lambda_5 = 0$) and ($\lambda_4 = \lambda_5 = 0$ or $\lambda_0 = \lambda_1 = 0$).

In general, a branching scheme for (76) is a family of dichotomies $\{L_s, R_s\}_{s \in S}$ indexed by a finite set $S$ and with $L_s, R_s \subset \mathcal{V}(\mathcal{P})$ such that for every $P \in \mathcal{P}$ we have $V(P) = \bigcap_{s \in S} \big(\mathcal{V}(\mathcal{P}) \setminus T_s\big)$, where $T_s = L_s$ or $T_s = R_s$ for each $s \in S$. For such a branching scheme a valid formulation is given by

$$\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v v = x, \qquad \sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v \left(m_P v + c_P\right) \leq z \tag{77a}$$

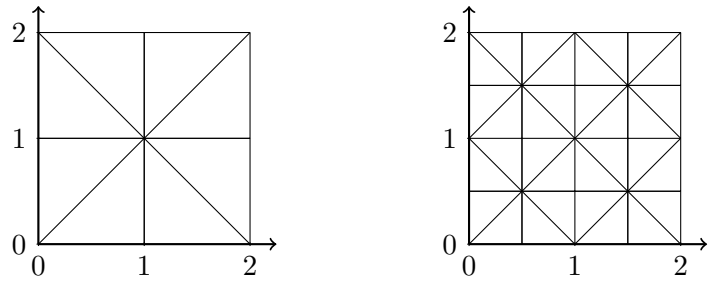
$$\lambda_v \geq 0 \quad \forall v \in \mathcal{V}(\mathcal{P}), \qquad \sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v = 1 \tag{77b}$$

$$\sum_{v \in L_s} \lambda_v \leq y_s, \qquad \sum_{v \in R_s} \lambda_v \leq (1 - y_s), \qquad y_s \in \{0,1\} \quad \forall s \in \mathcal{S}. \tag{77c}$$

For (77) to have a logarithmic number of binary variables, we need a branching scheme with a logarithmic number of dichotomies. Such a scheme was introduced in [132] and [133] for the case when the family of polytopes $\mathcal{P}$ is topologically equivalent or compatible [2] with a triangulation known as $J_1$ or "Union Jack" [125]. For simplicity we first describe the formulation for the case when $\mathcal{P} = J_1$ and then show how to extend the formulation to the case where $\mathcal{P}$ is compatible with $J_1$.

$J_1$ is defined for $D = [0, K]^n$ for $K \in \mathbb{Z}$ even. The vertex set of $J_1$ is given by $\mathcal{V} = \{0, \dots, K\}^n$. The simplices of $J_1$ are constructed as follows. Let $N = \{1, \dots, n\}$, $\mathcal{V}^0 = \{v \in \mathcal{V} : v_i \text{ is odd}, \forall i \in N\}$, $\text{Sym}(N)$ be the group of all permutations on $N$ and $e^i$ be the $i$-th unit vector of $\mathbb{R}^n$. For each $(v^0, \pi, s) \in \mathcal{V}^0 \times \text{Sym}(N) \times \{-1, 1\}^n$ define $j_1(v^0, \pi, s)$ to be the simplex whose vertices are $\{y^i\}_{i=0}^n$ where $y^0 = v^0$ and $y^i = y^{i-1} + s_{\pi(i)} e^{\pi(i)}$ for each $i \in N$. Triangulation $J_1$ of $D$ is given by all these simplices, which is illustrated in Figure 13(a) for $D = [0, 2]^2$. A branching scheme for $J_1$ is constructed by dividing index set $S$ into two sets $S_1$ and $S_2$. The first set is given by $S_1 := N \times \{1, \dots, \lceil \log_2(K) \rceil\}$ and $L_{(s_1,s_2)} := \{v \in \mathcal{V} : v_{s_1} \in O(s_2, 1)\}$, $R_{(s_1,s_2)} := \{v \in \mathcal{V} : v_{s_1} \in O(s_2, 0)\}$ for each $(s_1, s_2) \in S_1$, where $O(l, b) := \left\{k \in \{0, \dots, K\} : \left(k = 0 \text{ or } G_l^k = b\right) \text{ and } \left(k = K \text{ or } G_l^{k+1} = b\right)\right\}$ for an arbitrary but fixed set of binary vectors $(G^l)_{l=1}^K \subset \{0, 1\}^{\lceil \log_2(K) \rceil}$ such that $G^l$ and $G^{l+1}$ differ in at most one component for each $l \in \{1, \dots, \lceil \log_2(K) \rceil - 1\}$. There are many different sets of vectors with this property and they are usually referred to as *reflective binary* or *Gray codes* [138]. The second set is given by $S_2 := \{(s_1, s_2) \in N^2 : s_1 < s_2\}$ and $L_{(s_1,s_2)} := \{v \in \mathcal{V} : v_{s_1} \text{ is even and } v_{s_2} \text{ is odd}\}$, $R_{(s_1,s_2)} := \{v \in \mathcal{V} : v_{s_1} \text{ is odd and } v_{s_2} \text{ is even}\}$ for each $(s_1, s_2) \in S_2$.

Following [132] and [133] we refer to the formulation obtained with this scheme as the *logarithmic branching convex combination model* and denote it by Log. As mentioned before, Log can be extended to any family of polytopes $\mathcal{P}$ that is compatible with the $J_1$ triangulation. This requires the existence of a bijection $\varphi : \{0, \dots, K\}^n \to \mathcal{V}(\mathcal{P})$ between the vertices of $J_1$ and the family $\mathcal{P}$ such that $v_1, \dots, v_{n+1}$ are the vertices of a simplex in $J_1$ if and only if $\varphi(v_1), \dots, \varphi(v_{n+1})$ are the vertices of a polytope in $\mathcal{P}$. For example, taking




(a) $J_1$ triangulation of $[0,2]^2$. (b) 1/2 scaled $J_1$ triangulation of $[0,2]^2$.

**Figure 13:** Examples of triangulations of subsets of $\mathbb{R}^2$.

$\varphi : \{0,\ldots,4\}^2 \to \{0,1/2,1,3/2,2\}^2$ given by $\varphi(v_1,v_2) = (v_1/2, v_2/2)$ we have that the 1/2 scaled $J_1$ triangulation depicted in Figure 13(b) is compatible with the $J_1$ triangulation of $[0,4]^2$. Using bijection $\varphi$ the formulation for $\mathcal{P}$ is simply obtained by replacing (77a) by $\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v \varphi(v) = x$ and $\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v \left(m_P \varphi(v) + c_P\right) \leq z$. For the function defined in (70), for $G^1 = (0,0)^T$, $G^2 = (1,0)^T$, $G^3 = (1,1)^T$, $G^4 = (0,1)^T$ and $\varphi(0) = 0$, $\varphi(1) = 1$, $\varphi(2) = 2$, $\varphi(3) = 4$, $\varphi(4) = 5$, Log is given by

$$0\lambda_0 + 1\lambda_1 + 2\lambda_2 + 4\lambda_3 + 5\lambda_4 = x, \quad 10\lambda_0 + 32\lambda_1 + 40\lambda_2 + 5\lambda_3 + 15\lambda_4 \leq z$$

$$\lambda_0,\, \lambda_1,\, \lambda_2,\, \lambda_3,\, \lambda_4 \geq 0, \quad \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$$

$$\lambda_2 \leq y_1, \quad \lambda_0 + \lambda_4 \leq (1 - y_1), \quad \lambda_3 + \lambda_4 \leq y_2, \quad \lambda_0 + \lambda_1 \leq (1 - y_2), \quad y_1,\, y_2 \in \{0,1\}.$$

A similar formulation can be obtained from a branching scheme introduced in [96], but the resulting formulation has a linear instead of logarithmic number of binary variables.

### 4.3.3 Multiple choice model

This formulation has no requirement on the family of polytopes and is given by

$$\sum_{P\in\mathcal{P}} x^P = x, \qquad \sum_{P\in\mathcal{P}} \left(m_P x^P + c_P y_P\right) \leq z \tag{78a}$$

$$A_P x^P \leq y_P b_P \quad \forall P \in \mathcal{P} \tag{78b}$$

$$\sum_{P\in\mathcal{P}} y_P = 1, \qquad y_P \in \{0,1\} \quad \forall P \in \mathcal{P}, \tag{78c}$$

where $A_P x \leq b_P$ is the set of linear inequalities describing $P$. This formulation has been studied in [5, 36, 67, 87] and [103]. We refer to this formulation as the *multiple choice*

*model* and denote it by MC. For the function defined in (70) it is given by

$$x^{[0,1]} + x^{[1,2]} + x^{[2,4]} + x^{[4,5]} = x$$

$$(22x^{[0,1]} + 10y_{[0,1]}) + (8x^{[1,2]} + 24y_{[1,2]}) + (-17.5x^{[2,4]} + 75y_{[2,4]}) + (10x^{[4,5]} - 35y_{[4,5]}) \leq z$$

$$0y_{[0,1]} \leq x^{[0,1]} \leq y_{[0,1]}, \quad 1y_{[1,2]} \leq x^{[1,2]} \leq 2y_{[1,2]},$$

$$2y_{[2,4]} \leq x^{[2,4]} \leq 4y_{[2,4]}, \quad 4y_{[4,5]} \leq x^{[4,5]} \leq 5y_{[4,5]}$$

$$y_{[0,1]} + y_{[1,2]} + y_{[2,4]} + y_{[4,5]} = 1, \quad y_{[0,1]},\, y_{[1,2]},\, y_{[2,4]},\, y_{[4,5]} \in \{0,1\}$$

### 4.3.4 Incremental model

This formulation requires $\mathcal{P}$ to be a triangulation with a special ordering property. This property always holds for univariate functions so for simplicity we describe the formulation for this case first. For univariate function $f : [l, u] \to \mathbb{R}$ and for $\mathcal{P} = \{[d_{k-1}, d_k]\}_{k=1}^{K}$ where $l = d_0 \leq d_1 \leq \ldots \leq d_K = u$, the formulation is given by

$$d_0 + \sum_{k=1}^{K} \delta_k \left(d_k - d_{k-1}\right) = x, \qquad f(d_0) + \sum_{k=1}^{K} \delta_k \left(f(d_k) - f(d_{k-1})\right) \leq z \tag{79a}$$

$$\delta_1 \leq 1, \quad \delta_K \geq 0, \quad \delta_{k+1} \leq y_k \leq \delta_k, \quad y_k \in \{0,1\} \quad \forall k \in \{1, \ldots, K-1\}. \tag{79b}$$

The extension to multivariate functions [140] requires the family of polytopes to be a triangulation $\mathcal{T}$ that complies with the following ordering properties:

O1. The simplices in $\mathcal{T}$ can be ordered as $T_1, \ldots, T_{|\mathcal{T}|}$ so that $T_i \cap T_{i-1} \neq \emptyset$ for each $i \in \{2, \ldots, |\mathcal{T}|\}$.

O2. For the order above, the vertices of each simplex $T_i$ can be ordered as $v_i^0, \ldots, v_i^{|V(T_i)|-1}$ in a way such that $v_{i-1}^{|V(T_i)|-1} = v_i^0$ for $i \in \{2, \ldots, |\mathcal{T}|\}$.

These properties are required to represent $(x, z)$ *incrementally* akin to (79a) for the univariate case. Fortunately these conditions are met for many triangulations including $J_1$ [140].

For a given order complying with O1–O2 the formulation is given by

$$v_0^0 + \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^{|V(T_i)|-1} \delta_i^j \left(v_i^j - v_i^0\right) = x, \quad f(v_0^0) + \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^{|V(T_i)|-1} \delta_i^j \left(f(v_i^j) - f(v_i^0)\right) \leq z \tag{80a}$$

$$\sum_{j=1}^{|V(T_1)|-1} \delta_1^j \leq 1, \quad \delta_i^j \geq 0 \quad \forall i \in \{1, \ldots, |\mathcal{T}|\},\, j \in \{1, \ldots, |V(T_i)| - 1\} \tag{80b}$$

$$y_i \leq \delta_i^{|V(T_i)|-1}, \quad \sum_{j=1}^{|V(T_{i+1})|-1} \delta_{i+1}^j \leq y_i, \quad y_i \in \{0,1\} \quad \forall i \in \{1, \ldots, |\mathcal{T}| - 1\}. \tag{80c}$$

This formulation has been studied in [36, 41, 40, 72, 95, 105, 112, 118, 128] and [140] and it is sometimes referred to as the *delta method*. Following [36] and [72] we refer to it as the *incremental model* and denote it by Inc. For the function defined in (70) it is given by

$$10 + 22\delta_1 + 8\delta_2 - 35\delta_3 + 10\delta_4 \leq z, \quad 0 + \delta_1 + \delta_2 + 2\delta_3 + \delta_4 = x$$

$$y_1 \leq \delta_1 \leq 1, \quad y_2 \leq \delta_2 \leq y_1, \quad y_3 \leq \delta_3 \leq y_2, \quad 0 \leq \delta_4 \leq y_3, \quad y_1, y_2, y_3 \in \{0,1\}.$$

### *4.4 Properties of Mixed Integer Programming Formulations*

In this section we study some properties of the formulations. We begin by studying the strength of the formulations as a model of epi($f$) ignoring possible interactions with other constraints. For this case a motivating problem is the minimization of $f : D \subset \mathbb{R}^n \to \mathbb{R}$ over its domain $D$ given by

$$\min_{x \in D} f(x) = \min_{(x,z) \in \text{epi}(f)} z. \tag{81}$$

We then study the effects of interactions with other constraints using as a motivating problem

$$\min_{x \in X} f(x) = \min_{(x,z) \in \text{epi}(f) \cap (X \times \mathbb{R})} z, \tag{82}$$

where $X \subset D$ is any compact set. Finally, we study the sizes of the formulations and their requirements on the family of polytopes $\mathcal{P}$ used to describe the piecewise linear function.

Consider a MIP formulation of epi($f$) given by a polytope $P \subset \mathbb{R}^{n+p+q+1}$ complying with (68). The linear programming (LP) relaxation of the formulation is then simply $P$. Alternative MIP formulations are usually compared with respect to the tightness of their LP relaxation in the absence of additional constraints. In this regard, the strongest possible

property of a MIP formulation is to require that all vertices of its LP relaxation comply with the corresponding integrality requirements. Formulations with this property are referred to as *locally ideal* in [105] and [106]. It is shown in [82, 105] and [140] that CC is not locally ideal. However all of the other formulations from Section 4.3 are locally ideal.

**Theorem 4.2.** *All formulations from Section 4.3 except CC are locally ideal.*

*Proof.* All models except CC, DLog and Log have been previously shown to be locally ideal [6, 67, 87, 105, 118, 140], so we only need to prove that DLog and Log are locally ideal.

For Log assume for contradiction that there exists an vertex $(x, z, \lambda, y)$ of (77) such that $y_s \in (0, 1)$ for some $s \in S$. We divide the proof in two main cases.

**Case 1:** $\sum_{v \in L_s} \lambda_v < y_s$ **and** $\sum_{v \in R_s} \lambda_v < (1 - y_s)$**.** For $\varepsilon > 0$ define $(x^1, z^1, \lambda^1, y^1)$ and $(x^2, z^2, \lambda^2, y^2)$ as $x^1 = x^2 = x$, $z^1 = z^2 = z$, $\lambda^1 = \lambda^2 = \lambda$, $y^1 = y + \varepsilon$ and $y^2 = y - \varepsilon$. For sufficiently small $\varepsilon$ we have that $(x^1, z^1, \lambda^1, y^1)$ and $(x^2, z^2, \lambda^2, y^2)$ comply with (77) and $(x, z, \lambda, y) = 1/2(x^1, z^1, \lambda^1, y^1) + 1/2(x^2, z^2, \lambda^2, y^2)$. This contradicts $(x, z, \lambda, y)$ being a vertex.

**Case 2:** $\sum_{v \in L_s} \lambda_v = y_s$ **or** $\sum_{v \in R_s} \lambda_v = (1 - y_s)$**.** Without loss of generality we may assume that $\sum_{v \in L_s} \lambda_v = y_s$. We then have $v_s \in L_s$ such that $0 < \lambda_{v_s} < 1$ and $v_l \notin L_s$ such that $0 < \lambda_{v_l} < 1$. If $\sum_{v \in R_s} \lambda_v = (1 - y_s)$ we additionally select $v_l \in R_s$. For $\varepsilon > 0$ we define $(x^1, z^1, \lambda^1, y^1)$ and $(x^2, z^2, \lambda^2, y^2)$ in the following way. First let $\lambda^1_k = \lambda^1_k = \lambda_k$ for all $k \notin \{v_s, v_l\}$, $\lambda^1_{v_s} = \lambda_{v_s} + \varepsilon$, $y^1_s = y_s + \varepsilon$, $\lambda^2_{v_s} = \lambda_{v_s} - \varepsilon$, $y^2_s = y_s - \varepsilon$, $\lambda^1_{v_l} = \lambda^1_{v_l} - \varepsilon$ and $\lambda^2_{v_l} = \lambda^2_{v_l} + \varepsilon$. To define $y^1_t$ and $y^2_t$ for each $t \in S \setminus \{s\}$ we only need to consider the following four cases (note that $L_t \cap R_t = \emptyset$ and that without loss of generality we can exchange $R_t$ and $L_t$):

(a) $v_s, v_l \in L_t$ and $v_s, v_l \notin R_t$.

(b) $v_s \in L_t$ and $v_l \in R_t$.

(c) $v_s \in L_t$, $v_l \notin L_t$ and $v_l \notin R_t$ (case $v_l \in L_t$, $v_s \notin L_t$ and $v_s \notin R_t$ is analogous).

(d) $v_s$, $v_l \notin L_t$ and $v_s, v_l \notin R_t$.

For case a) we can simply set $y_t^1 = y_t^2 = y$. For case b) we have $0 < y_t < 1$ and we can set $y_t^1 = y_t + \varepsilon$ and $y_t^2 = y_t - \varepsilon$. For case c) we either have $\sum_{v \in L_t} \lambda_v < y_t$ or $\sum_{v \in L_t} \lambda_v = y_t$. For the first case we can simply set $y_t^1 = y_t^2 = y$. For the second case we have $0 < y_t < 1$ and $\sum_{v \in R_t} \lambda_v < (1 - y_t)$ and we can set $y_t^1 = y_t + \varepsilon$ and $y_t^2 = y_t - \varepsilon$. For case d) we can set $y_t^1 = y_t^2 = y$. Finally we set $x^1 = x + \varepsilon(v_s - v_l)$, $x^2 = x - \varepsilon(v_s - v_l)$, $z^1 = z + \varepsilon(f(v_s) - f(v_l))$ and $z^2 = z - \varepsilon(f(v_s) - f(v_l))$. We again have that for sufficiently small $\varepsilon$ $(x^1, z^1, \lambda^1, y^1)$ and $(x^2, z^2, \lambda^2, y^2)$ comply with (77) and $(x, z, \lambda, y) = 1/2(x^1, z^1, \lambda^1, y^1) + 1/2(x^2, z^2, \lambda^2, y^2)$.

For DLog the proof is analogous. □

For a locally ideal formulation $P$ of $\text{epi}(f)$ we have

$$\min_{(x,z,\lambda,y) \in P} z = \min_{x \in D} f(x), \tag{83}$$

which allows solving (81) directly as an LP and can be useful for solving (82) with a branch-and-bound algorithm. However, as noted in [36] and [72], property (83) might still hold for non-locally ideal formulations such as CC. In fact, we will see that (83) is implied by a geometric property introduced by Jeroslow and Lowe, but is weaker than the locally ideal property.

A slightly restricted version of Proposition 3.1 in [67] states that for any closed set $S \subset \mathbb{R}^n \times \mathbb{R}$ and for any binary mixed-integer programming model $P \subset \mathbb{R}^{n+p+q+1}$ for $S$, the projection of $P$ onto the first $n + 1$ variables contains the convex hull of $S$. Jeroslow and Lowe referred to a model $P$ of $S$ as *sharp* when the projection is exactly the convex hull of $S$. By letting $S$ be the epigraph of piecewise linear function $f$ we directly get the following result.

**Theorem 4.3.** *[36, 67, 87] Let $D \subset \mathbb{R}^n$ be a polytope, $f : D \to \mathbb{R}$ be a continuous piecewise linear function, $P \subset \mathbb{R}^{n+p+q+1}$ be a MIP formulation for* $\text{epi}(f)$ *satisfying* (68) *and $P_{(x,z)}$ the projection of $P$ onto $(x, z)$. Then* $\text{epi}(\text{convenv}_D(f)) = \text{conv}(\text{epi}(f)) \subset P_{(x,y)}$ *where* $\text{convenv}_D$ *is the lower convex envelope of $f$ over $D$.*

A formulation $P$ of $\text{epi}(f)$ is said to be sharp when $\text{epi}(\text{convenv}_D(f)) = \text{conv}(\text{epi}(f)) = P_{(x,y)}$. Because $\min_{x \in D} f(x) = \min_{x \in D} \text{convenv}_D(f)(x)$ we have that (83) holds for sharp

formulations. Sharpness has been shown to hold for some formulations in [36, 64, 66, 67, 68, 72, 87, 105] and [118] and the following proposition states that it holds for any locally ideal formulation.

**Proposition 4.4.** *Any locally ideal formulation is sharp.*

*Proof.* We need to prove $P_{(x,y)} \subset \text{conv}(\text{epi}(f))$. If $x \in P_{(x,y)}$ then because $P$ is locally ideal there exist $\lambda \in \mathbb{R}^p$, $y \in [0,1]^q$ such that $(x, z, \lambda, y) = (0, h, 0, 0) + \sum_{i \in I} \mu_i (x^i, z^i, \lambda^i, y^i)$ for $h \geq 0$, $|I| < \infty$, $\mu \in \mathbb{R}^I_+$, $\sum_{i \in I} \mu_i = 1$, and $(x^i, z^i, \lambda^i, y^i) \in P$ with $y^i \in \{0,1\}^q$ for every $i \in I$. Then by (68) $(x^i, z^i) \in \text{epi}(f)$ for all $i \in I$ and hence $(x, z) \in \text{conv}(\text{epi}(f))$. □

We then directly have that all formulations except CC are sharp. As noted in Section 4.3.2, CC can be obtained from DCC in a way which reduces its tightness. Fortunately, this loss of tightness does not affect the sharpness properties of CC so the following theorem holds.

**Theorem 4.5.** *All formulations from Section 4.3 are sharp.*

*Proof.* This is direct from Theorem 4.2 for all formulations except CC. For CC the result follows by noting that the projection onto the $x$ and $z$ variables of the polyhedron given by $\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v v = x$, $\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v f(v) \leq z$, $\lambda_v \geq 0 \quad \forall v \in \mathcal{V}(\mathcal{P})$ and $\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v = 1$ is clearly contained in $\text{conv}(\text{epi}(f))$. □

Sharpness is not preserved when $x$ complies with additional constraints, so a property similar to (83) does not hold for (82). However, it is still possible to characterize the LP bound obtained when a sharp formulation is used to model the objective function of a larger model. The following theorem follows directly from the definitions of sharpness and convex envelopes.

**Theorem 4.6.** *Let $D \subset \mathbb{R}^n$ be a polytope, $f : D \to \mathbb{R}$ be a continuous piecewise linear function, $P \subset \mathbb{R}^{n+p+q+1}$ be a sharp binary mixed-integer programming model for* $\text{epi}(f)$ *and $X$ be a compact set. Then* $\min_{x,z,\lambda,y}\{z : (x, z, \lambda, y) \in P, \quad x \in X\} = \min_{x \in X} \text{convenv}_D(f)(x)$.

For the case where $X$ is a polytope this has also been studied in [36] and [38] and together with Theorem 4.5 yields the following corollary.

**Corollary 4.7.** *All formulations from Section 4.3 give the same LP bound for solving* (82).

Now we present the sizes of all the formulations given in Section 4.3. We give the number of extra constraints and extra variables besides $z$ and $x$ and also indicate the number of extra variables that are binary. Table 15 shows this information for all models. Except for Log and MC the sizes are given as a function of $n$, $|\mathcal{P}|$ and the number of vertices $|\mathcal{V}(\mathcal{P})|$ or $|V(P)|$. For MC the size is a function of $n$, $|\mathcal{P}|$ and the number of facets of polytope $P$ denoted by $F(P)$. In particular if $\mathcal{P}$ is a triangulation we have that $|F(P)| \leq n+1$ for all $P \in \mathcal{P}$. For Log the size is a function of $|\mathcal{V}(\mathcal{P})|$ and $|S|$ where $S$ is the branching scheme for the $J_1$ triangulation of $[0,K]^n$. In this case we have $|\mathcal{P}| = K^n n!$ and $|S| = n\lceil \log_2(K) \rceil + n(n-1)/2$, but it is not clear how to explicitly relate these numbers together when $n > 2$. However we can see that $|S|$ grows asymptotically as $\log_2(|\mathcal{P}|)$ only when $n$ is fixed. More specifically, for fixed $n$ we have $|S| \sim \log_2(|\mathcal{P}|)$ (i.e. $\lim_{K\to\infty} |S|/\log_2(|\mathcal{P}|) = 1$) with $|S| = \log_2(|\mathcal{P}|)$ for $K$ of the form $2^r$, but for fixed $K$ we have $\log_2(|\mathcal{P}|) \in o(|S|)$ (i.e. $\lim_{n\to\infty} \log_2(|\mathcal{P}|)/|S| = 0$).

**Table 15:** Sizes of Formulations

| Model | Constraints | Additional Variables | Binaries |
|---|---|---|---|
| DCC | $n + \lvert\mathcal{P}\rvert + 2$ | $\lvert\mathcal{P}\rvert + \sum_{P\in\mathcal{P}} \lvert V(P)\rvert$ | $\lvert\mathcal{P}\rvert$ |
| DLog | $n + 2\lceil \log_2(\lvert\mathcal{P}\rvert)\rceil + 2$ | $2\lceil \log_2(\lvert\mathcal{P}\rvert)\rceil + \sum_{P\in\mathcal{P}} \lvert V(P)\rvert$ | $2\lceil \log_2(\lvert\mathcal{P}\rvert)\rceil$ |
| CC | $n + 3 + \lvert\mathcal{V}(\mathcal{P})\rvert$ | $\lvert\mathcal{V}(\mathcal{P})\rvert + \lvert\mathcal{P}\rvert$ | $\lvert\mathcal{P}\rvert$ |
| Log | $n + 2 + 2\lvert S\rvert$ | $\lvert\mathcal{V}(\mathcal{P})\rvert + \lvert S\rvert$ | $\lvert S\rvert$ |
| MC | $n + 2 + \sum_{P\in\mathcal{P}} F(P)$ | $(n+1)\lvert\mathcal{P}\rvert$ | $\lvert\mathcal{P}\rvert$ |
| Inc | $1 + 2\lvert\mathcal{P}\rvert$ | $\lvert\mathcal{P}\rvert - 1 + \sum_{P\in\mathcal{P}}(\lvert V(P)\rvert - 1)$ | $\lvert\mathcal{P}\rvert - 1$ |

Finally, we summarize the requirements that the different formulations have on the family of polytopes $\mathcal{P}$ used to describe the piecewise linear function. The first type of requirement concerns the description of the polytopes in $\mathcal{P}$ as either the convex hull of a finite number of points (vertex representation) or as the feasible region of a system of linear inequalities (inequality representation). Although conversion between the two descriptions can be done efficiently for special cases of $\mathcal{P}$ such as triangulations, the description requirements can be an important factor in the choice of the formulation when general polytopes are used. We have seen that every formulation except MC uses the vertex representation.

The second type of requirements concerns the need for a particular family of polytopes $\mathcal{P}$. Although requiring $\mathcal{P}$ to be of a special class such as a triangulation is usually not too restrictive, it can be an important factor when the function is constructed as the interpolation of a non-linear function [30, 108]. We have seen that DCC, DLog, CC, and MC have no requirement on $\mathcal{P}$. Inc requires $\mathcal{P}$ to be any triangulation which complies with conditions O1–O2 described in Section 4.3.4 and Log requires $\mathcal{P}$ to be the $J_1$ triangulation.

## *4.5 Computational Experiments for Continuous Functions*

In this section we computationally test the formulations for continuous piecewise linear functions. Our tests are on transportation problems with piecewise linear objective functions. We believe these problems provide enough additional constraints to provide meaningful results while allowing the piecewise linear objectives to dominate the optimization effort.

All models were generated using Ilog Concert 2 and solved using CPLEX 11 on a 2.4GHz workstation with 2GB of RAM. Furthermore, all tests were run with a time limit of 10000 seconds.

### 4.5.1 Continuous Separable Concave Functions

The first set of experiments considers formulations for univariate functions. The instances tested for these formulations are the same transportation problems with concave separable piecewise linear objectives considered in [132]. These instances are based on the $10 \times 10$ transportation problems used in [73] and [131]. Each of the problems include the supply and demand information and capacities $u_e$ for each arc $e$. The problems also include the subdivision of $[0, u_e]$ into 4 randomly selected intervals and their generation is described in [73]. For each of the 5 instances we constructed several randomly generated piecewise linear separable objective functions. These objective functions are of the form $\sum_{e \in E} f_e(x_e)$ where $E$ is the set of arcs of the transportation problem and $f_e(x_e)$ is a continuous non-decreasing concave piecewise linear function of the flow $x_e$ on arc $e$. We chose to use this class of functions because they are widely used in practice and are challenging enough to provide meaningful computational results. Each $f_e(x_e)$ is affine in $K$ segments and has $f_e(0) = 0$. The slopes for each segment of a particular $f_e$ were generated by obtaining

a sample of size $K$ from set $\{z/1000 : z \in \{1, \ldots, 2000\}\}$ and sorting them to ensure concavity. We considered $K = 4$, 8, 16, and 32 and for each $K$ and for each of the 5 transportation problems we generated 20 objective functions for a total of 100 instances for each $K$. To obtain the subdivisions of $[0, u_e]$ into 8, 16 and 32 intervals we simply recursively divided in half each of the intervals starting with the original 4 from [73]. Furthermore, we independently generated the objective functions for each choice of $K$.

We tested all mixed integer formulations from Section 4.3 and in addition we tested the traditional SOS2 formulation of piecewise linear functions (see for example [72]) which does not include binary variables. We implemented this formulation using CPLEX's built in support for SOS2 constraints and we refer to it as SOS2 in the computational results.

Table 16 shows the minimum, average, maximum and standard deviation of the solve times in seconds. The table also shows the number of times the solves failed because the time limit was reached and the number of times each formulation had the fastest solve time (win or tie).

For $K = 4$ we see that the average solve time for all formulations is of the same order of magnitude, but for larger $K$'s the difference between models becomes noticeable. Many conclusions could be extracted from these results, but they should be taken with care as they can depend on both the instances and the solver used. For example, MC is faster on average than Inc for $K$'s ranging from 4 to 16, but in previous tests using CPLEX 9.1 the average solve time for Inc was always better than or comparable to MC. Nevertheless, we make the following observations.

We see that the logarithmic formulations Log and DLog can have a significant advantage over the other formulations (up to over an order of magnitude for $K = 32$) for $K$'s larger than 4 and that, as expected, this advantage grows with $K$. Another interesting observation concerns SOS2, which in previous tests with CPLEX 9.1 was significantly slower than most mixed integer programming formulations. It seems that the reason for this bad performance was more of an implementation issue than a property of the SOS2 based formulation [131]. As the results show, the implementation of SOS2 constraints has

**Table 16:** Solve times for univariate continuous functions [s].

(a) 4 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| MC | 0.2 | **1.3** | 8.3 | 1.5 | 45 | 0 |
| SOS2 | 0.1 | **1.7** | 7.9 | 1.3 | 26 | 0 |
| Log | 0.2 | **2.1** | 12.4 | 2.3 | 24 | 0 |
| DLog | 0.3 | **2.1** | 10.3 | 2.2 | 4 | 0 |
| Inc | 0.4 | **2.4** | 11.6 | 2.5 | 2 | 0 |
| DCC | 0.3 | **2.6** | 14.0 | 2.5 | 0 | 0 |
| CC | 0.3 | **4.6** | 23.0 | 4.3 | 0 | 0 |

(b) 8 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| MC | 1.2 | **9.9** | 39 | 7.0 | 41 | 0 |
| Log | 0.6 | **12.3** | 84 | 10.5 | 31 | 0 |
| DLog | 0.8 | **13.2** | 91 | 11.6 | 5 | 0 |
| SOS2 | 0.8 | **15.8** | 202 | 23.0 | 23 | 0 |
| DCC | 2.6 | **42.7** | 252 | 46.6 | 0 | 0 |
| Inc | 5.1 | **43.0** | 163 | 29.3 | 0 | 0 |
| CC | 2.6 | **81.0** | 570 | 96.6 | 0 | 0 |

(c) 16 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| Log | 0.5 | **24** | 96 | 18 | 80 | 0 |
| DLog | 0.8 | **32** | 132 | 25 | 17 | 0 |
| MC | 1.9 | **97** | 730 | 122 | 2 | 0 |
| SOS2 | 1.9 | **109** | 1030 | 167 | 1 | 0 |
| Inc | 29.8 | **302** | 1442 | 239 | 0 | 0 |
| CC | 3.9 | **351** | 3691 | 517 | 0 | 0 |
| DCC | 3.9 | **1366** | 10000 | 2120 | 0 | 3 |

(d) 32 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| Log | 2.5 | **43** | 194 | 39 | 90 | 0 |
| DLog | 5.5 | **63** | 328 | 53 | 8 | 0 |
| SOS2 | 10.0 | **925** | 10000 | 1900 | 2 | 2 |
| Inc | 271.0 | **981** | 4039 | 685 | 0 | 0 |
| CC | 67.5 | **1938** | 10000 | 2560 | 0 | 4 |
| MC | 22.5 | **2246** | 10000 | 3208 | 0 | 9 |
| DCC | 89.6 | **8163** | 10000 | 3141 | 0 | 69 |

been significantly improved in CPLEX 11 which allows SOS2 to always be among the 5 best formulations. In fact, it is only for $K = 32$ that we have mixed integer formulations outperforming SOS2 by more than an order of magnitude.

In an attempt to explain the results from Table 16 we study some characteristics of the solves by CPLEX. In Tables 17, 18, 19 and 20 we present some results for the instances with $K = 4$, $K = 8$, $K = 16$ and $K = 32$ respectively. Corollary 4.7 states that all MIP formulations should provide the same LP relaxation bound and so should SOS2 [72]. We confirmed this is true up to small numerical errors and, as expected, the common bound was not equal to the optimal MIP solution resulting in an average integrality GAP of 4%, 5%, 6% and 6% (calculated as $100(z_{IP} - z_{LP})/z_{IP}$ where $z_{IP}$ and $z_{LP}$ are the optimal values of the mixed integer program and its LP relaxation respectively) for $K = 4$, $K = 8$, $K = 16$ and $K = 32$ respectively. However, an equality in the LP relaxation bound does not necessarily imply an equality on the LP bound obtained at the root node by CPLEX as this includes preprocessing and cuts. For this reason we present in Tables 18(a), 19(a), 20(a) and 21(a) the percentage of the integrality GAP that was closed by CPLEX at the root node for the different formulations and values of $K$ (this was calculated as $100(z_{root} - z_{LP})/(z_{IP} - z_{LP})$ where $z_{root}$ is the optimal values of the root relaxation obtained by CPLEX after preprocessing and cutting planes). A second issue is the time required to solve the LP relaxation of the different formulations, which we present in Tables 18(b), 19(b), 20(b) and 21(b). Because the solve times were very small for all formulations, the results from these tables are in milliseconds. Finally, in Tables 18(c), 19(c), 20(c) and 21(c) we present the number of nodes processed by CPLEX.

**Table 17:** Solve characteristics for univariate continuous functions and $K = 4$.

(a) GAP closed at root node by CPLEX. [%]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 26 | **58** | 100 | 17.7 |
| DLog | 16 | **37** | 62 | 10.8 |
| Inc | 12 | **37** | 100 | 16.0 |
| DCC | 16 | **36** | 60 | 10.2 |
| Log | 13 | **36** | 61 | 10.0 |
| CC | 11 | **25** | 43 | 6.7 |
| SOS2 | 0 | **0** | 0 | 0.0 |

(b) LP relaxation solve time. [ms]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| SOS2 | 0 | **2.8** | 10 | 2.8 |
| Log | 0 | **5.3** | 10 | 5.3 |
| DLog | 0 | **5.7** | 10 | 5.7 |
| DCC | 0 | **5.9** | 10 | 5.9 |
| CC | 0 | **7.1** | 10 | 7.1 |
| MC | 0 | **9.4** | 20 | 9.4 |
| Inc | 0 | **12.4** | 20 | 12.4 |

(c) Branch-and-bound nodes processed.

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 0 | **234** | 891 | 216 |
| Inc | 1 | **357** | 2081 | 365 |
| DLog | 22 | **504** | 2677 | 529 |
| Log | 14 | **587** | 3569 | 617 |
| DCC | 10 | **798** | 5960 | 897 |
| CC | 30 | **964** | 8938 | 1139 |
| SOS2 | 220 | **1974** | 13434 | 1833 |

**Table 18:** Solve characteristics for univariate continuous functions and $K = 8$.

(a) GAP closed at root node by CPLEX. [%]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 17.4 | **37** | 61.2 | 9.5 |
| DCC | 11.8 | **23** | 38.6 | 6.0 |
| DLog | 4.7 | **20** | 46.7 | 7.2 |
| Log | 6.2 | **19** | 35.6 | 6.6 |
| Inc | 5.3 | **19** | 39.3 | 6.9 |
| CC | 9.3 | **16** | 39.2 | 4.9 |
| SOS2 | 0.0 | **0** | 1.2 | 0.1 |

(b) LP relaxation solve time. [ms]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| SOS2 | 0 | **4.3** | 10 | 4.3 |
| Log | 0 | **9.5** | 20 | 9.5 |
| DCC | 0 | **11.5** | 20 | 11.5 |
| CC | 0 | **11.6** | 20 | 11.6 |
| DLog | 0 | **11.9** | 20 | 11.9 |
| MC | 10 | **24.3** | 40 | 24.3 |
| Inc | 20 | **38.1** | 50 | 38.1 |

(c) Branch-and-bound nodes processed.

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 64 | **535** | 2003 | 301 |
| Inc | 142 | **1970** | 8814 | 1611 |
| DLog | 134 | **2419** | 17114 | 2415 |
| Log | 120 | **2591** | 22541 | 2777 |
| DCC | 549 | **13956** | 120035 | 19253 |
| CC | 500 | **17276** | 127110 | 22467 |
| SOS2 | 606 | **21833** | 337199 | 39081 |

**Table 19:** Solve characteristics for univariate continuous functions and $K = 16$.

(a) GAP closed at root node by CPLEX. [%]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 10.9 | **26** | 53 | 7.2 |
| DCC | 7.1 | **17** | 48 | 5.9 |
| DLog | 2.0 | **17** | 51 | 8.1 |
| Log | 2.0 | **17** | 51 | 7.7 |
| Inc | 2.6 | **14** | 35 | 5.9 |
| CC | 5.6 | **10** | 21 | 2.8 |
| SOS2 | 0.0 | **0** | 0 | 0.0 |

(b) LP relaxation solve time. [ms]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| SOS2 | 0 | **5.9** | 10 | 5.9 |
| Log | 0 | **15.8** | 20 | 15.8 |
| CC | 10 | **23.3** | 40 | 23.3 |
| DLog | 10 | **27.7** | 40 | 27.7 |
| DCC | 10 | **29.9** | 40 | 29.9 |
| MC | 50 | **89.6** | 120 | 89.5 |
| Inc | 90 | **139.0** | 180 | 138.9 |

(c) Branch-and-bound nodes processed.

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 52 | **2809** | 27890 | 4392 |
| DLog | 44 | **4129** | 19900 | 3978 |
| Log | 45 | **4428** | 23921 | 4167 |
| Inc | 204 | **5139** | 25162 | 4118 |
| CC | 245 | **28895** | 241524 | 38696 |
| SOS2 | 1487 | **98050** | 959307 | 155930 |
| DCC | 461 | **302134** | 2345087 | 461223 |

The complexity of CPLEX makes it hard to infer categorical conclusions about these results, but we will comment on some interesting patterns. Note that a larger formulation might have an LP relaxation which is slower to solve, but it might allow CPLEX to close a larger percentage of the integrality GAP. This can lead to fewer branch-and-bound nodes needed to solve the problem, which can translate to faster solve speeds. An example of this behavior is MC, which has the second slowest solve time for its LP relaxation, but allows CPLEX to close the largest percentage of the integrality GAP resulting in the best performances in number of nodes for every $K$ but 32 and in solve times for $K = 4$ and 8. On the other hand, having a small formulation can have the reverse effect on the LP relaxation solve speeds and closed GAP, but might still provide an advantage. For example, SOS2 is one of the smallest formulations as it does not include any binary variables. We can see that CPLEX does not close a significant percentage of the integrality GAP for SOS2, which translates into a need to process a large number of nodes. However, having the fastest solve time for its LP relaxation allows this formulation to still have an excellent performance with respect to solve times. Still, faster solves of its LP relaxation and large percentages of root GAP closed might not necessarily translate to better performance. For example, DCC is on

**Table 20:** Solve characteristics for univariate continuous functions and $K = 32$.

(a) GAP closed at root node by CPLEX. [%]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| MC | 9.5 | **18.8** | 31.4 | 4.8 |
| Log | 1.7 | **15.0** | 32.3 | 6.7 |
| DCC | 5.8 | **13.4** | 25.4 | 3.7 |
| Inc | 1.8 | **9.5** | 24.4 | 4.3 |
| CC | 2.0 | **5.6** | 9.5 | 1.5 |
| DLog | 0.1 | **1.8** | 11.6 | 1.9 |
| SOS2 | 0.0 | **0.0** | 0.0 | 0.0 |

(b) LP relaxation solve time. [ms]

| model | min | **avg** | max | std |
|---|---|---|---|---|
| SOS2 | 0 | **12** | 20 | 11 |
| Log | 10 | **30** | 40 | 30 |
| CC | 20 | **39** | 60 | 39 |
| DLog | 40 | **60** | 70 | 60 |
| DCC | 60 | **93** | 110 | 93 |
| MC | 230 | **418** | 600 | 418 |
| Inc | 410 | **534** | 670 | 534 |

(c) Branch-and-bound nodes processed.

| model | min | **avg** | max | std |
|---|---|---|---|---|
| DLog | 382 | **4776** | 27375 | 4926 |
| Log | 276 | **5287** | 25797 | 5505 |
| Inc | 964 | **8196** | 40352 | 7315 |
| MC | 471 | **28855** | 146197 | 37678 |
| CC | 1762 | **80224** | 505999 | 103995 |
| SOS2 | 2752 | **471156** | 4707352 | 943424 |
| DCC | 5097 | **916227** | 1485910 | 389175 |

average comparable to or better than Inc and DLog with respect to both solve speed of its LP relaxation and GAP closed at the root node. However, Inc and DLog have a better or comparable performance than DCC in terms of both solve times and nodes processed. This is particularly surprising for DLog which is essentially the same as DCC but with fewer variables. A possible explanation for this behavior is that Log, Inc and DLog allow CPLEX to perform a more effective branch-and-bound search. DCC produces unbalanced branch-and-bound trees as fixing a binary variable to zero produces very little change compared to fixing the same variable to one. In contrast, Log and DLog are designed to produce balanced branch-and-bound trees, and Inc also produces a fairly balanced tree since fixing a binary to a particular value in Inc usually fixes many other variables to take the same value.

### 4.5.2 Continuous Non-Separable Functions

We now consider non-separable functions of two variables. For these experiments we selected a series of two commodity transportation problems with 5 supply nodes and 2 demand nodes. These instances were constructed by combining two $5 \times 2$ transportation problems

generated in a manner similar to the instances used in [131]. The supplies, demands and individual commodity arc capacities for each commodity were obtained from two different transportation problems and the joint arc capacities were set to 3/4 of the sum of the corresponding individual arc capacities. We considered an objective function of the form $\sum_{e \in E} f_e(x_e^1, x_e^2)$ where $E$ is the common set of 10 arcs of the transportation problems and $f_e(x_e^1, x_e^2)$ is a piecewise linear function of the flows $x_e^i$ in arc $e$ of commodity $i$ for $i = 1, 2$. Each $f_e(x_e^1, x_e^2)$ for arc $e$ with individual arc capacities $u_e^i$ for commodity $i = 1, 2$ was constructed by triangulating $[0, u_e^1] \times [0, u_e^2]$ with the $J_1$ triangulation induced by the grid obtained from the subdivision of $[0, u_e^1]$ and $[0, u_e^2]$ into $K$ intervals as determined from the respective original transportation problems. For $K$ ranging from 4 to 16 the number of vertices and triangles range from 25 to 289 and from 32 to 512 respectively. Using this triangulation we then obtained $f_e(x_e^1, x_e^2)$ by interpolating $g\left(\left\|\left(x_e^1, x_e^2\right)\right\|\right)$ where $\|\cdot\|$ is the Euclidean norm and $g : \left[0, \left\|\left(u_e^1, u_e^2\right)\right\|\right] \to \mathbb{R}$ is a continuous concave piecewise linear function randomly generated in a similar way to the univariate functions of Section 4.5.1. The idea of this function is to use the sub-linearity of the Euclidean norm to consider discounts for sending the two commodities in the same arc and concave function $g$ to consider economies of scale. We selected 5 combinations of different pairs of the original transportation problems and for each one of these we generated 20 objective functions for a total of 100 instances for each $K$.

Table 21 shows the usual statistics for the solve times with different grid sizes for all the appropriate formulations. We again used a limit of 10000 seconds and only tested a formulation for the next largest $K$ if it had failed in less than 5 instances in the previous $K$.

Logarithmic models Log and DLog were among the best performers for all grid sizes, probably because for two variable functions $|\mathcal{P}|$ grows much faster with $k$ than in the univariate case. For example, for $k = 4$ a $k \times k$ grid yields $|\mathcal{P}| = 32$ which is comparable to $k = 32$ in the univariate case. In addition, the smaller number of continuous variables is what probably allows Log to be the best performer overall.

**Table 21:** Solve times for two variable multi-commodity transportation problems. [s].

(a) $4 \times 4$ grid.

| model | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| Log | 0.4 | **2.7** | 9.3 | 2.0 | 93 | 0 |
| MC | 1.2 | **5.6** | 17.1 | 3.1 | 7 | 0 |
| DLog | 1.6 | **7.6** | 25.5 | 5.2 | 0 | 0 |
| CC | 5.9 | **17.8** | 107.2 | 14.5 | 0 | 0 |
| Inc | 2.8 | **31.7** | 126.5 | 25.8 | 0 | 0 |
| DCC | 8.1 | **36.8** | 476.1 | 50.6 | 0 | 0 |

(b) $8 \times 8$ grid.

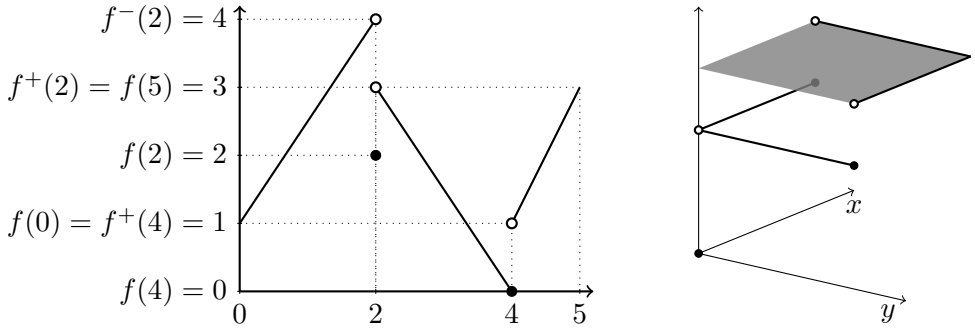
| model | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| Log | 1.7 | **13** | 33 | 5.4 | 100 | 0 |
| DLog | 17.8 | **45** | 135 | 20.2 | 0 | 0 |
| MC | 30.9 | **398** | 5328 | 583.6 | 0 | 0 |
| Inc | 99.5 | **769** | 6543 | 1110.5 | 0 | 0 |
| CC | 102.9 | **4412** | 10000 | 3554.6 | 0 | 13 |
| DCC | 237.0 | **6176** | 10000 | 3385.9 | 0 | 31 |

(c) $16 \times 16$ grid.

| model | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| Log | 27 | **56** | 118 | 19 | 100 | 0 |
| DLog | 125 | **325** | 1064 | 128 | 0 | 0 |
| Inc | 772 | **4857** | 10000 | 3429 | 0 | 20 |
| MC | 2853 | **9266** | 10000 | 1678 | 0 | 77 |

## *4.6 Extension to Lower Semicontinuous Functions*

In this section we study the extension of the formulations to discontinuous functions such as the ones in Figure 14. Consider first the univariate piecewise linear discontinuous function $f$ depicted in Figure 14(a), for which $f^-(d) = \lim_{\substack{x \to d \\ x \leq d}} f(x)$ and $f^+(d) = \lim_{\substack{x \to d \\ x \geq d}} f(x)$. Function $f$ is now only affine in $[0, 2)$, $\{2\}$, $(2, 4]$ and $(4, 5]$. However, because $f$ is lower semicontinuous we have that epi($f$) is closed and is still the union of polyhedra with common recession cone $C_1^+$. Hence we can model epi($f$) as a binary mixed-integer programming problem. The example from Figure 14(a) shows that to consider discontinuous univariate




(a) Univariate Function $f$. (b) Bivariate Function $h$.

**Figure 14:** Lower semicontinuous piecewise linear functions.

piecewise linear functions we need to use intervals that are not necessarily of the form $[d_{i-1}, d_i]$ for $d_{i-1} < d_i$. The inclusion of points described as $\{d\} = [d, d]$ complies with Definition 4.1 as we did not require the polytopes to be full dimensional. In contrast, the inclusion of non closed intervals such as $[0, 2)$ requires the use of sets other than polytopes. The simplest extension we can use is to consider bounded sets that can be described by a finite number of strict and non-strict linear inequalities. These sets are usually referred to as *copolytopes* [70]. Using copolytopes instead of polytopes we get the following definition for not necessarily continuous piecewise linear functions.

**Definition 4.8** (Piecewise Linear Function)**.** *Let $D \subset \mathbb{R}^n$ be a compact set. A (not necessarily continuous) function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ is piecewise linear if and only if there exists a finite family of copolytopes $\mathcal{P}$ complying with $D = \bigcup_{P \in \mathcal{P}} P$ and* (71) *for some*

$\{m_P\}_{P\in\mathcal{P}} \subseteq \mathbb{R}^n$ *and* $\{c_P\}_{P\in\mathcal{P}} \subseteq \mathbb{R}$.

For example, function $f$ from Figure 14(a) can be described as

$$f(x) := \begin{cases} 1.5x + 1 & x \in [0, 1) \\ 2 & x \in [2, 2] \\ -1.5x + 6 & x \in (2, 4] \\ 2x - 7 & x \in (4, 5] \end{cases} \tag{84}$$

and function $h$ from Figure 14(b) can be described as

$$h(x, y) := \begin{cases} 3 & (x, y) \in P_1 \\ 2 & (x, y) \in P_2 \\ 2 & (x, y) \in P_3 \\ 0 & (x, y) \in P_4. \end{cases} \tag{85}$$

for $P_1 = (0, 1]^2$, $P_2 = \{(x, y) \in \mathbb{R}^2 : x = 0,\, y > 0\}$, $P_3 = \{(x, y) \in \mathbb{R}^2 : y = 0,\, x > 0\}$ and $P_4 = \{(0, 0)\}$.

A piecewise linear function as defined in Definition 4.8 is not necessarily lower semicontinuous, but this condition is crucial for obtaining a mixed integer programming model. For a lower semicontinuous piecewise linear function $f$ we have a direct extension of characterization (72) to

$$\operatorname{epi}(f) = C_n^+ + \bigcup_{P\in\mathcal{P}} \operatorname{conv}\left(\{(v, m_P v + c_P)\}_{v\in V(\overline{P})}\right), \tag{86}$$

where $V(\overline{P})$ denotes the set of vertices of the closure $\overline{P}$ of $P$. We note that the closure of a copolytope $P = \{x \in \mathbb{R}^n : a_i x \le b_i \,\forall i \in \{1, \dots, p\},\, a_i x < b_i \,\forall i \in \{p + 1, \dots, m\}\}$ is $\overline{P} = \{x \in \mathbb{R}^n : a_i x \le b_i \,\forall i \in \{1, \dots, m\}\}$. For example, for function $f$ defined in (84) we have

$$\operatorname{epi}(f) = \{(0, r) : r \ge 0\} + \Big(\operatorname{conv}\big(\{(0, 1), (2, 4)\}\big) \cup \operatorname{conv}\big(\{(2, 2)\}\big) \cup \operatorname{conv}\big(\{(2, 3), (4, 0)\}\big) \cup \operatorname{conv}\big(\{(4, 1), (5, 3)\}\big)\Big).$$

This characterization allows some formulations from Section 4.3 to be directly extended to lower semicontinuous functions. Other formulations can be extended by using ad-hoc techniques when the discontinuities considered are simple enough. We study these extensions and techniques in the following subsections. We also comment on the theoretical properties of the resulting formulations.

### 4.6.1 Formulations with Direct Extension

Formulations DCC, DLog and MC directly model epi($f$) so their extension to the lower semicontinuous case is achieved by replacing characterization (72) of epi($f$) for continuous $f$ by characterization (86) of epi($f$) for lower semicontinuous $f$. Because $V(P)$ in (72) is replaced by $V(\overline{P})$ in (86) the extension of DCC is obtained by replacing $V(P)$ by $V(\overline{P})$ in (73). For univariate functions this extension has been noted in [36] and [118]. For function $f$ defined in (84) DCC is given by

$$
\begin{aligned}
0\lambda_{[0,2),0} + 2\left(\lambda_{[0,2),2} + \lambda_{[2,2],2} + \lambda_{(2,4],2}\right) + 4\left(\lambda_{(2,4],4} + \lambda_{(4,5],4}\right) + 5\lambda_{(4,5],5} &= x \\
1\lambda_{[0,2),0} + 4\lambda_{[0,2),2} + 2\lambda_{[2,2],2} + 3\lambda_{(2,4],2} + 0\lambda_{(2,4],4} + 1\lambda_{(4,5],4} + 3\lambda_{(4,5],5} &\leq z \\
\lambda_{[0,2),0},\, \lambda_{[0,2),2},\, \lambda_{[2,2],2},\, \lambda_{(2,4],2},\, \lambda_{(2,4],4},\, \lambda_{(4,5],4},\, \lambda_{(4,5],5} &\geq 0 \\
\lambda_{[0,2),0} + \lambda_{[0,2),2} = y_{[0,2)}, \quad \lambda_{[2,2],2} &= y_{[2,2]}, \\
\lambda_{(2,4],2} + \lambda_{(2,4],4} = y_{(2,4]}, \quad \lambda_{(4,5],4} + \lambda_{(4,5],5} &= y_{(4,5]} \\
y_{[0,2)} + y_{[2,2]} + y_{(2,4]} + y_{(4,5]} = 1, \quad y_{[0,2)},\, y_{[2,2]},\, y_{(2,4]},\, y_{(4,5]} &\in \{0,1\}.
\end{aligned}
$$

Similarly, the extension of DLog is obtained by replacing $V(P)$ by $V(\overline{P})$ in (74). For function $f$ defined in (84) and for $B([0,2)) = (0,0)^T$, $B([2,2]) = (0,1)^T$, $B((2,4]) =$

$(1,1)^T, B((4,5]) = (1,0)^T$ DLog is given by

$$
\begin{aligned}
0\lambda_{[0,2),0} + 2\left(\lambda_{[0,2),2} + \lambda_{[2,2],2} + \lambda_{(2,4],2}\right) + 4\left(\lambda_{(2,4],4} + \lambda_{(4,5],4}\right) + 5\lambda_{(4,5],5} &= x \\
1\lambda_{[0,2),0} + 4\lambda_{[0,2),2} + 2\lambda_{[2,2],2} + 3\lambda_{(2,4],2} + 0\lambda_{(2,4],4} + 1\lambda_{(4,5],4} + 3\lambda_{(4,5],5} &\leq z \\
\lambda_{[0,2),0},\, \lambda_{[0,2),2},\, \lambda_{[2,2],2},\, \lambda_{(2,4],2},\, \lambda_{(2,4],4},\, \lambda_{(4,5],4},\, \lambda_{(4,5],5} &\geq 0 \\
\lambda_{(2,4],2} + \lambda_{(2,4],4} + \lambda_{(4,5],4} + \lambda_{(4,5],5} \leq y_1, \quad \lambda_{[0,2),0} + \lambda_{[0,2),2} + \lambda_{[2,2],2} &\leq (1 - y_1) \\
\lambda_{[2,2],2} + \lambda_{(2,4],2} + \lambda_{(2,4],4} \leq y_2, \quad \lambda_{[0,2),0} + \lambda_{[0,2),2} + \lambda_{(4,5],4} + \lambda_{(4,5],5} &\leq (1 - y_2) \\
y_1,\, y_2 &\in \{0,1\}.
\end{aligned}
$$

The extension of MC is obtained from (78) by replacing (78b) by $A_{\overline{P}}\lambda_P \leq y_P b_{\overline{P}} \quad \forall P \in \mathcal{P}$ where $A_{\overline{P}}\lambda_P \leq b_{\overline{P}}$ is the set of linear inequalities describing polytope $\overline{P}$. For univariate functions this extension has been noted in [36]. For function $f$ defined in (84) MC is given by

$$
\begin{aligned}
x^{[0,2)} + x^{[2,2]} + x^{(2,4]} + x^{(4,5]} &= x \\
(1.5x^{[0,2)} + 1y_{[0,2)}) + (0x^{[2,2]} + 2y_{[2,2]}) + (-1.5x^{(2,4]} + 6y_{(2,4]}) + (2x^{(4,5]} - 7y_{[4,5]}) &\leq z \\
0y_{[0,2)} \leq x^{[0,2)} \leq y_{[0,2)}, \quad 2y_{[2,2]} \leq x^{[2,2]} &\leq 2y_{[2,2]}, \\
2y_{(2,4]} \leq x^{(2,4]} \leq 4y_{(2,4]}, \quad 4y_{(4,5]} \leq x^{(4,5]} &\leq 5y_{(4,5]} \\
y_{[0,2)} + y_{[2,2]} + y_{(2,4]} + y_{(4,5]} = 1, \quad y_{[0,2)},\, y_{[2,2]},\, y_{(2,4]},\, y_{(4,5]} &\in \{0,1\}.
\end{aligned}
$$

### 4.6.2 Ad-Hoc Extension for Univariate Functions

For simple discontinuities we can use ad-hoc techniques to adapt formulations that cannot handle discontinuities directly. We explore two such techniques for univariate functions.

#### *4.6.2.1 Break Point Duplication Technique*

The first technique is from [131] and involves duplicating break points at which a univariate function is discontinuous. For a univariate lower semicontinuous piecewise linear function $f : [0,u] \to \mathbb{R}$ we always have an integer $K$ and real numbers $(d_k)_{k=0}^K$ and $(f_k)_{k=0}^K$ such that $0 = d_0 \leq d_1 \leq \ldots \leq d_K = u$, $f_k$ is equal to $f(d_k)$, $f^-(d_k)$ or $f^+(d_k)$ and

$$
\text{epi}(f) = C_1^+ + \left(\bigcup_{k=1}^{K} \text{conv}\left(\{(d_{k-1}, f_{k-1}), (d_k, f_k)\}\right)\right) \tag{87}
$$

For example, for $f$ defined in (84) characterization (87) is obtained with $K = 6$, $d_0 = 0$, $d_1 = d_2 = d_3 = 2$, $d_4 = d_5 = 4$, $d_6 = 5$, $f_0 = f(0) = 1$, $f_1 = f^-(2) = 4$, $f_2 = f(2) = 2$, $f_3 = f^+(2) = 3$, $f_4 = f(4) = 0$, $f_5 = f^+(4) = 1$ and $f_6 = f(5) = 3$.

Using this characterization (87) we can adapt CC to obtain the formulation given by

$$\sum_{k=0}^{K} \lambda_k d_k = x, \quad \sum_{k=0}^{K} \lambda_k f_k \leq z, \quad \sum_{k=0}^{K} \lambda_k = 1, \quad \lambda_k \geq 0 \quad \forall k \in \{0, \dots, K\} \tag{88a}$$

$$\lambda_0 \leq y_1, \quad \lambda_K \leq y_K, \quad \lambda_k \leq y_k + y_{k+1} \quad \forall k \in \{1, \dots, K-1\}, \tag{88b}$$

$$\sum_{k=1}^{K} y_k = 1, \quad y \in \{0,1\}^K. \tag{88c}$$

We can also adapt Inc, Log and SOS2. For Log we replace (88b) by the corresponding constraints (77c), which in this case are $\sum_{k \in L_s} \lambda_k \leq y_s$, $\sum_{k \in R_s} \lambda_k \leq (1 - y_s)$ and $y_s \in \{0,1\}$ for all $s \in \{1, \dots, \lceil \log_2(K) \rceil\}$, where

$$L_s := \left\{ k \in \{0, \dots, K\} \,:\, \left(k = 0 \text{ or } G^k_l = 1\right) \text{ and } \left(k = K \text{ or } G^{k+1}_l = 1\right) \right\}$$

and

$$R_s := \left\{ k \in \{0, \dots, K\} \,:\, \left(k = 0 \text{ or } G^k_l = 0\right) \text{ and } \left(k = K \text{ or } G^{k+1}_l = 0\right) \right\}$$

for an arbitrary but fixed set of vectors $(G^l)_{l=1}^K \subset \{0,1\}^{\lceil \log_2(K) \rceil}$ that form a Gray code. For Inc we obtain the formulation given by

$$d_0 + \sum_{k=1}^{K} \delta_k \left(d_k - d_{k-1}\right) = x, \quad f_0 + \sum_{k=1}^{K} \delta_k \left(f_k - f_{k-1}\right) \leq z, \tag{89a}$$

$$\delta_1 \leq 1, \quad \delta_K \geq 0, \quad \delta_{k+1} \leq y_k \leq \delta_k, \quad y_k \in \{0,1\} \quad \forall k \in \{1, \dots, K-1\}. \tag{89b}$$

For SOS2 the adaptation is analoguous to the one for CC and is described in [131]. We denote these models CC Dup, Inc Dup, Log Dup and SOS2 Dup. For $f$ defined in (84)

CC Dup is given by

$$0\lambda_0 + 2(\lambda_1 + \lambda_2 + \lambda_3) + 4(\lambda_4 + \lambda_5) + 5\lambda_6 = x,$$

$$1\lambda_0 + 4\lambda_1 + 2\lambda_2 + 3\lambda_3 + 0\lambda_4 + 1\lambda_5 + 3\lambda_6 \leq z$$

$$\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \geq 0, \quad \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 = 1$$

$$\lambda_0 \leq y_0, \quad \lambda_1 \leq y_1 + y_2, \quad \lambda_2 \leq y_2 + y_3, \quad \lambda_3 \leq y_3 + y_4,$$

$$\lambda_4 \leq y_4 + y_5, \quad \lambda_5 \leq y_5 + y_6, \quad \lambda_6 \leq y_6$$

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1, \quad y \in \{0,1\}^6,$$

for $G^1 = (0,0,0)^T$, $G^2 = (1,0,0)^T$, $G^3 = (1,1,0)^T$, $G^4 = (0,1,0)^T$, $G^5 = (0,1,1)^T$, $G^6 = (1,1,1)^T$ Log Dup is given by

$$0\lambda_0 + 2(\lambda_1 + \lambda_2 + \lambda_3) + 4(\lambda_4 + \lambda_5) + 5\lambda_6 = x,$$

$$1\lambda_0 + 4\lambda_1 + 2\lambda_2 + 3\lambda_3 + 0\lambda_4 + 1\lambda_5 + 3\lambda_6 \leq z$$

$$\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \geq 0, \quad \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 = 1$$

$$\lambda_2 + \lambda_6 \leq y_1, \quad \lambda_0 + \lambda_4 \leq (1 - y_1), \quad \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 \leq y_2, \quad \lambda_0 + \lambda_1 \leq (1 - y_2)$$

$$\lambda_5 + \lambda_6 \leq y_3, \quad \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 \leq (1 - y_3), \quad y \in \{0,1\}^3$$

and Inc Dup is given by

$$0 + 2\delta_1 + 0\delta_2 + 0\delta_3 + 2\delta_4 + 0\delta_5 + 1\delta_6 = x,$$

$$1 + 3\delta_1 - 2\delta_2 + 1\delta_3 - 3\delta_4 + 1\delta_5 + 2\delta_6 \leq z$$

$$y_1 \leq \delta_1 \leq 1, \quad y_2 \leq \delta_2 \leq y_1, \quad y_3 \leq \delta_3 \leq y_2,$$

$$y_4 \leq \delta_4 \leq y_3, \quad y_5 \leq \delta_5 \leq y_4, \quad 0 \leq \delta_6 \leq y_5,$$

$$y \in \{0,1\}^5.$$

#### *4.6.2.2 Fixed Charge Technique*

The second technique can be applied when all discontinuities of $f$ are caused by fixed charge type jumps. In this case, $f$ is the sum of a continuous function $f_C$ of the form (69) and a

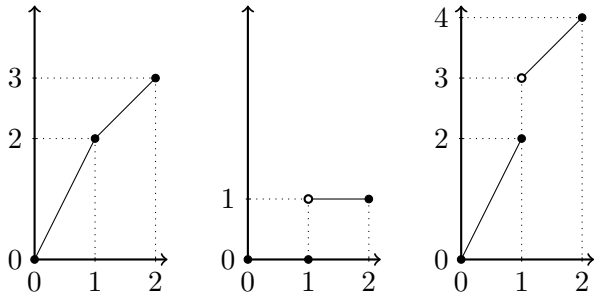
lower semicontinuous non-decreasing step function

$$f_J(x) := \begin{cases} 0 & x = 0 \\ b_k & x \in (d_{k-1}, d_k] \quad \forall k \in \{1, \ldots, K\} \end{cases} \tag{90}$$

for $(d_k)_{k=0}^K \in \mathbb{R}^{K+1}$, $(b_k)_{k=1}^K \in \mathbb{R}_+^K$ such that $0 = d_0 < d_1 < \ldots < d_K = u$ and $0 \leq b_1 \leq b_2 \leq \ldots \leq b_K$. Hence, for $(m_k)_{k=1}^K \in \mathbb{R}^K$ and $(c_k)_{k=1}^K \in \mathbb{R}^K$, $f$ can be described as

$$f(x) := \begin{cases} c_1 & x = 0 \\ m_k x + c_k + b_k & x \in (d_{k-1}, d_k] \quad \forall k \in \{1, \ldots, K\}. \end{cases} \tag{91}$$

This is illustrated by function $\tilde{g} = \tilde{g}_C + \tilde{g}_J$ in Figure 15. $\tilde{g}$ can be described in form (91) for




(a) $\tilde{g}_C$. (b) $\tilde{g}_J$. (c) $\tilde{g}$.

**Figure 15:** Decomposition of fixed charged lower semicontinuous piecewise linear function.

$K = 2$, $d_0 = 0$, $d_1 = 1$, $d_2 = 2$, $m_1 = 2$, $m_2 = 1$, $c_1 = 0$, $c_2 = 1$, $b_1 = 0$ and $b_2 = 1$, which yields

$$\tilde{g}(x) := \begin{cases} 2x & x \in [0, 1] \\ x + 2 & x \in (1, 2]. \end{cases} \tag{92}$$

By using the relation $f = f_C + f_J$ we can construct a model for epi($f$) from models for epi($f_C$) and epi($f_J$). This combination of models is referred to as *model linkage* in [68] where it is shown to computationally perform relatively poorly, in part because formulation sharpness is not preserved by model linkage and in part because of poor coordination between the binary variables of the linked models. Fortunately, as noted in [87], it is sometimes possible to improve model coordination by using ad-hoc techniques. We illustrate this possible coordination by using two specific examples. In both cases we need a lower

semicontinuous function $f : [0, u] \to \mathbb{R}$ which is continuous and zero valued at zero and hence has $0 = c_1 = b_1$ in characterizations (90) and (91). The first coordination is for the model obtained by linking CC and the model of $f_J$ given by

$$\sum_{k=0}^{K} d_k \lambda_{d_k} = x, \quad \sum_{k=1}^{K} b_k w_k \le z, \quad \sum_{k=0}^{K} \lambda_{d_k} = 1, \quad \sum_{k=1}^{K} w_k = 1, \tag{93a}$$

$$0 \le \lambda_{d_0} \le w_1, \quad 0 \le \lambda_{d_K} \le w_K \tag{93b}$$

$$0 \le \lambda_{d_k} \le (w_k + w_{k+1}) \ \forall k \in \{1, \dots, K-1\}, \quad w_k \in \{0, 1\} \ \forall k \in \{1, \dots, K\}. \tag{93c}$$

To coordinate we identify the $\lambda_{d_k}$ variables of the models and force $w_k = y_{[d_{k-1}, d_k]}$. The resulting model is given by

$$\sum_{k=0}^{K} d_k \lambda_{d_k} = x, \ \ \lambda_{d_0} m_1 d_0 + \sum_{k=1}^{K} \Big( \lambda_{d_k} (m_k d_k + c_k) + b_k w_k \Big) \le z, \tag{94a}$$

$$\sum_{k=0}^{K} \lambda_{d_k} = 1, \ \ 0 \le \lambda_{d_0} \le w_1 \tag{94b}$$

$$0 \le \lambda_{d_K} \le w_K, \quad 0 \le \lambda_{d_k} \le (w_k + w_{k+1}) \quad \forall k \in \{1, \dots, K-1\}, \tag{94c}$$

$$\sum_{k=1}^{K} w_k = 1, \quad w \in \{0, 1\}^K. \tag{94d}$$

We refer to this formulation as the coordinated convex combination model and denote it by CC Coord. For $\tilde{g}$ defined in (92) CC Coord is given by

$$0\lambda_0 + 1\lambda_1 + 2\lambda_2 = x, \quad 0w_1 + 1w_2 + 0\lambda_0 + 2\lambda_1 + 3\lambda_2 \le z,$$

$$\lambda_0 + \lambda_1 + \lambda_2 = 1$$

$$\lambda_0 \le w_1, \quad \lambda_1 \le w_1 + w_2, \quad \lambda_2 \le w_2, \quad \lambda_0, \lambda_1, \lambda_2 \ge 0, \quad w_1 + w_2 = 1, \quad w_1, w_2 \in \{0, 1\}$$

A similar coordination can be achieved by linking Inc and another model of $f_J$. The resulting model is given by

$$\sum_{k=1}^{K} \delta_k (d_k - d_{k-1}) = x, \qquad \sum_{k=1}^{K} (m_k d_k - m_k d_{k-1}) \delta_k + \sum_{k=1}^{K-1} (b_{k+1} - b_k) y_k \le z \tag{95a}$$

$$\delta_1 \le 1, \quad \delta_K \ge 0, \quad \delta_{k+1} \le y_k \le \delta_k, \quad y_k \in \{0, 1\} \quad \forall k \in \{1, \dots, K-1\}. \tag{95b}$$

This model has been studied in [71]. We refer to this formulation as the coordinated incremental model and denote it by Inc Coord. For $\tilde{g}$ defined in (92) Inc Coord is given

by

$$\delta_1 + \delta_2 = x, \quad 2\delta_1 + 1\delta_2 + w_1 \leq z, \quad w_1 \leq \delta_1 \leq 1, \quad 0 \leq \delta_2 \leq w_1, \quad w_1 \in \{0,1\}.$$

### 4.6.3 Theoretical Properties of Formulations

Regarding the properties of the formulations, it is direct that Proposition 4.4, Theorem 4.3 and Theorem 4.6 also hold for lower semicontinuous piecewiselinear functions. It is also direct that DCC, DLog and MC remain locally ideal for lower semicontinuous functions, that Inc Dup, Log Dup and Inc Coord are locally ideal and that CC Dup is sharp, but not locally ideal. Finally, it is direct that CC Coord is not locally ideal, but the following proposition holds.

**Proposition 4.9.** *CC Coord is sharp.*

*Proof.* It suffices to show that for $f$ defined in (91) and for any vertex $(\lambda^*, w^*)$ of

$$\sum_{k=0}^{K} \lambda_{d_k} = 1, \quad \sum_{k=1}^{K} w_k = 1, \quad w \in \{0,1\}^K \tag{96a}$$

$$0 \leq \lambda_{d_0} \leq w_1, \quad 0 \leq \lambda_{d_K} \leq w_K, \tag{96b}$$

$$0 \leq \lambda_{d_k} \leq (w_k + w_{k+1}) \quad \forall k \in \{1, \ldots, K-1\} \tag{96c}$$

we have $(x^*, \underline{z}^*) \in \text{conv}(\text{epi}(f))$ for $\underline{z}^* := \lambda^*_{d_0} m_1 d_0 + \sum_{k=1}^{K} \left(\lambda^*_{d_k}(m_k d_k + c_k) + b_k w^*_k\right)$, $x^* := \sum_{k=0}^{K} d_k \lambda^*_{d_k}$.

From Proposition 4 of [82] we have that the vertices of (96) are of the following forms:

1. $\lambda^*_{d_l} = w^*_l = 1, \quad \lambda^*_{d_k} = 0 \quad \forall k \neq l, \quad w^*_k = 0 \quad \forall k \neq l.$
2. $\lambda^*_{d_l} = w^*_{l+1} = 1, \quad \lambda^*_{d_k} = 0 \quad \forall k \neq l, \quad w^*_k = 0 \quad \forall k \neq l+1.$
3. $\lambda^*_{d_{l-1}} = \lambda^*_{d_l} = w^*_l = w^*_{l+1} = 1/2, \quad \lambda^*_{d_k} = 0 \quad \forall k \notin \{l-1, l\}, \quad w^*_k = 0 \quad \forall k \notin \{l, l+1\}.$
4. $\lambda^*_{d_{l-1}} = \lambda^*_{d_l} = w^*_{l-1} = w^*_l = 1/2, \quad \lambda^*_{d_k} = 0 \quad \forall k \notin \{l-1, l\}, \quad w^*_k = 0 \quad \forall k \notin \{l-1, l\}.$

For case 1 we have $x^* = d_l$ and $\underline{z}^* = m_l d_l + c_l + b_l$ so $(x^*, \underline{z}^*) \in \text{epi}(f)$. For case 2 and $l \geq 1$ we have $x^* = d_l$ and $\underline{z}^* = m_l d_l + c_l + b_{l+1} \geq m_l d_l + c_l + b_l$ so $(x^*, \underline{z}^*) \in \text{epi}(f)$. For case 2 and $l = 0$ we have $x^* = d_0$ and $\underline{z}^* = 0$ so $(x^*, \underline{z}^*) \in \text{epi}(f)$. For case 3 we have $x^* = (d_{l-1} + d_l)/2$ and

$\underline{z}^* = (m_{l-1}d_{l-1}+c_{l-1}+b_l+m_l d_l+c_l+b_{l+1})/2 \geq (m_l(d_{l-1}+d_l)/2+c_l+b_l$ so $(x^*, \underline{z}^*) \in \text{epi}(f)$. For case 4 we have $x^* = (d_{l-1}+d_l)/2$ and $\underline{z}^* = (m_{l-1}d_{l-1}+c_{l-1}+b_{l-1}+m_l d_l+c_l+b_l)/2$, but $(d_k, m_k d_k + c_k + b_k) \in \text{epi}(f)$ so $(x^*, \underline{z}^*) \in \text{conv}(\text{epi}(f))$. □

## *4.7 Computational Experiments for Lower Semicontinuous Functions*

In this section we computationally test the MIP formulations for lower semicontinuous piecewise linear functions. We use the same transportation problems from Section 4.5.

### 4.7.1 Discontinuous Separable Functions

The first set of experiments considers formulations for univariate lower semicontinuous functions. The instances tested in this section were obtained from the transportation problems from Section 4.5.1 by modifying functions $f_e(x_e)$ of the flow $x_e$ on arc $e$. Each function $f_e(x_e)$ affine in segments $\{[d_{k-1}, d_k]\}_{k=1}^K$ was transformed into a discontinuous function by adding fixed charge jumps in each of the breakpoints $\{d_k\}_{k=0}^K$. Each jump was randomly generated by independently selecting an integer in $[10, 50]$ using a uniform distribution.

We tested MC, DCC and DLog as they can directly handle lower semicontinuous functions. However, we modified DLog as it initially performed poorly (for $K = 4$ it had an average solve time of 562 seconds and a maximum solve time of 6615 seconds). We believe that this poor performance was due to $|\mathcal{P}|$ not being a power of two (for $K = 4$ we have $\mathcal{P} = \{d_0 = 0, (d_0, d_1], (d_1, d_2], (d_2, d_3], (d_3, d_4]\}$) as this is a common problem with binary encoded formulations [35]. To resolve this we subtracted $f_e^+(0)$ from each function $f_e(x_e)$ and reset the value of $f_e(0)$ to 0. This eliminated the fixed charges at 0 leaving each $f_e(x_e)$ continuous and zero valued at 0. To restore the fixed charges we added a binary variable $y^e \in \{0, 1\}$ for each $e \in E$ with objective coefficient equal to the original fixed charge $f_e^+(0)$ and constraint $x_e \leq u_e y^e$. We also tested CC Coord and Inc Coord with the fixed charge elimination technique because they require functions that are continuous and zero valued at 0. We additionally tested the formulations obtained by applying the break point duplication technique to CC, Log, Inc and SOS2. Additional combinations of models and techniques are not included either because they are redundant (e.g. DCC directly handles lower semicontinuous functions and hence does not require the break point duplication technique)

or because they are not compatible (e.g. we are not aware of any effective coordination technique for Log). Table 22 shows the usual statistics for these instances.

**Table 22:** Solve times for univariate discontinuous functions [s].

(a) 4 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| MC | 0.5 | **5.5** | 30 | 5.2 | 76 | 0 |
| Inc Coord | 0.8 | **7.3** | 40 | 6.3 | 15 | 0 |
| DLog FC | 0.8 | **9.0** | 41 | 6.5 | 6 | 0 |
| Inc Dup | 1.0 | **10.7** | 61 | 8.4 | 3 | 0 |
| Log Dup | 1.0 | **13.0** | 69 | 8.7 | 0 | 0 |
| DCC | 2.0 | **14.8** | 75 | 9.5 | 0 | 0 |
| CC Coord | 1.1 | **15.7** | 116 | 14.1 | 0 | 0 |
| SOS2 Dup | 3.2 | **56.7** | 522 | 75.3 | 0 | 0 |
| CC Dup | 7.4 | **78.9** | 646 | 105.3 | 0 | 0 |

(b) 8 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| MC | 0.0 | **16** | 107 | 23 | 86 | 0 |
| DLog FC | 0.3 | **32** | 123 | 21 | 9 | 0 |
| Log Dup | 2.1 | **43** | 241 | 38 | 4 | 0 |
| Inc Coord | 7.9 | **70** | 298 | 51 | 0 | 0 |
| Inc Dup | 18.7 | **84** | 300 | 51 | 0 | 0 |
| DCC | 0.0 | **366** | 10000 | 1110 | 1 | 1 |
| SOS2 Dup | 8.8 | **476** | 5919 | 853 | 0 | 0 |
| CC Coord | 21.3 | **699** | 5438 | 1014 | 0 | 0 |
| CC Dup | 8.1 | **895** | 10000 | 1644 | 0 | 2 |

(c) 16 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| DLog FC | 23 | **106** | 445 | 88 | 55 | 0 |
| MC | 13 | **263** | 2697 | 401 | 29 | 0 |
| Log Dup | 12 | **331** | 10000 | 1055 | 16 | 1 |
| Inc Coord | 108 | **333** | 2037 | 247 | 0 | 0 |
| Inc Dup | 105 | **405** | 1548 | 278 | 0 | 0 |
| SOS2 Dup | 51 | **1952** | 10000 | 2587 | 0 | 6 |
| CC Dup | 177 | **4409** | 10000 | 3223 | 0 | 18 |
| CC Coord | 342 | **6018** | 10000 | 3624 | 0 | 36 |
| DCC | 110 | **8046** | 10000 | 3551 | 0 | 76 |

(d) 32 segments.

| model | min | **avg** | max | std | win | fail |
|---|---|---|---|---|---|---|
| DLog FC | 54 | **779** | 5395 | 958 | 84 | 0 |
| Inc Coord | 287 | **1586** | 10000 | 1457 | 1 | 1 |
| Inc Dup | 315 | **1935** | 10000 | 1984 | 2 | 4 |
| Log Dup | 77 | **2661** | 10000 | 3268 | 4 | 12 |
| MC | 116 | **4282** | 10000 | 4070 | 9 | 30 |

Again MC is one of the best performers except for $K = 32$ where the logarithmic models again have the advantage. The duplication and coordination techniques only seem

to work well for Inc and Log which were already faster than CC in the continuous case. This could explain their advantage when using the duplication and coordination techniques as well. However, this explanation does not hold for SOS2, which did very well in the continuous case, but performed poorly here.

### 4.7.2 Discontinuous Non-Separable Functions

The set of experiments in this section considers non-separable functions of two variables. The instances tested in this section where obtained from the $5 \times 2$ multi commodity transportation problems from Section 4.5.2 by replacing function $f_e(x_e^1, x_e^2)$ of the flows $x_e^i$ in arc $e$ of commodity $i$ for $i = 1, 2$. To define the new function we use the $K \times K$ grid $\{d_0^1, \ldots, d_K^1\} \times \{d_0^2, \ldots, d_K^2\}$ obtained from the subdivision of $[0, u_e^1]$ and $[0, u_e^2]$ into $K$ intervals as determined from the respective original transportation problems. We select two random samples of size $K$ from set $\{0, 1, \ldots, 10K-1\}$ and sort them in non-increasing order to obtain $(r_k^i)_{k=1}^K$ for each $i = 1, 2$. We then define $s_0^i = 0$ and $s_k^i = r_k^i(d_k^i - d_{k-1}^i) + s_{k-1}^i$ for each $k \in \{1, \ldots, K\}$ and $i = 1, 2$. $f_e(x_e^1, x_e^2)$ is defined as

$$f_e(x_e^1, x_e^2) := \begin{cases} x_e^1 + x_e^2 & (x, y) = (0, 0) \\ x_e^1 + x_e^2 + s_k^1 & x \in (d_{k-1}^1, d_k^1], \quad y = 0 \\ x_e^1 + x_e^2 + s_k^2 & y \in (d_{k-1}^2, d_k^2], \quad x = 0 \\ x_e^1 + x_e^2 + 0.75(s_k^1 + s_l^2) & (x, y) \in (d_{k-1}^1, d_k^1] \times (d_{l-1}^2, d_l^2]. \end{cases}$$

The idea is that for each commodity there is a fixed shipping charge for arc $e$ that depends on the interval $(d_{k-1}^i, d_k^i]$ in which the amount $x_e^i$ shipped falls. We have that this fixed charge divided by the amount shipped is non-increasing because of economies of scale and that if both commodities are shipped through arc $e$ there is a 75% discount on the sum of the fixed charges.

We only tested MC, DCC and DLog as they can handle general lower semicontinuous piecewise linear functions. Table 23 shows the usual statistics for different grid sizes. We again see that MC is always faster than DCC and is only significantly slower than DLog for the largest grids. Finally, we note that the smaller solve times for these instances

**Table 23:** Solve times for non-separable functions [s].

(a) $4 \times 4$ grid.

| stat | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| MC | 0.1 | **2.3** | 8.8 | 1.8 | 97 | 0 |
| DLog | 0.4 | **6.0** | 19.3 | 3.9 | 3 | 0 |
| DCC | 0.9 | **9.9** | 29.8 | 6.6 | 0 | 0 |

(b) $8 \times 8$ grid.

| stat | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| DLog | 1.1 | **17** | 59 | 11 | 51 | 0 |
| MC | 1.0 | **19** | 122 | 18 | 49 | 0 |
| DCC | 8.4 | **83** | 377 | 64 | 0 | 0 |

(c) $16 \times 16$ grid.

| stat | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| DLog | 4.8 | **55** | 201 | 36 | 96 | 0 |
| MC | 10.2 | **209** | 1138 | 195 | 4 | 0 |
| DCC | 51.2 | **890** | 2993 | 542 | 0 | 0 |

(d) $32 \times 32$ grid.

| stat | min | **avg** | max | std | wins | fail |
|---|---|---|---|---|---|---|
| DLog | 56 | **319** | 1385 | 201 | 100 | 0 |
| MC | 151 | **4310** | 10000 | 3780 | 0 | 25 |
| DCC | 1648 | **8504** | 10000 | 2545 | 0 | 65 |

when compared to the ones in Section 4.5.2 could be due to the fact that here the only nonlinearities in the objective functions are fixed charges.

## *4.8 Conclusions*

We studied the modeling of piecewise linear functions as MIPs. We reviewed several new and existing formulations for continuous functions with particular attention paid to their extension to the multivariate non-separable case. We also compared these formulations both with respect to their theoretical properties and their relative computational performance. In addition we studied several ways to extend these formulations to consider lower semicontinuous functions.

Because of the limited computational experiments it is hard to reach categorical conclusions. However there are several trends that, combined with the theoretical properties of the formulations, provide general guidelines for the use of the different formulations by practitioners. For example, when the number of polytopes defining the piecewise linear

function is small MC seems to be one of the best choices. Furthermore it seems to be always preferable to CC and DCC. Another example concerns functions defined by a large number of polytopes. In this case the sizes of logarithmic formulations DLog and Log can give them a significant computational advantage. Finally, for lower semicontinuous functions it seems that, with the exception of SOS2 Dup, special ad-hoc techniques only provide an advantage when they are used to adapt formulations that already performed well in the continuous case.

# CHAPTER V

# MIXED INTER LINEAR PROGRAMMING FORMULATIONS FOR LINEAR PROGRAMMING WITH PROBABILISTIC CONSTRAINTS

## *5.1 Introduction*

Let $\xi$ be a $d$-dimensional random vector with finite support on $\{\xi^1, \ldots, \xi^S\} \subset \mathbb{R}^d_+$ such that $\mathbb{P}(\xi = \xi^s) = 1/S$ for each $s \in \{1, \ldots, S\}$. We consider Mixed Integer Linear Programming (MILP) formulations for the Linear Programming (LP) problem with a joint probabilistic constraints given by

$$\max \qquad cx \tag{97a}$$

$$s.t.$$

$$\mathbb{P}\big(x \geq \xi\big) \geq 1 - \delta \tag{97b}$$

$$x \in \mathbb{R}^d_+ \tag{97c}$$

$$x \in X \tag{97d}$$

where $c \in \mathbb{R}^d$, $X \subset \mathbb{R}^d_+$ is a polyhedron and $\delta \in (0, 1)$. Note that (97) can also consider probabilistic constraints of the form $\mathbb{P}\big(Ax \geq \xi\big) \geq 1 - \delta$ for a deterministic matrix $A$ by appropriately modifying $X$ and $\{\xi^1, \ldots, \xi^S\}$.

LPs with joint probabilistic or chance constraints of the form (97b) for arbitrary distributions of $\xi$ have been extensively studied and have many applications (see for example [109, 117] and the references within). The discrete distribution case has been studied in [16, 45, 74, 89, 90, 116] and used in applications in [16, 17, 83, 91, 114]. In particular, [114] studies MILP formulations for binary distributions and [74, 89, 90, 113] study MILP formulation for general finite distributions. Discrete finite distributions also appear naturally in Sample Average Approximations (SAA) of general probabilistic constraints [88].

A standard MILP formulation of probabilistic constraint (97b) [113] is

$$x \geq (1 - z_s)\xi^s \quad \forall s \in \{1, \ldots, S\} \tag{98a}$$

$$\sum_{s=1}^{S} z_s \leq \lfloor \delta S \rfloor, \tag{98b}$$

$$z_s \in \{0, 1\} \qquad \forall s \in \{1, \ldots, S\}. \tag{98c}$$

This formulation uses binary variables $z \in \{0,1\}^S$ such that $z_s = 0$ if $x \geq \xi^s$ and $z_s = 1$ if $x \not\geq \xi^s$ and restricts the number of violated $x \geq \xi^s$ inequalities through cardinality constraint (98b). Formulation, (98) can be very weak, so valid inequalities for it have been developed in [74, 89, 90]. These valid inequalities significantly strengthen the formulation, but are deduced by only considering one row of d-row system (98a) at a time. In this chapter we study the strength of these 1-row valid inequalities and evaluate the potential advantage of developing valid inequalities that consider more than one row at a time.

In Section 5.2 we review existing MILP formulations for constraint (97b). In Section 5.3 we introduce an extended MILP formulation of (97b) that generalizes a formulation introduced in [74] for the case $d = 1$. In Section 5.4 we study the strength of the 1-row relaxation theoretically and in Section 5.5 we compare the strength of the 1-row and 2-row relaxations computationally. Finally, in Section 5.6 we present some conclusions.

## *5.2 Existing MILP Formulations*

We now review existing MILP formulations for $Q_x := \{x \in \mathbb{R}^d : \mathbb{P}\big(x \geq \xi\big) \geq 1 - \delta\}$.

Let $k := \lfloor \delta S \rfloor$ and for each $x \in \mathbb{R}^d$ let $v(x) := \{s \in \{1, \ldots, S\} : x \not\geq \xi^s\}$ be the index set for which $x$ violates constraint $x \geq \xi$. We then have that $Q_x = \{x \in \mathbb{R}^d : |v(x)| \leq k\}$ and hence we have the following disjunctive characterization

$$Q_x = \bigcup_{\substack{J \subset \{1,\ldots,S\} \\ |J| = S-k}} \Big\{ x \in \mathbb{R}^d : x \geq \xi^s \quad \forall s \in J \Big\}. \tag{99}$$

Then using Corollary 2.1.2 of [9] we can construct an extended MILP formulation of $Q_x$ that is as strong as any MILP formulation for $Q_x$. Unfortunately, the size of this formulation is $\Theta\left(d\binom{S}{S-k}\right)$ and hence exponential in $S$ for fixed $\delta$, which makes it computationally

impractical. The number of disjunctions of the right hand side of (99) can be significantly reduced by considering so-called $(1-\delta)$-efficient points [16, 45, 109, 116]. A point $p \in \mathbb{R}^d$ is $(1-\delta)$-efficient if

$$\mathbb{P}(p \geq \xi) \geq 1-\delta \text{ and } \mathbb{P}(p-q \geq \xi) < 1-\delta \quad \forall q \in \mathbb{R}^d_+ \setminus \{\mathbf{0}\},$$

which for a finite distribution is equivalent to

$$|v(p)| \leq k \text{ and } |v(p-q)| > k \quad \forall q \in \mathbb{R}^d_+ \setminus \{\mathbf{0}\}.$$

By letting $\mathcal{B}$ be the set of $(1-\delta)$-efficient points of $Q_x$ we have that

$$Q_x = \bigcup_{b \in \mathcal{B}} \{x \in \mathbb{R}^d \,:\, x \geq b\}. \tag{100}$$

The number of disjunctions in (100) is never larger than the number in (99) and as the following example shows it is usually much smaller.

**Example 5.1.** *Let* $d = 2$, $S = 5$, $\xi^1 = (0, 20)$, $\xi^2 = (10, 10)$, $\xi^3 = (20, 0)$, $\xi^4 = (11, 21)$, $\xi^5 = (21, 11)$ *and* $k = 3$. *This data is illustrated in Figure 16, where the solid lines indicate the borders of sets* $\{x \geq \xi^i\}$ *and* $Q_x$ *is given by the shaded region. The number of disjunctions in* (99) *is* 10 *and the* $(1-\delta)$*-efficient points* $(10, 20)$ *and* $(20, 10)$ *are circled in the figure.*

In Section 5.3 we will see that although $|\mathcal{B}|$ is not exponential in $S$ it is exponential in $d$. Although it is not practical to use these disjunctive formulations directly they can be used as a base for specialized algorithms [16, 44, 45, 46] or to construct valid inequalities [116, 113].

Formulation (98) is a much smaller but weaker alternative to the disjunctive formulations (99) and (100). However, it can be significantly strengthened with the valid inequalities developed in [74, 89, 90]. These inequalities are all based on a 1-row relaxation of (98) that we review in the following section.

### 5.2.1 1-row Relaxation

We denote the feasible region of formulation (98) by $Q$ so that $Q_x$ is the projection of $Q$ onto the $x$ variables. We can strengthen formulation (98) by adding strong valid inequalities for

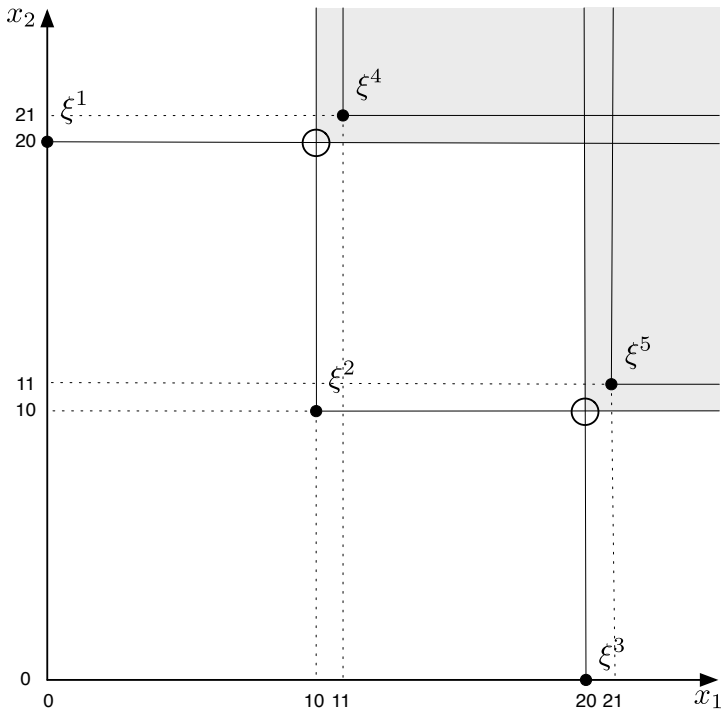



**Figure 16:** Example 5.1.

conv($Q$), but this set can be extremely complicated. Instead of studying conv($Q$) directly we can study a natural relaxation of $Q$ given by $Q_j := \{(x_j, z) \in \mathbb{R} \times \{0,1\}^S : x_j \geq (1 - z_s)\xi_j^s \quad \forall s \in \{1, \ldots, S\}, \quad \sum_{s=1}^S z_s \leq k\}$. $Q_j$ is the projection of $Q$ onto the $(x_j, z)$ variables and $Q = \bigcup_{j=1}^d \{(x, z) \in \mathbb{R}^d \times \{0,1\}^S : (x_j, z) \in Q_j\}$ so we can strengthen $Q$ with valid inequalities for conv($Q_j$). $Q_j$ only considers one row of constraints (98a) so we refer to conv($Q_j$) as a *1-row* relaxation of $Q$ and to its valid inequalities as *1-row* valid inequalities

A simple strengthening from studying conv($Q_j$) can be obtained by noting that if $x_j \in Q_j$ then $x_j \geq \xi_j^{[k+1]_j}$ where $\xi_j^{[1]_j} \geq \xi_j^{[2]_j} \geq \ldots \xi_j^{[S]_j}$ [89, 90]. This requirement can also be deduced by studying the one-dimensional marginal distributions of $\xi$ [16, 44] and yields

$$Q_x \subset M_x := \left\{ x \in \mathbb{R}^d : x_j \geq \xi_j^{[k+1]_j} \quad \forall j \in \{1, \ldots, d\} \right\}. \tag{101}$$

We refer to $M_x$ as the *marginal* relaxation of $Q_x$. By using $M_x$ formulation (98) is strengthened in [89, 90] to

$$x_j \geq \xi_j^s + (\xi_j^{[k+1]_j} - \xi_j^s) z_s \quad \forall s \in \{[1]_j, \ldots, [k]_j\},\ j \in \{1, \ldots, n\} \tag{102a}$$

$$\sum_{s=1}^{S} z_s \leq k \tag{102b}$$

$$z_s \in \{0, 1\} \qquad \forall s \in \{1, \ldots, S\}. \tag{102c}$$

Formulation (102) is additionally strengthened in [74, 89, 90] by adding strong 1-row valid inequalities. These 1-row valid inequalities could be considered to be somewhat myopic, but computational results in [74, 89, 90] show that they are extremely effective. This raises the question of how good is the relaxation obtained by adding all 1-row inequalities and if it is worth developing multi-row inequalities. To answer these questions we need to be able to study $H_{x,z} := \left\{(x, z) \in \mathbb{R}^d \times [0, 1]^S : (x_j, z) \in \text{conv}(Q_j)\, \forall j \in \{1, \ldots, d\}\right\}$ and its projection onto the $x$ variables that we denote by $H_x$. Fortunately, extended formulations for $Q$ developed in [74, 89, 90] are such that their LP relaxations are equivalent to $H_{x,z}$. We describe the smaller of these formulations in the next section.

### 5.2.2 Extended 1-row Formulation

We present a compact extended formulation for $Q_j$ introduced in [74]. For simplicity, we first fix $j$ and assume that $\xi_j^1 \geq \xi_j^2 \geq \ldots \geq \xi_j^S$. Let

$$Q_j^r := \left\{(x_j, z) \in \mathbb{R} \times \{0, 1\}^S : x_j = \xi^r, \quad z_s = 1\, \forall s \leq r - 1, \quad \sum_{s=r}^{S} z_s \leq (k - r + 1)\right\} \cap Q_j \tag{103}$$

be the set of feasible solutions to $Q_j$ such that $x_j = \xi^r$. We have that $Q_j^r = \emptyset$ for all $r > k+1$ and

$$Q_j = \left\{(x_j, z) \in \mathbb{R}^{S+1} : x_j \geq 0, \quad z_s = 0, \forall s \in \{1, \ldots, S\}\right\} + \bigcup_{r=1}^{k+1} Q_j^r. \tag{104}$$

Formulation (10) in Chapter 1 for disjunctive set (104) results in the MILP formulation for $Q_j$ given by

$$x_j \geq \sum_{r=1}^{k+1} y_r \xi_j^{r_j} \tag{105a}$$

$$\sum_{r=1}^{k+1} y_r = 1 \tag{105b}$$

$$0 \leq y_r \leq 1 \qquad \forall r \in \{1, \ldots, k+1\} \tag{105c}$$

$$w_{s,r} \geq 0 \qquad r \in \{1, \ldots, k+1\},\ s \in \{1, \ldots, S\} \tag{105d}$$

$$w_{s,r} \leq y_r \qquad r \in \{1, \ldots, k+1\},\ s \in \{1, \ldots, S\} \tag{105e}$$

$$w_{s,r} \geq y_r \qquad r \in \{1, \ldots, k+1\},\ s \in \{1, \ldots, r-1\} \tag{105f}$$

$$\sum_{s=r}^{S} w_{s,r} \leq y_r(k-r+1) \quad \forall r \in \{1, \ldots, k+1\} \tag{105g}$$

$$z_s = \sum_{r=1}^{k+1} w_{s,r} \qquad \forall s \in \{1, \ldots, S\} \tag{105h}$$

$$0 \leq z_s \leq 1 \qquad \forall s \in \{1, \ldots, S\} \tag{105i}$$

$$z \in \mathbb{Z}^S \tag{105j}$$

$$y \in \mathbb{Z}^{k+1} \tag{105k}$$

Variables $w$ in this formulation are the copies of the $z$ variables for each polytope on the right hand side of (104) and variables $y$ indicate which one of these polytopes is selected.

The extreme points of $Q_j^r$ have integral $z$ variables because it is a totally unimodular system. Hence, by Theorem 1.1 in Chapter 1 we have that the LP relaxation of (105) given by (105a)–(105i) has extreme points that naturally comply with the integrality requirements.

A formulation for $Q$ is also constructed in [74] by combining formulations (105) for each $j \in \{1, \ldots, d\}$. This formulation does not require assumption $\xi_j^1 \geq \xi_j^2 \geq \ldots \xi_j^S$ and is given

by

$$x_j \geq \sum_{r=1}^{k+1} y_{j,r} \xi_j^{[r]_j} \qquad \forall j \in \{1, \ldots, d\} \tag{106a}$$

$$\sum_{r=1}^{k+1} y_{j,r} = 1 \qquad \forall j \in \{1, \ldots, d\} \tag{106b}$$

$$0 \leq y_{j,r} \leq 1 \qquad \forall r \in \{1, \ldots, k+1\},\ j \in \{1, \ldots, d\} \tag{106c}$$

$$w_{j,s,r} \geq 0 \qquad r \in \{1, \ldots, k+1\},\ s \in \{1, \ldots, S\},\ j \in \{1, \ldots, d\} \tag{106d}$$

$$w_{j,s,r} \leq y_{j,r} \qquad r \in \{1, \ldots, k+1\},\ s \in \{1, \ldots, S\}\ j \in \{1, \ldots, d\} \tag{106e}$$

$$w_{j,[s]_j,r} \geq y_{j,r} \qquad r \in \{1, \ldots, k+1\},\ s \in \{1, \ldots, r-1\},\ j \in \{1, \ldots, d\} \tag{106f}$$

$$\sum_{s=r}^{S} w_{j,[s]_j,r} \leq y_{j,r}(k-r+1) \quad \forall r \in \{1, \ldots, k+1\},\ j \in \{1, \ldots, d\} \tag{106g}$$

$$z_s = \sum_{r=1}^{k+1} w_{j,s,r} \qquad \forall s \in \{1, \ldots, S\},\ j \in \{1, \ldots, d\} \tag{106h}$$

$$0 \leq z_s \leq 1 \qquad \forall s \in \{1, \ldots, S\} \tag{106i}$$

$$z \in \mathbb{Z}^s \tag{106j}$$

$$y_{j,r} \in \mathbb{Z} \qquad \forall r \in \{1, \ldots, k+1\},\ j \in \{1, \ldots, d\} \tag{106k}$$

The extreme points of the LP relaxation of (106) given by (106a)–(106i) do not necessarily comply with the integrality requirements. However, it is clear that the projection onto the $(x, z)$ variables of the LP relaxation of (106) is equal to $H_{x,z}$. Formulation (106) is compact in the sense that it has a polynomial number of variables and constraints: $S + d(2 + k + S + kS)$ variables and $S + d/2(8 + 5k + k^2 + 2(2 + k)S)$ constraints except for non-negativity. Furthermore, Theorem 8 of [74] shows that formulation (106) is at least as strong as formulation (102).

Another extended formulation for $Q$ was given earlier in [89, 90], but this early formulation has an exponential number of constraints. We note however, that formulation (105) can also be obtained by using Corollary 1 of [90] and Proposition 1 of [97].

### 5.2.3 Blending

1-row inequalities for general MILPs are usually combined with row aggregation to obtain stronger results (e.g. [93]). An analogous procedure for 1-row valid inequalities for conv($Q$) was introduced in [74], where it was referred to as *blending*. The procedure can be described as follows.

For any $\pi \in \mathbb{R}^d_+$ [74] introduces the *blending set* given by $Q(\pi) := \{(y,z) \in \mathbb{R} \times \{0,1\}^S : y \geq (1-z_s)\pi^T\xi^s \quad \forall s \in \{1,\ldots,S\}, \quad \sum_{s=1}^S z_s \leq k\}$. The interest of the blending set is that if $y \geq \delta^T z + \delta_0$ is a valid inequality for $Q(\pi)$ then $\pi^T x \geq \delta^T z + \delta_0$ is a valid inequality for $Q$. When $\pi = e^j$ the blending set gives the usual 1-row relaxation $Q_j$ of $Q$, but it was shown in [74] that other choices of $\pi$ can yield inequalities that cannot be obtained from the 1-row relaxations. For $d = 2$ [74] also gives simple necessary conditions on $\pi$ for $(\bar{x},\bar{z}) \notin \text{conv}(Q(\pi))$ when $(\bar{x}_j,\bar{z}) \in \text{conv}(Q_j)$ for $j \in \{1,2\}$.

## *5.3 Extended Formulation for* $d > 1$

We now construct an extended formulation for $Q$ that generalizes formulation (105) for $Q_j$ by considering multiple rows of the probabilistic constraint system. The first step of this generalization is to identify the $d$-dimensional analog of $Q^r_j$ defined in (103). For $g \in \mathbb{R}^d$ this analog is the set of feasible solutions to $Q$ such that $x = g$ given by

$$Q^g := \left\{ (x,z) \in \mathbb{R}^d \times \{0,1\}^S : x = g, \quad z_s = 1 \, \forall s \in v(g), \quad \sum_{s \notin v(g)} z_s \leq (k - |v(g)|) \right\} \tag{107}$$

where $v(g) = \{s \in \{1,\ldots,S\} : g \not\geq \xi^s\}$ is the set of realizations of $\xi$ for which $g$ violates constraint $g \geq \xi$. The second step is to select an appropriate set of points $g$ to construct the analog for $Q_x$ of disjunctive characterization (104) of $Q_j$. An initial candidate set is $G = Q_x \cap \prod_{j=1}^d \{\xi_j^{[1]_j}, \ldots, \xi_j^{[k+1]_j}\}$ for which we clearly have

$$Q = \left\{(x,z) \in \mathbb{R}^{S+d} : x \geq 0, \quad z_s = 0, \forall s \in \{1,\ldots,S\}\right\} + \bigcup_{g \in G} Q^g, \tag{108}$$

but, as the following simple proposition shows, we can reduce the size of $G$ while maintaining validity of (108).

**Proposition 5.1.** *Let*

$$\tilde{G} := \bigcup_{l=0}^{k} \Big\{ x \in \mathbb{R}^d : v(p) \leq l \text{ and } v(p-q) > l \quad \forall q \in \mathbb{R}^d_+ \setminus \{\mathbf{0}\}. \Big\} \tag{109}$$

*be the set of points that are $(1-\delta')$-efficient points for some $\delta' \in [0, \delta)$. Then*

$$Q = \big\{(x,z) \in \mathbb{R}^{S+d} : x \geq 0, \quad z_s = 0, \forall s \in \{1,\dots,S\}\big\} + \bigcup_{g \in \tilde{G}} Q^g, \tag{110}$$

*Proof.* Let $(x^0, z^0) \in Q$ and let $s_j := \arg\max_{s=1}^{S} \{\xi_j^s : x_j^0 \geq \xi_j^s\}$ and $\xi^0 \in \mathbb{R}^d$ such that $\xi_j^0 := \xi_j^{s_j}$. Let $k_0 := |v(\xi^0)|$. Then $\xi^0$ is $(1 - k_0/S)$-efficient, and hence in $\tilde{G}$, and $(x^0, z^0) \in Q^{\xi^0} + \big\{(x,z) \in \mathbb{R}^{S+d} : x \geq 0, \quad z_s = 0, \forall s \in \{1,\dots,S\}\big\}$. Hence

$$(x^0, z^0) \in \big\{(x,z) \in \mathbb{R}^{S+d} : x \geq 0, \quad z_s = 0, \forall s \in \{1,\dots,S\}\big\} + \bigcup_{g \in \tilde{G}} Q^g.$$

The reverse inclusion is direct. □

We can again use formulation (10) in Chapter 1 for disjunctive set (110) to get the MILP formulation of $Q$ given by

$$x \geq \sum_{g \in \tilde{G}} y_g g \tag{111a}$$

$$\sum_{g \in \tilde{G}} y_g = 1 \tag{111b}$$

$$0 \leq y_g \leq 1 \qquad \forall g \in \tilde{G} \tag{111c}$$

$$w_{s,g} \geq 0 \qquad g \in \tilde{G},\ s \in \{1,\dots,S\} \tag{111d}$$

$$w_{s,g} \leq y_g \qquad g \in \tilde{G},\ s \in \{1,\dots,S\} \tag{111e}$$

$$w_{s,g} \geq y_g \qquad g \in \tilde{G},\ s \in v(g) \tag{111f}$$

$$\sum_{s \notin v(g)} w_{s,g} \leq y_g(k - |v(g)|) \quad \forall g \in \tilde{G} \tag{111g}$$

$$z_s = \sum_{g \in \tilde{G}} w_{s,g} \qquad \forall s \in \{1,\dots,S\} \tag{111h}$$

$$0 \leq z_s \leq 1 \qquad \forall s \in \{1,\dots,S\} \tag{111i}$$

$$z \in \mathbb{Z}^S \tag{111j}$$

$$y_g \in \mathbb{Z} \qquad \forall g \in \tilde{G}. \tag{111k}$$

We also have that the extreme points of $Q^g$ have integral $z$ variables because it is a totally unimodular system so by Theorem 1.1 in Chapter 1 we have that the LP relaxation of (111) given by (111a)–(111i) also has extreme points that naturally comply with the integrality requirements.

The size of formulation (111) is $\Theta(|\tilde{G}|*S)$ and because $\tilde{G} \subset G$ we have that $|\tilde{G}|$ is $O(k^d)$. However, $|\tilde{G}|$ is usually strictly smaller than $|G|$. For instance, in Example 5.1 we have that $G = \{(10,21),\ (11,21),\ (20,21),\ (21,21),\ (10,20),\ (11,20),\ (20,20),\ (21,20),\ (20,11),\ (21,11),\ (20,10),\ (21,10)\}$, but $\tilde{G} = G \setminus \{(10,21),(11,20),(21,10),(20,11)\}$. Furthermore, for $\xi$'s with special structures $|G|$ can be significantly smaller than $|\tilde{G}|$. For example, if $\xi^1 \geq \xi^2 \geq \ldots \geq \xi^S$ then $|G| = (k+1)^d$, but $\tilde{G} = \{\xi^s\}_{s=1}^{k+1}$ and (111) reduces to the formulation in Theorem 9 of [74]. Unfortunately, as the following example shows, it is also easy to construct instances for which $|\tilde{G}|$ is $\Omega(k^d)$.

**Example 5.2.** *Let $d \geq 3$, $S = d\,k$ and $\{\xi^s\}_{s=1}^{S} := \bigcup_{j=1}^{d}\bigcup_{l=1}^{k}\{\xi \in \mathbb{R}^d : \xi_j = l, \quad \xi_i = 0\,\forall i \neq j\}$. Then the $(1-k/S)$-efficient points are the solutions to*

$$\sum_{i=1}^{d} x_i = (d-1)k \tag{112a}$$

$$x \in \mathbb{Z}^d_+. \tag{112b}$$

*The solutions to* (112) *are the so-called* weak $s$-compositions *of $(d-1)k$ of which there are exactly $\binom{(d-1)k+d-1}{d-1}$ (e.g. [121, p.15]). Hence the number of $(1-k/S)$-efficient points is at least $(1+k)^{d-1}$ and $|\tilde{G}|$ is $\Omega(k^d)$ because $\tilde{G}$ contains all $(1-k/S)$-efficient points.*

Because of its size, formulation (111) is only useful for very small values of $d$. Fortunately, in a similar way to the construction of formulation (106) we can combine several copies of formulation (111) for small values of $d$ to obtain a formulation of $Q$ for large $d$. To achieve this we select set $D_l \subset \{1,\ldots,d\}$ for $l \in \{1,\ldots,L\}$ such that $\cup_{l=1}^{L} D_l = \{1,\ldots,d\}$. Then for each $g \in \mathbb{R}^{D_l}$ we let $v_l(g) := \{s \in \{1,\ldots,S\} : \exists j \in D_l \text{ s.t. } g_l < \xi^s_l\}$ and for each $l \in \{1,\ldots,L\}$ we let

$$\tilde{G}_l := \bigcup_{l=0}^{k} \left\{ x \in \mathbb{R}^{D_l} : v_l(x) \leq l, \quad v_l(x-q) > l\,\forall q \in \mathbb{R}^{D_l}_+ \setminus \{\mathbf{0}\} \right\}. \tag{113}$$

Using these sets we obtain the formulation of $Q$ given by

$$x_j \geq \sum_{g \in \tilde{G}_l} y_g g_j \qquad \forall j \in D_l, \quad l \in \{1, \ldots, L\} \tag{114a}$$

$$\sum_{g \in \tilde{G}_l} y_g = 1 \qquad \forall l \in \{1, \ldots, L\} \tag{114b}$$

$$0 \leq y_g \leq 1 \qquad \forall g \in \tilde{G}_l,\, l \in \{1, \ldots, L\} \tag{114c}$$

$$w_{s,g} \geq 0 \qquad g \in \tilde{G}_l,\, s \in \{1, \ldots, S\} \tag{114d}$$

$$w_{s,g} \leq y_g \qquad g \in \tilde{G}_l,\, s \in \{1, \ldots, S\} \tag{114e}$$

$$w_{s,g} \geq y_g \qquad g \in \tilde{G}_l,\, s \in v_l(g) \tag{114f}$$

$$\sum_{s \notin v_l(g)}^{S} w_{s,g} \leq y_g(k - |v_l(g)|) \quad \forall g \in \tilde{G}_l \tag{114g}$$

$$z_s = \sum_{g \in \tilde{G}_l} w_{s,g} \qquad \forall s \in \{1, \ldots, S\}, \quad l \in \{1, \ldots, L\} \tag{114h}$$

$$0 \leq z_s \leq 1 \qquad \forall s \in \{1, \ldots, S\} \tag{114i}$$

$$z_s \in \mathbb{Z} \qquad \forall s \in \{1, \ldots, S\} \tag{114j}$$

$$y_g \in \mathbb{Z} \qquad \forall g \in \tilde{G}_l,\, l \in \{1, \ldots, L\}. \tag{114k}$$

If we let $x_{D_l} := (x_j)_{j \in D_l} \subset \mathbb{R}^{|D_l|}$, it is straightforward that the projection onto the $(x, z)$ variables of the LP relaxation of (114) given by (114a)–(114i) is equal to

$$H_{x,z}\left(\{D_l\}_{l=1}^{L}\right) := \left\{(x, z) \in \mathbb{R}^d \times [0,1]^S \,:\, (x_{D_l}, z) \in \operatorname{conv}(Q_{D_l})\, \forall l \in \{1, \ldots, L\}\right\} \tag{115}$$

where

$$Q_{D_l} := \{(x_{D_l}, z) \in \mathbb{R}^{|D_l|} \times \{0,1\}^S \,:\, \sum_{s=1}^{S} z_s \leq k,$$
$$x_j \geq (1 - z_s)\xi_j^s \quad \forall s \in \{1, \ldots, S\}, j \in D_l\}. \tag{116}$$

### *5.4 Strength of 1-row Relaxation*

We now study the strength of 1-row relaxations of $Q$. Our aim is to understand the advantages of MILP formulations of $Q_x$ whose LP relaxation is equal or close to $H_{x,z}$ such as (106) or (102) strengthened by the valid inequalities from [74, 89, 90] respectively.

The strength of an MILP formulation of disjunctive sets such as $Q_x$ is usually evaluated by two possible properties. The first property is to require that the extreme points of the LP relaxation of the MILP naturally comply with the MILP's integrality requirements. A MILP that has this property while modeling the disjunctive set in the absence of additional constraints is usually referred to as *locally ideal* [105, 106]. In our case, a formulation of $Q_x$ whose LP relaxation is equal to $H_{x,z}$ will be locally ideal if $H_{x,z} = \text{conv}(Q)$. The second property is slightly weaker as it only considers the original variables of the disjunctive set. This property is to require that the projection of the LP relaxation of the MILP formulation onto the original variables is equal to the convex hull of the disjunctive set. A formulation that complies with this property is usually referred to as *sharp* [67, 87]. In our case, a formulation of $Q_x$ whose LP relaxation is equal to $H_{x,z}$ will be locally ideal if $H_x = \text{conv}(Q_x)$. Because optimizing linear functions over $Q_x$ is NP-hard [90] we would not expect formulations with $H_{x,z}$ as their LP relaxation to be either locally ideal or sharp in general. However, the favorable computational results in [74, 89, 90] suggest that these formulations could be almost sharp for some classes of problems.

We begin our study with some negative results by showing that in the worst case $H_x$ is not only far from $\text{conv}(Q_x)$, but that it can be arbitrarily close to marginal relaxation $M_x$. We also show that, as expected, adding valid inequalities obtained through the blending procedure introduced in [74] will not always yield $H_{x,z} = \text{conv}(Q)$ or $H_x = \text{conv}(Q_x)$. We then present some positive results showing that formulations with $H_{x,z}$ as their LP relaxation can be sharp or close to sharp for some simple cases.

### 5.4.1 Negative results

The following example shows that $H_x$ can be arbitrarily close to marginal relaxation $M_x$.

**Example 5.3.** *Let $\varepsilon > 0$, $L \geq 0$, $d = 2$, $S = 2m$, $k = S - 1$, $\xi^s = (D + \varepsilon(s-1), D + M - \varepsilon(s-1))$ for all $s \in \{1, \ldots, m\}$ and $\xi^s = (D + M - \varepsilon(s-m-1), D + \varepsilon(s-m-1))$ for all $s \in \{m+1, \ldots, 2m\}$. Figure 17 illustrates this data for $L = 0$.*

*The marginal relaxation for this data is $M_x = \{x \in \mathbb{R}^n : x_1, x_2 \geq L\}$. We now show that by varying $m$ and $\varepsilon$ we can obtain a solution $(\tilde{x}_1, \tilde{x}_2, \tilde{z}) \in H_{x,z}$ such that $(\tilde{x}_1, \tilde{x}_2)$ is as*

*close as desired to* $(L, L)$*, the shortest element in* $M_x$*. For simplicity we assume* $L = 0$*. The case* $L > 0$ *follows directly by a simple translation.*

*For every* $i \in \{1, \ldots, m\}$ *we have that* $(x_1^i, z^i)$ *given by* $x_1^i = \varepsilon(i-1)$*,* $z_i^i = 0$ *and* $z_j^i = 1$ *for* $j \neq i$ *is in* $Q_1$*. We also have that* $(x_1^\alpha, z^\alpha)$ *given by* $x_1^\alpha = M$*,* $z_j^\alpha = 1$ *for* $j \in \{1, \ldots, m\}$ *and* $z_j^\alpha = 0$ *for* $j \in \{m+1, \ldots, 2m\}$ *is also in* $Q_1$*. Hence* $(\tilde{x}_1, \tilde{z}) = 1/(m+1)(x_1^\alpha, z^\alpha) + \sum_{i=1}^m 1/(m+1)(x_1^i, z^i) \in \mathrm{conv}(Q_1)$*. Similarly, for every* $i \in \{m+1, \ldots, 2m\}$ *we have that* $(x_2^i, z^i)$ *given by* $x_2^i = \varepsilon(i-1)$*,* $z_i^i = 0$ *and* $z_j^i = 1$ *for* $j \neq i$ *is in* $Q_2$*. We also have that* $(x_2^\beta, z^\beta)$ *given by* $x_2^\beta = M$*,* $z_j^\beta = 0$ *for* $j \in \{1, \ldots, m\}$ *and* $z_j^\beta = 1$ *for* $j \in \{m+1, \ldots, 2m\}$ *is also in* $Q_2$*. Hence* $(\tilde{x}_2, \tilde{z}) = 1/(m+1)(x_1^\beta, z^\beta) + \sum_{i=m+1}^{2m} 1/(m+1)(x_1^i, z^i) \in \mathrm{conv}(Q_2)$*. Then,* $(\tilde{x}_1, \tilde{x}_2, \tilde{z}) \in H_{x,z}$ *and* $(\tilde{x}_1, \tilde{x}_2) \in H_x$*. We also have that* $\tilde{x}_1 = \tilde{x}_2 \leq (m(m-1)\varepsilon + M)/(m+1)$ *so by taking* $\varepsilon = 1/m^2$ *we get that* $\tilde{x}_1 = \tilde{x}_2 \leq (m - 1 + mM)/(m + m^2) \xrightarrow{m \to \infty} 0$*.*

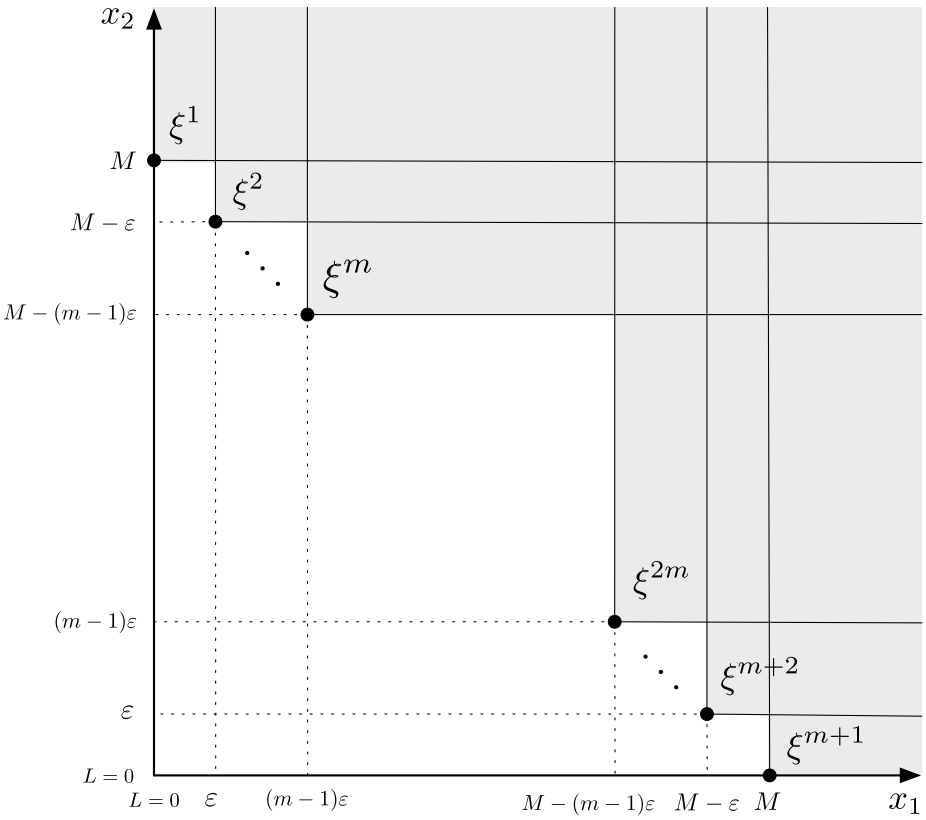



**Figure 17:** Example 5.3 for $L = 0$.

Example 5.3 can be modified to obtain examples with other characteristics. For example,

the example still works if we take $\{\xi^i\}_{i=1}^m$ and $\{\xi^i\}_{i=m+1}^{2m}$ to be any set of points in $[0,(m-1)\varepsilon] \times [M-(m-1)\varepsilon, M]$ and $[M-(m-1)\varepsilon, M] \times [0,(m-1)\varepsilon]$ respectively such that $\xi_2^1 = \max_{i=1}^m \xi_2^i$ and $\xi_1^{m+1} = \max_{i=m+1}^{2m} \xi_1^i$. For $L > 0$ the example also works if we add points $\{\xi_i\}_{i=2m+1}^{(2m-1)p} \subset \{x \in \mathbb{R}_+^2 : x_1, x_2 < L\}$ and change $k$ to $S/p$ for $p \in \mathbb{Z}$.

The blending procedure described in Section 5.2.3 can be used to strengthen $H_{x,z}$. However, as the following example shows, it does not always yield $\text{conv}(Q_x)$.

**Example 5.4.** *Let* $d = 2$, $S = 5$, $\xi^1 = (0,20)$, $\xi^2 = (10,10)$, $\xi^3 = (20,0)$, $\xi^4 = (11,21)$, $\xi^5 = (21,11)$ *and* $k = 3$. *Figure 16 illustrates this data.*

*For this case we have that* $(x_1^1, z^1) = (10,0,0,1,1,1)$, $(x_1^2, z^2) = (11,1,0,1,0,1)$ *and* $(x_1^3, z^3) = (21,1,0,0,1,0)$ *are all in* $Q_1$ *so* $(\tilde{x}_1, \tilde{z}) = (14, 2/3, 0, 2/3, 2/3, 2/3) = 1/3(x_1^1, z^1) + 1/3(x_1^2, z^2) + 1/3(x_1^2, z^2) \in \text{conv}(Q_1)$. *Similarly, we have that* $(x_2^4, z^4) = (10,1,0,0,1,1)$, $(x_2^5, z^5) = (11,1,0,1,1,0)$ *and* $(x_2^6, z^6) = (21,0,0,1,0,1)$ *are all in* $Q_2$ *so* $(\tilde{x}_2, \tilde{z}) = (14,2/3,0,2/3,2/3,2/3) = 1/3(x_2^4, z^4) + 1/3(x_2^5, z^5) + 1/3(x_2^6, z^6) \in \text{conv}(Q_2)$. *Hence* $(\tilde{x}_1, \tilde{x}_2, \tilde{z}) = (14,14,2/3,0,2/3,2/3,2/3) \in H_{x,z}$ *and then* $(14,14) \in H_x$. *We also have that* $Q_x = \{x \in \mathbb{R}^2 : x_1 \geq 10,\, x_2 \geq 20\} \cup \{x \in \mathbb{R}^2 : x_1 \geq 20,\, x_2 \geq 10\}$ *and hence* $(14,14) \notin \text{conv}(Q_x)$.

*We now show that* $(\hat{y}, \hat{z}) = (\pi_1 14 + \pi_2 14, 2/3, 0, 2/3, 2/3, 2/3) \in \text{conv}(Q(\pi))$ *for all* $\pi \in \mathbb{R}_+^2$. *If* $\pi_1 = 0$ *or* $\pi_2 = 0$ *we have that* $Q(\pi) = Q_1$ *or* $Q(\pi) = Q_2$ *so the result follows directly. We divide the remaining possibilities into the following cases*

*(a)* $0 < \pi_2 \leq 3/7\pi_1 \Leftrightarrow \pi^T\xi^1 < \pi^T\xi^2 < \pi^T\xi^4 \leq \pi^T\xi^3 < \pi^T\xi^5$

*(b)* $0 < 3/7\pi_1 < \pi_2 \leq \pi_1 \Leftrightarrow \pi^T\xi^1 \leq \pi^T\xi^2 \leq \pi^T\xi^3 < \pi^T\xi^4 \leq \pi^T\xi^5$

*(c)* $0 < \pi_1 \leq 3/7\pi_2 \Leftrightarrow \pi^T\xi^3 < \pi^T\xi^2 < \pi^T\xi^5 \leq \pi^T\xi^1 < \pi^T\xi^4$

*(d)* $0 < 3/7\pi_2 < \pi_1 \leq \pi_2 \Leftrightarrow \pi^T\xi^3 \leq \pi^T\xi^2 \leq \pi^T\xi^1 < \pi^T\xi^5 \leq \pi^T\xi^4$

*For case* (a) *we have that* $(y^1, z^1) = (\pi^T\xi^2, 0,0,1,1,1)$, $(y^2, z^2) = (\pi^T\xi^4, 1,0,1,0,1)$ *and* $(y^3, z^3) = (\pi^T\xi^5, 1,0,0,1,0)$ *are all in* $Q(\pi)$ *so* $(\hat{y}, \hat{z}) = 1/3(y^1, z^1) + 1/3(y^2, z^2) + 1/3(y^2, z^2) \in \text{conv}(Q(\pi))$. *For case* (b) $(y^4, z^4) = (\pi^T\xi^2, 0,0,1,1,1)$, $(y^5, z^5) = (\pi^T\xi^4, 1,0,0,0,1)$ *and* $(y^6, z^6) = (\pi^T\xi^5, 1,0,1,1,0)$ *are all in* $Q(\pi)$ *so* $(\hat{y}, \hat{z}) = 1/3(y^4, z^4) + 1/3(y^5, z^5) + 1/3(y^6, z^6) \in \text{conv}(Q(\pi))$. *Cases* (c) *and* (d) *follow from the symmetry of the data.*

In contrast to the results in Example 5.4, it is possible for the blending procedure to strengthen $H_{x,z}$ to the point that its projection onto the $x$ variables is exactly $\text{conv}(Q_x)$. However, as the next example shows, even when this condition hold the blending procedure might still not give $\text{conv}(Q)$.

**Example 5.5.** *Let* $d = 2$, $S = 4$, $\xi^1 = (0, 10)$, $\xi^2 = (1, 11)$, $\xi^3 = (10, 0)$, $\xi^4 = (11, 1)$ *and* $k = 3$. *Figure 18 illustrates this data.*

*Using Porta [33] we can check that* $\text{conv}(Q_1)$ *is given by*

$$
\begin{aligned}
x_1 \geq\ & 11 - z_2 - 9z_3 - z_4 && \text{(117a)}\\
x_1 \geq\ & 11 - z_2 - 10z_4 && \text{(117b)}\\
x_1 \geq\ & 11 - 10z_3 - z_4 && \text{(117c)}\\
x_1 \geq\ & 11 - 11z_4 && \text{(117d)}\\
x_1 \geq\ & 10 + z_1 - 9z_3 - z_4 && \text{(117e)}\\
x_1 \geq\ & 10 + z_1 - 10z_4 && \text{(117f)}\\
x_1 \geq\ & -8 + 10z_1 + 9z_2 - z_4 && \text{(117g)}\\
3 \geq\ & z_1 + z_2 + z_3 + z_4 && \text{(117h)}\\
z_i \in\ & [0, 1] \qquad \forall i \in \{1, \ldots, 4\} && \text{(117i)}
\end{aligned}
$$

*and* $\text{conv}(Q_2)$ *is given by*

$$
\begin{aligned}
x_2 \geq\ & 11 - 9z_1 - z_2 - z_4 && \text{(118a)}\\
x_2 \geq\ & 11 - 10z_2 - z_4 && \text{(118b)}\\
x_2 \geq\ & 11 - 10z_1 - z_2 && \text{(118c)}\\
x_2 \geq\ & 11 - 11z_2 && \text{(118d)}\\
x_2 \geq\ & 10 - 9z_1 - z_2 + z_3 && \text{(118e)}\\
x_2 \geq\ & 10 - 10z_2 + z_3 && \text{(118f)}\\
x_2 \geq\ & -8 - z_2 + 10z_3 + 9z_4 && \text{(118g)}\\
3 \geq\ & z_1 + z_2 + z_3 + z_4 && \text{(118h)}\\
z_i \in\ & [0, 1] \qquad \forall i \in \{1, \ldots, 4\}. && \text{(118i)}
\end{aligned}
$$

*We can also check that* $(\bar{x}_1, \bar{x}_2, \bar{z}) = (4, 4, 2/3, 2/3, 2/3, 2/3)$ *is feasible for* (117a)–(118i) *and* $\bar{x} \notin \text{conv}(Q^x) = \{x \in \mathbb{R}^2 : x_1 + x_2 \geq 10, \quad x_1, x_2 \geq 0\}$. *To obtain* $\text{conv}(Q_x)$ *we can use the blending procedure for* $\bar{\pi}_1 = \bar{\pi}_2 = 1$. *Using Porta we can check that* $\text{conv}(Q(\bar{\pi}))$ *is given by*

$$x_1 + x_2 \geq \quad 12 \qquad -2z_2 \tag{119a}$$

$$x_1 + x_2 \geq \quad 12 \qquad\qquad -2z_4 \tag{119b}$$

$$x_1 + x_2 \geq \quad 8 + 2z_1 \qquad +2z_3 \tag{119c}$$

$$3 \geq \qquad z_1 + \ z_2 + \ z_3 + \ z_4 \tag{119d}$$

$$z_i \in [0, 1] \qquad\qquad \forall i \in \{1, \ldots, 4\} \tag{119e}$$

*and that the extreme points* $(\hat{x}, \hat{z})$ *of* (117a)–(119e) *are all such that* $\hat{x} \in \text{conv}(Q_x)$. *However,* $(\tilde{x}, \tilde{z}) = (11/2, 11/2, 1/2, 1/2, 1/2, 1/2)$ *is an extreme points of* (117a)–(119e), *which shows that these inequalities do not give* $\text{conv}(Q)$.

*We now show that* $(\hat{y}, \hat{z}) = (\pi_1 11/2 + \pi_2 11/2, 1/2, 1/2, 1/2, 1/2) \in \text{conv}(Q(\pi))$ *for all* $\pi \in \mathbb{R}^2_+$. *If* $\pi_1 = 0$ *or* $\pi_2 = 0$ *we have that* $Q(\pi) = Q_1$ *or* $Q(\pi) = Q_2$ *so the result follows directly. We divide the remaining possibilities into the following cases*

*(a)* $0 < \pi_2 \leq 9/11\pi_1 \Leftrightarrow \pi^T\xi^1 < \pi^T\xi^2 \leq \pi^T\xi^3 < \pi^T\xi^4$

*(b)* $0 < 9/11\pi_1 < \pi_2 < \pi_1 \Leftrightarrow \pi^T\xi^1 \leq \pi^T\xi^3 < \pi^T\xi^2 \leq \pi^T\xi^4$

*(c)* $0 < \pi_1 \leq 9/11\pi_2 \Leftrightarrow \pi^T\xi^3 < \pi^T\xi^4 \leq \pi^T\xi^1 < \pi^T\xi^2$

*(d)* $0 < 9/11\pi_2 < \pi_1 < \pi_2 \Leftrightarrow \pi^T\xi^3 \leq \pi^T\xi^1 < \pi^T\xi^4 \leq \pi^T\xi^2$

*For case* (a) *we have that* $(y^1, z^1) = (\pi^T\xi^1, 0, 1, 1, 1)$ *and* $(y^2, z^2) = (\pi^T\xi^4, 1, 0, 0, 0)$ *are in* $Q(\pi)$ *so* $(\hat{y}, \hat{z}) = 1/2(y^1, z^1) + 1/2(y^2, z^2) \in \text{conv}(Q(\pi))$. *For case* (b) *we also have that* $(y^1, z^1)$ *and* $(y^2, z^2)$ *are in* $Q(\pi)$ *so* $(\hat{y}, \hat{z}) = 1/2(y^1, z^1) + 1/2(y^2, z^2) \in \text{conv}(Q(\pi))$. *Cases* (c) *and* (d) *follow from the symmetry of the data.*

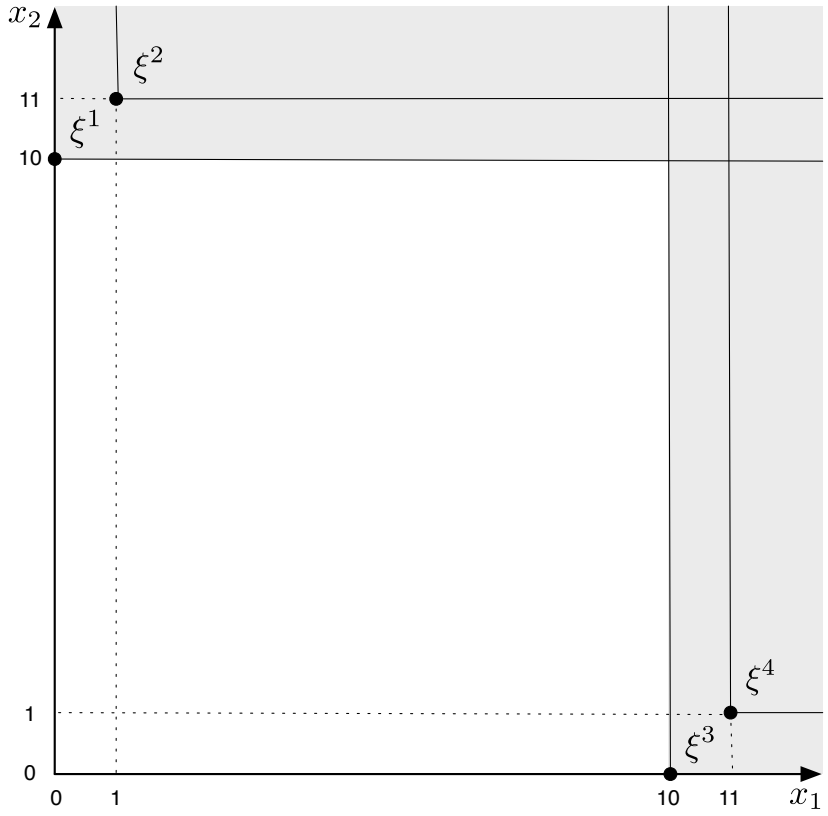



**Figure 18:** Example 5.5.

### 5.4.2 Positive Results

We now give some positive results concerning the strength of $H_x$. We first show that for simple cases $H_x$ is sharp and we then show that for an interesting example $H_x$ is nearly sharp in a very specific sense.

#### *5.4.2.1 Simple Structures for $d = 2$*

We now show that formulation (106) is sharp for simple configurations with $d = 2$. To achieve this we study how formulation (106) could fail to be sharp for simple configurations. For example, suppose that $d = 2$, $S = 2$, $k = 1$, $\xi^1 = (a_1, b_2)$ and $\xi^2 = (a_2, b_2)$ for $a_1 < a_2$ and $b_1 < b_2$. As illustrated in Figure 19, for this simple two point configuration we have

that $x \in \text{conv}(Q_x)$ if and only if

$$x_1 \geq \lambda a_1 + (1-\lambda)a_2,$$
$$x_2 \geq \lambda b_2 + (1-\lambda)b_1$$

for $\lambda \in [0,1]$, but constraints (106a)–(106c) of formulation (106) only force

$$x_1 \geq \lambda a_1 + (1-\lambda)a_2 \tag{120a}$$
$$x_2 \geq \mu b_1 + (1-\mu)b_2 \tag{120b}$$

for $\lambda$, $\mu \in [0,1]$, which allows $x$ to be equal to $(a_1, b_1) \notin \text{conv}(Q_x)$.

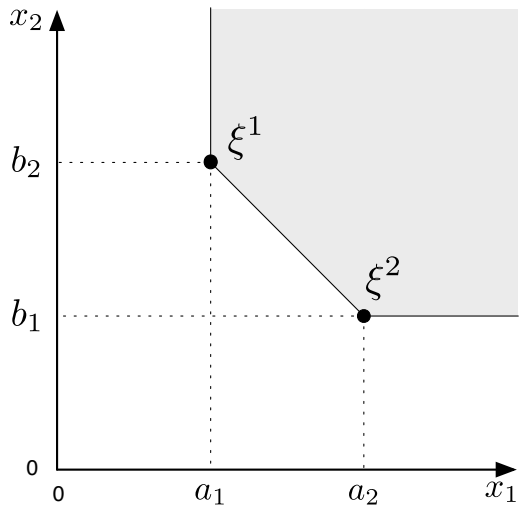



**Figure 19:** Simple Configuration.

To prove that formulation (106) is sharp for this case we need to show that its remaining constraints restrict convex combination multipliers $\lambda$, $\mu$ in (120) to only yield points in $\text{conv}\left(\{(a_1, b_2), (a_2, b_1)\}\right)$ on the right hand side of (120). The following lemma gives sufficient requirements on the multipliers to achieve this condition for the two point configuration in Figure 19 and a similar three point configuration.

**Lemma 5.2.** *(a) Let $a_1$, $a_2$, $b_1$, $b_2 \in \mathbb{R}$ such that $a_1 < a_2$ and $b_1 < b_2$. Also let $(x_1, x_2) \in$*

$\mathbb{R}^2$ *be such that*

$$x_1 \geq \lambda a_1 + (1-\lambda)a_2 \tag{121}$$

$$x_2 \geq \mu b_1 + (1-\mu)b_2 \tag{122}$$

$$1 \geq \lambda + \mu \tag{123}$$

*for* $\lambda, \mu \in \mathbb{R}_+$. *Then*

$$\operatorname{conv}\left(\{(a_1, b_2), (a_2, b_1)\}\right) + \mathbb{R}^2_+$$

*(b) Let* $a_1$, $a_2$, $a_3$, $b_1$, $b_2$, $b_3 \in \mathbb{R}$ *such that* $a_1 < a_2 < a_3$ *and* $b_1 < b_2, b_3$. *Also let* $(x_1, x_2) \in \mathbb{R}^2$ *be such that*

$$x_1 \geq \lambda_1 a_1 + \lambda_2 a_2 + (1-\lambda_1-\lambda_2)a_3 \tag{124}$$

$$x_2 \geq \mu_1 b_1 + \mu_2 b_2 + (1-\mu_1-\mu_2)b_3 \tag{125}$$

$$1 \geq \mu_1 + \mu_2 + \lambda_1 \tag{126}$$

$$1 \geq \lambda_1 + \lambda_2 + \mu_1 \tag{127}$$

*for* $\lambda, \mu \in \mathbb{R}_+$. *Then*

$$\operatorname{conv}\left(\{(a_1, b_3), (a_2, b_2), (a_3, b_1)\}\right) + \mathbb{R}^2_+$$

*Proof.* For (a) the result follows because by multiplying (123) by $(a_2 - a_1) > 0$ and adding it to (122) we get $x_1 \geq \mu a_2 + (1-\mu)a_1$.

For (b) the result follows because by multiplying (126) by $(a_2 - a_1) > 0$ and (127) by $(a_3 - a_2) > 0$ and adding them to (124) we get $x_1 \geq \mu_1 a_3 + \mu_2 a_2 + (1-\mu_1-\mu_2)a_1$.

□

Using this lemma we can show that if the structure associated to the marginals is simple then (106) is sharp.

**Proposition 5.3.** *Let* $s_1 = [k+1]_1$ *and* $s_2 = [k+1]_2$. $H_x = \operatorname{conv}(Q_x)$ *if any of the following conditions hold*

1. $s_1 = s_2$.

2. $\exists j \in \{1,2\}$ *such that* $\xi_j^{s_1} = \xi_j^{s_2}$.

3. *For every* $s \in \{1,\dots,S\} \setminus \{s_1, s_2\}$ *we have that*

$$\xi^s \leq \xi^{s_i}\, \forall i \in \{1,2\} \text{ or } \xi^s \geq \xi^{s_i}\, \forall i \in \{1,2\}. \tag{128}$$

4. *There is a single* $s_0 \in \{1,\dots,S\}$ *such that* (128) *holds for every* $s \in \{1,\dots,S\} \setminus \{s_0, s_1, s_2\}$, $\xi_1^{s_1} < \xi_1^{s_0} < \xi_1^{s_2}$ *and* $\xi_2^{s_1} > \xi_2^{s_0} > \xi_2^{s_2}$.

*Proof.* The cases of Proposition 5.3 are depicted in Figure 20.

For cases 1 and 2 we have that $M_x = \text{conv}(Q_x)$ so the result follows from $M_x \subset H_x$.

For case 3 without loss of generality we may assume that $\xi_1^{s_1} < \xi_1^{s_2}$ and $\xi_2^{s_2} < \xi_2^{s_1}$. In this case we have that, thanks to constraint (106b)–(106c), a solution to the LP relaxation of (106) has

$$x_1 \geq \sum_{r=1}^{k+1} y_{1,r}\xi_1^{[r]_1} \geq y_{1,k+1}\xi_1^{s_1} + (1 - y_{1,k+1})\xi_1^{s_2} \tag{129}$$

$$x_2 \geq \sum_{r=1}^{k+1} y_{2,r}\xi_2^{[r]_2} \geq y_{2,k+1}\xi_2^{s_2} + (1 - y_{2,k+1})\xi_2^{s_1} \tag{130}$$

Now, because $s_2 = [k]_1$ we have that constraint (106d), (106f) and (106h) imply

$$y_{1,k+1} \leq w_{1,s_2,k+1} \leq z_{s_2}. \tag{131}$$

In addition, we have that constraint (106g) implies that $w_{2,s_2,k+1} = 0$ and hence constraints (106h), (106e) and (106b) imply

$$z_{s_2} = \sum_{r=1}^{k} w_{2,s_2,r} \leq \sum_{r=1}^{k} y_{2,r} = 1 - y_{2,k+1}. \tag{132}$$

We then have that $y_{1,k+1} + y_{2,k+1} \leq 1$ and the result follows from Lemma 5.2 part (a) and the fact that $\text{conv}(Q_x) = \text{conv}\left(\{\xi^{s_1}, \xi^{s_2}\}\right) + \mathbb{R}^2_+$.

For case 4 we have that, thanks to constraint (106b)–(106c), a solution to the LP relaxation of (106) has

$$x_1 \geq \sum_{r=1}^{k+1} y_{1,r}\xi_1^{[r]_1} \geq y_{1,k+1}\xi_1^{s_1} + y_{1,k}\xi_1^{s_0} + (1 - y_{1,k+1} - y_{1,k})\xi_1^{s_2} \tag{133}$$

$$x_2 \geq \sum_{r=1}^{k+1} y_{2,r}\xi_2^{[r]_2} \geq y_{2,k+1}\xi_2^{s_2} + y_{1,k}\xi_2^{s_0} + (1 - y_{2,k+1} - y_{1,k})\xi_2^{s_1} \tag{134}$$

Now, because $s_2 = [k-1]_1$ we have that constraint (106d), (106f) and (106h) imply

$$y_{1,k+1}, y_{1,k} \leq w_{1,s_2,k+1} \leq z_{s_2}. \tag{135}$$

In addition, we again have that constraint (106g) implies that $w_{2,s_2,k+1} = 0$ and hence constraints (106h), (106e) and (106b) imply (132). We then have that

$$y_{1,k+1} + y_{1,k} + y_{2,k+1} \leq 1 \tag{136}$$

By a symmetric argument we also have that

$$y_{2,k+1} + y_{2,k} + y_{1,k+1} \leq 1. \tag{137}$$

The result follows from Lemma 5.2 part (b) and the fact that $\text{conv}(Q_x) = \text{conv}\left(\{\xi^{s_1}, \xi^{s_2}, \xi^{s_0}\}\right) + \mathbb{R}^2_+$.

□

#### *5.4.2.2 Integers in Box*

We now study the case in which $\{\xi_s\}_{s=1}^S$ are all the points with integer coordinates in a box $\prod_{j=1}^d [1, M_j] \subset \mathbb{R}^d$ for $M \in \mathbb{Z}^d$. These points can be thought as the equally probable realizations of the random variable $\xi = \lceil \eta \rceil$ for $\eta$ uniformly distributed in $\prod_{j=1}^d [0, M_j]$. Because the distribution function of $\eta$ is log-concave on $\mathbb{R}^d$ we have that the distribution of $\xi$ is log-concave on $\mathbb{Z}^d$ and hence $Q_x \cap \mathbb{Z}^d = \text{conv}(Q_x) \cap \mathbb{Z}^d$ [45, 117]. Furthermore, this property allows us to precisely characterize $Q_x$ as a mixed integer nonlinear programming problem.

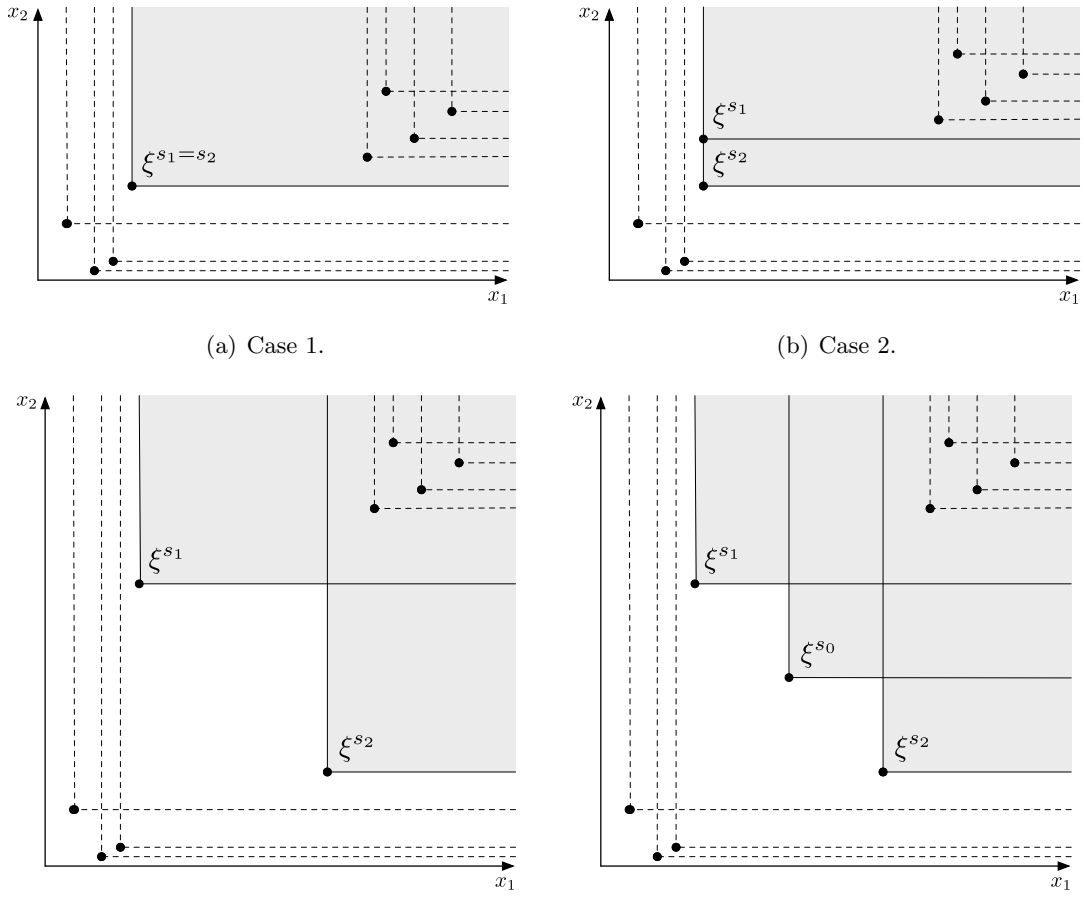
**Lemma 5.4.** *Let $\xi$ be uniformly distributed in $\prod_{j=1}^d \{1, \ldots, M_j\}$ for $M \in \mathbb{Z}^d$. Then $x \in Q_x$ is equivalent to the nonlinear integer programming problem given by*

$$x \geq y \tag{138a}$$

$$\prod_{j=1}^d y_j \geq (1-\delta) \prod_{j=1}^d M_j \tag{138b}$$

$$M \geq y \geq 0 \tag{138c}$$

$$y \in \mathbb{Z}^d \tag{138d}$$




(a) Case 1.

(b) Case 2.

(c) Case 3.

(d) Case 4.

**Figure 20:** Cases of Proposition 5.3.

*Proof.* $\xi$ uniformly distributed in $\prod_{j=1}^{d}\{1,\ldots,M_j\}$ is equivalent to $\xi = \lceil \eta \rceil$ for $\eta$ uniformly distributed in $\prod_{j=1}^{d}[0, M_j]$ with the roundup operation taken componentwise. The result then follows from

$$\mathbb{P}_{\xi}(x \geq \xi) = \mathbb{P}_{\eta}(x \geq \lceil \eta \rceil) = \mathbb{P}_{\eta}\left(\exists y \in \mathbb{Z}^d \text{ s.t. } x \geq y \geq \eta\right).$$

□

(138) is a nonlinear formulation of $Q_x$ for the specific form of $\xi$ considered. It is straightforward that this formulation is nearly sharp in the following sense.

**Lemma 5.5.** *Let $M \in \mathbb{Z}^d$ and let $W_x := \{x \in \mathbb{R}^d : \prod_{j\in J} x_j \geq (1-\delta)\prod_{j\in J} M_j \, \forall J \subset \{1,\ldots,d\}\}$ be the projection onto the $x$ variables of the continuous relaxation of* (138) *given*

*by* (138a)–(138c). *Then* $W_x \subset Q_x + B_\infty$ *for* $\xi$ *uniformly distributed in* $\prod_{j=1}^{d}\{1,\ldots,M_j\}$ *and* $B_\infty := \{x \in \mathbb{R}^d : ||x||_\infty \leq 1\}$.

Figure 21 illustrates this problem for $d = 2$, $M_1 = 13$, $M_2 = 9$ and $\delta = 0.4$. Points $\{\xi^s\}_{s=1}^{S}$ are depicted by the blue dots, $Q_x$ is the region above the thick line and $W_x$ is given by the shaded region.

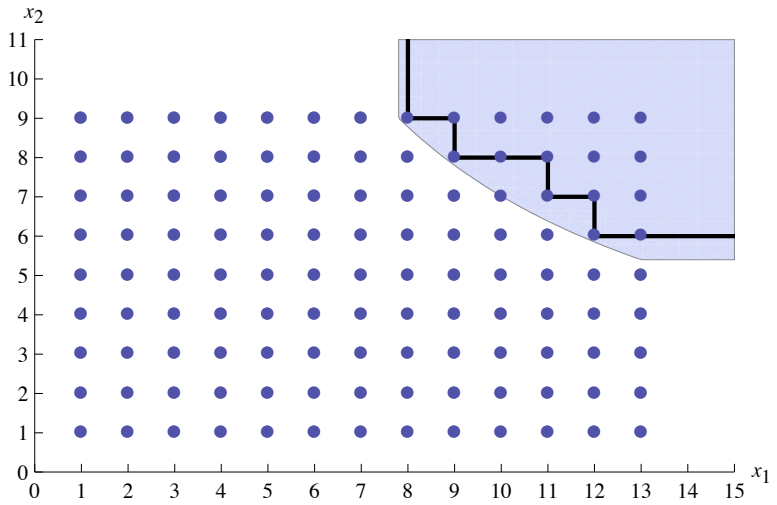



**Figure 21:** Integers in box for $d = 2$, $M_1 = 13$, $M_2 = 9$ and $\delta = 0.4$.

We conjecture that for $\xi$ uniformly distributed in $\prod_{j=1}^{d}\{1,\ldots,M_j\}$ with $M \in \mathbb{Z}^d$ any formulation of $Q_x$, such as (106), whose LP relaxation projected onto the $x$ variables is contained in $H_x$ is stronger than (138).

**Conjecture 5.6.** $H_x \subset W_x$ *for* $\xi$ *uniformly distributed in* $\prod_{j=1}^{d}\{1,\ldots,M_j\}$ *with* $M \in \mathbb{Z}^d$.

Lemma 5.4 and Conjecture 5.6 would also show that any MILP formulation of $Q_x$ for $\xi$ uniformly distributed in $\prod_{j=1}^{d}\{1,\ldots,M_j\}$ with the strength of (106) is a strong MILP formulation for the nonlinear integer programming problem (138).

### *5.5 Computational Results*

We now present a computational study of the strength of the 1-row relaxation $H_{x,z}$. To evaluate the potential advantage of valid inequalities that consider more than one row of (98a) at a time we also study the strength of the 2-row relaxation for even $d$ given by $H^2_{x,z} := H_{x,z}\left(\{D_l\}_{l=1}^{L}\right)$ for $l = d/2$ and $D_l := \{2l-1, 2l\}$.

To test the strength of the 1-row relaxation we use the LP relaxation of formulation (106) and to test the 2-row relaxation we use the LP relaxation of formulation (114) for $l = d/2$ and $D_l := \{2l-1, 2l\}$. All formulations are generated using Ilog Concert and solved using CPLEX 11 in a 2.4GHz Xeon workstation with 2GB of RAM.

We test the fomulations using $\{\xi^s\}_{s=1}^S$ obtained as independent samples from the following distributions:

**Box** Uniform in $[1, 50]^d$.

**Circle** Uniform in $\{x \in \mathbb{R}^d_+ : ||x||_2 = 50\}$.

**Box-Box** Uniform in $[0, 1]^d$ with probability 0.88 and uniform in $[1, 2]^d$ with probability 0.12.

**Box-Circle** Uniform in $[0, 1]^d$ with probability 0.88 and uniform in $\{x \in \mathbb{R}^2_+ : ||x - \mathbf{1}||_2 = 50,\ x \geq \mathbf{1}\}$ with probability 0.12.

**Multi-Log** An equally weighted mixture of Gaussians with 18 modes. Each center is obtained from two independent samples from a lognormal distribution.

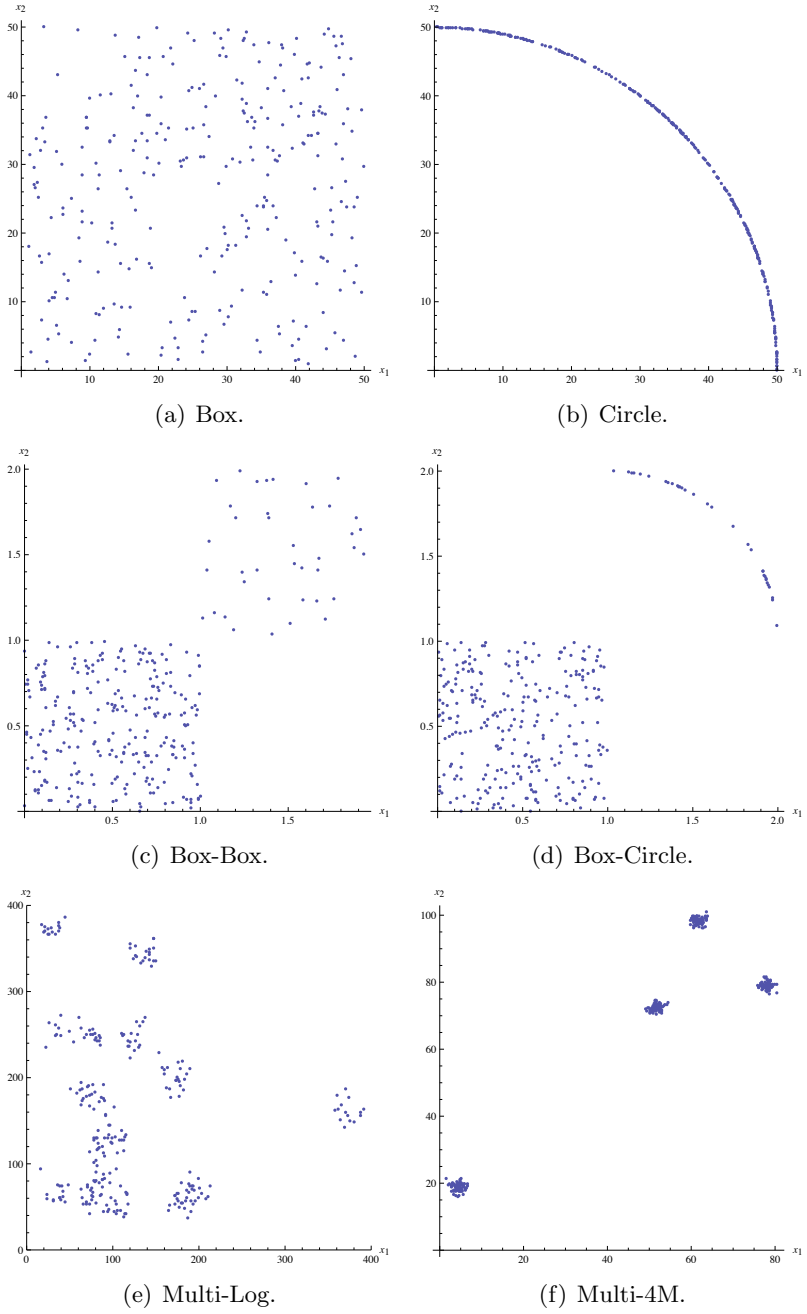
**Multi-R4** A equally weighted mixture of Gaussians with 4 modes. Each center is selected uniform in $[1, 100] \times [1, 100]$.

Examples of the points obtained from each distribution for $d = 2$ are depicted in Figure 22.

### 5.5.1 Sharpness Tests $d = 2$

To study the sharpness of 1-row formulation $H_{x,z}$ for $d = 2$ we compare the values $v_Q := \min\{x_1 + x_2 : x \in Q_x\}$ and $v_H := \min\{x_1 + x_2 : x \in H_x\}$. We calculate $v_Q$ by solving the MILP given by $\min_{x,z,y,w}\{x_1 + x_2 : (106a)–(106k)\}$ and we calculate $v_H$ by solving the LP given by $\min_{x,z,y,w}\{x_1 + x_2 : (106a)–(106i)\}$.

Because we are interested in the strength of 1-row formulation $H_{x,z}$ beyond marginal relaxation $M_x$ we first compare $v_Q$ against $v_M := \min\{x_1+x_2 : x \in M_x\}$, which we calculate



(a) Box. (b) Circle.

(c) Box-Box. (d) Box-Circle.

(e) Multi-Log. (f) Multi-4M.

**Figure 22:** Example distribution realizations for $d = 1$.

directly. Table 24 shows statistics for the marginal GAP given by $100 * (v_Q - v_M)/(v_Q)$ for 100 instances for each of the distributions considered. We use sample sizes $S$ of 100 and 300 and probabilities $\delta = 0.05$, $\delta = 0.10$, $\delta = 0.15$ and $\delta = 0.20$.

We see that the GAPs tend to increase both with $S$ and $\delta$. However, this increment is moderate and in some cases, such as Box-Box and Box-Circle, an increment in $\delta$ can result

**Table 24:** Marginal GAP for $d = 2$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.34 | 1.52 | 2.81 | 4.69 | 1.13 | 2.50 | 4.13 | 5.78 |
| | avg | **1.66** | **3.84** | **6.19** | **8.50** | **1.82** | **4.10** | **6.58** | **9.18** |
| | max | 3.79 | 8.63 | 11.19 | 12.57 | 2.71 | 6.04 | 9.00 | 11.98 |
| | std | 0.01 | 0.09 | 0.05 | 0.14 | 0.02 | 0.05 | 0.13 | 0.04 |
| Circle | min | 0.02 | 0.15 | 0.43 | 0.94 | 0.04 | 0.26 | 0.75 | 1.44 |
| | avg | **0.13** | **0.51** | **1.14** | **2.13** | **0.13** | **0.55** | **1.20** | **2.20** |
| | max | 0.47 | 1.12 | 2.87 | 3.60 | 0.30 | 1.06 | 1.95 | 3.22 |
| | std | 0.01 | 0.02 | 0.05 | 0.09 | 0.01 | 0.00 | 0.01 | 0.03 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 4.18 | 0.00 | 0.00 | 1.19 |
| | avg | **6.48** | **5.66** | **2.04** | **3.69** | **8.16** | **9.68** | **1.77** | **3.93** |
| | max | 15.85 | 22.19 | 12.65 | 8.86 | 15.88 | 23.78 | 18.71 | 6.64 |
| | std | 0.12 | 0.36 | 0.16 | 0.14 | 0.10 | 0.59 | 0.11 | 0.01 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.75 | 0.00 | 0.00 | 0.73 |
| | avg | **5.00** | **11.76** | **4.48** | **3.58** | **4.71** | **15.74** | **3.90** | **3.59** |
| | max | 22.50 | 32.64 | 29.04 | 26.70 | 9.93 | 31.54 | 33.04 | 6.28 |
| | std | 0.43 | 1.14 | 0.25 | 0.12 | 0.24 | 0.64 | 0.24 | 0.10 |
| Multi-Log | min | 0.09 | 0.94 | 1.26 | 2.51 | 0.00 | 1.47 | 2.27 | 4.32 |
| | avg | **4.36** | **12.28** | **16.82** | **18.94** | **3.67** | **12.28** | **18.25** | **20.41** |
| | max | 32.39 | 48.25 | 55.00 | 49.30 | 35.26 | 47.70 | 52.16 | 50.96 |
| | std | 0.39 | 0.91 | 1.12 | 1.12 | 0.28 | 0.24 | 0.45 | 0.44 |
| Multi-4M | min | 0.08 | 0.18 | 0.12 | 0.00 | 0.18 | 0.27 | 0.35 | 0.53 |
| | avg | **0.38** | **0.53** | **0.72** | **1.09** | **0.43** | **0.62** | **0.82** | **1.16** |
| | max | 1.17 | 1.18 | 1.36 | 3.06 | 0.74 | 1.19 | 1.60 | 2.24 |
| | std | 0.00 | 0.00 | 0.01 | 0.01 | 0.01 | 0.01 | 0.00 | 0.03 |

in a dramatic decrement of the GAP. Furthermore, the marginal GAP seems to be highly dependent on the distribution used.

Table 25 shows the same statistics for the 1-row GAP given by $100 * (v_Q - v_H)/(v_Q)$. We see that the 1-row GAP provides a significant improvement over the marginal GAP, as the former are quite small even for the cases in which the marginal GAP was large. However, for Box-Circle and Multi-Log we can still find instances with a large 1-row GAP.

We do not include results for the 2-row relaxations as these would always provide a GAP of zero.

**Table 25:** 1-row GAP for $d = 2$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.01** | **0.01** | **0.03** | **0.00** | **0.01** | **0.02** | **0.02** |
| | max | 0.19 | 0.39 | 0.23 | 0.63 | 0.04 | 0.11 | 0.45 | 0.60 |
| | std | 0.02 | 0.06 | 0.05 | 0.10 | 0.01 | 0.02 | 0.06 | 0.07 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| | max | 0.00 | 0.03 | 0.01 | 0.02 | 0.01 | 0.01 | 0.00 | 0.00 |
| | std | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.05** | **0.06** | **0.00** | **0.02** | **0.05** | **0.12** | **0.02** | **0.01** |
| | max | 1.29 | 2.56 | 0.30 | 0.53 | 0.74 | 2.01 | 1.78 | 0.13 |
| | std | 0.22 | 0.31 | 0.03 | 0.08 | 0.14 | 0.36 | 0.18 | 0.03 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.01** | **1.29** | **0.64** | **0.22** | **0.00** | **2.26** | **1.06** | **0.01** |
| | max | 0.56 | 11.15 | 11.66 | 13.92 | 0.00 | 14.65 | 20.56 | 0.13 |
| | std | 0.06 | 2.76 | 2.30 | 1.57 | 0.00 | 3.72 | 3.71 | 0.02 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.11** | **0.27** | **0.60** | **0.48** | **0.02** | **0.83** | **1.09** | **1.01** |
| | max | 9.55 | 14.48 | 24.76 | 14.50 | 1.16 | 16.91 | 32.81 | 22.66 |
| | std | 0.97 | 1.56 | 2.75 | 2.00 | 0.13 | 2.60 | 4.09 | 3.45 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| | max | 0.05 | 0.06 | 0.02 | 0.16 | 0.01 | 0.01 | 0.02 | 0.06 |
| | std | 0.01 | 0.01 | 0.00 | 0.02 | 0.00 | 0.00 | 0.00 | 0.01 |

### 5.5.2 Interaction with Other Constraints

To study the interaction of the 1-row and 2-row relaxations with other constraints we now study the probabilistically constrained transportation problem given by

$$v_{t,Q} := \min \quad \sum_{i=1}^{n} \sum_{j=1}^{d} c_{i,j} f_{i,j} \tag{139a}$$

*s.t.*

$$\sum_{j=1}^{d} f_{i,j} \leq b_i \tag{139b}$$

$$\sum_{i=1}^{n} f_{i,j} = x_j \tag{139c}$$

$$f_{i,j} \geq 0 \qquad \forall i \in \{1, \ldots, d\},\, j \in \{1, \ldots, n\} \tag{139d}$$

$$\mathbb{P}\big(x \geq \xi\big) \geq 1 - \delta. \tag{139e}$$

We set $d = 2$, $n = 40$ and we randomly generate new $c_{i,j}$'s and $b_i$'s for each instance of $\{\xi^s\}_{s=1}^S$ from the previous sections. Each $c_{i,j}$ is independently generated from the uniform distribution on $[1, 100]$. To obtain $b$ we first generate each $b_i$ independently from the uniform distribution on $[1, 100]$ and then rescale $b$ with respect to $\{\xi^s\}_{s=1}^S$ to obtain a feasible transportation problem for each scenario.

For the finite distribution case we have that (139e) is equivalent to $x \in Q_x$ so problem (139) corresponds to the minimization of a convex piecewise linear function of $x$ over $Q_x$. Hence, we have that $v_{t,\text{conv}(Q_x)} := \min_{f,x}\{\sum_{i=1}^n \sum_{j=1}^d c_{i,j} f_{i,j} \ : \ x \in \text{conv}(Q_x), \text{ (139b)–(139d)}\}$ is not necessarily equal to $v_{t,Q}$. For this reason, in addition to studying the 1-row relaxation optimal value given by $v_{t,H} := \min_{f,x}\{\sum_{i=1}^n \sum_{j=1}^d c_{i,j} f_{i,j} \ : \ x \in H_x, \text{ (139b)–(139d)}\}$ we also study the 2-row relaxation optimal value given by $v_{t,H^2} := \min_{f,x}\{\sum_{i=1}^n \sum_{j=1}^d c_{i,j} f_{i,j} \ : \ x \in H_x^2, \text{ (139b)–(139d)}\}$ where $H_x^2$ is the projection onto the $x$ variables of $H_{x,z}^2$. For $d = 2$ we have that $v_{t,H^2} = v_{t,\text{conv}(Q_x)}$.

Table 26 shows the statistics for the 1-row GAP given by $100 * (v_{t,Q} - v_{t,H})/(v_{t,Q})$ and Table 27 shows the statistics for the 2-row GAP given by $100 * (v_{t,Q} - v_{t,H^2})/(v_{t,Q})$.

Comparing Table 26 and Table 25 we see that the 1-row GAPs have been at least doubled for most cases. This is expected as now having a tight formulation does not guarantee good GAPs, which is confirmed by Table 27. However, also see that using a sharp formulation of $Q_x$ does result in fairly small GAPs.

#### *5.5.2.1 Experiments for $d = 4$*

We now repeat the experiments from Sections 5.5.1 and 5.5.2 for $d = 4$. For this case all formulations were much bigger and solve times increased significantly so we only tested 10 instances for each distribution. We begin with Table 28 that shows the marginal GAPs for this case.

As expected, we can see a moderate increment of the GAPs. We note that some values, such as the maximum for Multi-Log for $S = 300$ and $\delta = 0.1$, are larger in Table 24 than Table 28 because the former considers 100 instances per distribution and the later only 10.

**Table 26:** Transportation Problems 1-row GAP for $d = 2$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.01** | **0.05** | **0.06** | **0.10** | **0.01** | **0.03** | **0.06** | **0.11** |
| | max | 0.58 | 0.86 | 1.22 | 1.28 | 0.17 | 0.39 | 1.51 | 1.40 |
| | std | 0.06 | 0.17 | 0.18 | 0.24 | 0.03 | 0.07 | 0.19 | 0.22 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| | max | 0.00 | 0.10 | 0.07 | 0.06 | 0.01 | 0.02 | 0.03 | 0.10 |
| | std | 0.00 | 0.01 | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.12** | **0.16** | **0.00** | **0.06** | **0.13** | **0.32** | **0.03** | **0.03** |
| | max | 3.20 | 4.81 | 0.00 | 1.31 | 1.81 | 7.02 | 2.41 | 0.56 |
| | std | 0.47 | 0.68 | 0.00 | 0.20 | 0.33 | 1.00 | 0.24 | 0.08 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.16** | **3.06** | **1.37** | **0.55** | **0.01** | **5.18** | **2.23** | **0.03** |
| | max | 3.30 | 22.91 | 23.92 | 29.90 | 0.31 | 30.36 | 40.98 | 1.29 |
| | std | 0.63 | 5.94 | 4.50 | 3.81 | 0.04 | 7.78 | 7.63 | 0.14 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.19** | **0.98** | **2.04** | **1.57** | **0.14** | **2.20** | **2.60** | **2.67** |
| | max | 16.27 | 37.05 | 50.92 | 29.10 | 8.79 | 39.80 | 62.34 | 30.10 |
| | std | 1.64 | 4.37 | 6.16 | 4.66 | 1.00 | 6.11 | 7.76 | 6.70 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.01** | **0.00** | **0.00** | **0.00** | **0.01** |
| | max | 0.10 | 0.22 | 0.12 | 0.36 | 0.00 | 0.07 | 0.07 | 0.15 |
| | std | 0.01 | 0.02 | 0.01 | 0.04 | 0.00 | 0.01 | 0.01 | 0.02 |

Tables 29 and (30) present the 1-row and 2-row GAPs. We note that because $d > 2$ we can now have $\text{conv}(Q_x) \subsetneq H_x^2$ and hence $v_{H^2} < v_{\text{conv}(Q_x)}$. We also note that in many instances of Box, Circle, Multi-Log and Multi-4M with $S = 300$ the 2-row relaxations did not reach optimality after $10,000$ seconds of dual simplex iterations so the results on Table 30 are only upper bounds on the GAPs for these cases. This explains why the 2-row GAPs are sometimes larger than the 1-row GAPs.

We can see that the 1-row GAPs have increased, but with few exceptions remain fairly small. However, because Table 25 considers 100 instances per distribution and Table 29 only considers 10, it is again difficult to compare them specially for distributions that yield high GAP variability such as Box-Circle and Multi-Log. With respect to the 2-row GAPs

**Table 27:** Transportation Problem 2-row GAP for $d = 2$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.01** | **0.02** | **0.00** | **0.01** | **0.01** | **0.02** |
| | max | 0.23 | 0.00 | 0.30 | 0.86 | 0.00 | 0.39 | 0.39 | 0.71 |
| | std | 0.02 | 0.00 | 0.04 | 0.09 | 0.00 | 0.04 | 0.05 | 0.09 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| | max | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 | 0.02 | 0.03 | 0.10 |
| | std | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.02** | **0.02** | **0.00** | **0.01** | **0.02** | **0.03** | **0.00** | **0.01** |
| | max | 0.66 | 1.37 | 0.00 | 0.46 | 0.49 | 2.03 | 0.05 | 0.50 |
| | std | 0.08 | 0.14 | 0.00 | 0.06 | 0.07 | 0.22 | 0.01 | 0.06 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.05** | **0.21** | **0.14** | **0.06** | **0.01** | **0.47** | **0.17** | **0.01** |
| | max | 2.58 | 4.85 | 5.64 | 2.74 | 0.31 | 6.36 | 3.71 | 0.86 |
| | std | 0.32 | 0.65 | 0.65 | 0.34 | 0.04 | 1.08 | 0.66 | 0.09 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.02** | **0.13** | **0.37** | **0.49** | **0.10** | **0.10** | **0.39** | **0.49** |
| | max | 2.37 | 9.04 | 13.27 | 13.77 | 5.91 | 6.10 | 14.03 | 13.71 |
| | std | 0.24 | 0.94 | 1.63 | 1.94 | 0.72 | 0.66 | 1.76 | 1.79 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| | max | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| | std | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

we can see that they are in fact greater than zero, but can be significantly smaller than the 1-row GAPs.

Our final set of tables considers transportation problem (139) for $d = 4$. Table 31 presents the 1-row GAPs and Table 32 presents the 2-row GAPs. We note that again in many instances of Box, Circle, Multi-Log and Multi-4M with $S = 300$ the 2-row relaxations did not reach optimality after $10,000$ seconds of dual simplex iterations so the results on Table 32 are only upper bounds on the GAPs for these cases.

We can see that 1-row gaps can be quite large, but that the GAPs are significantly reduced when using a 2-row relaxation.

**Table 28:** Marginal GAP for $d = 4$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 1.67 | 5.01 | 7.82 | 10.92 | 2.56 | 5.62 | 8.99 | 12.13 |
| | avg | **3.03** | **6.20** | **9.64** | **12.77** | **2.96** | **6.43** | **10.01** | **13.66** |
| | max | 4.82 | 7.67 | 12.47 | 15.75 | 3.35 | 7.78 | 10.98 | 15.49 |
| | std | 0.45 | 0.12 | 0.44 | 0.25 | 0.10 | 0.21 | 0.11 | 0.23 |
| Circle | min | 4.44 | 8.99 | 14.69 | 19.33 | 6.03 | 10.41 | 15.65 | 20.00 |
| | avg | **5.48** | **10.80** | **15.85** | **20.89** | **6.59** | **11.83** | **16.70** | **21.05** |
| | max | 6.64 | 12.51 | 17.11 | 23.02 | 7.06 | 13.19 | 17.63 | 22.42 |
| | std | 0.05 | 0.57 | 0.01 | 0.28 | 0.05 | 0.45 | 0.21 | 0.00 |
| Box-Box | min | 5.57 | 0.00 | 0.67 | 2.10 | 10.14 | 0.59 | 1.12 | 4.53 |
| | avg | **9.67** | **13.35** | **6.85** | **4.87** | **13.15** | **19.22** | **2.26** | **6.16** |
| | max | 18.29 | 25.07 | 21.26 | 8.44 | 17.74 | 24.61 | 4.28 | 9.23 |
| | std | 0.21 | 2.82 | 2.06 | 0.33 | 0.04 | 1.33 | 0.05 | 0.04 |
| Box-Circle | min | 8.19 | 1.04 | 0.00 | 3.26 | 14.04 | 0.60 | 0.43 | 4.59 |
| | avg | **13.39** | **17.36** | **1.92** | **5.43** | **16.58** | **23.57** | **2.35** | **6.50** |
| | max | 20.18 | 24.24 | 5.85 | 9.98 | 20.72 | 28.03 | 4.76 | 9.68 |
| | std | 0.42 | 1.98 | 0.39 | 0.28 | 0.03 | 0.84 | 0.24 | 0.20 |
| Multi-Log | min | 1.15 | 7.32 | 12.14 | 22.73 | 1.48 | 8.93 | 14.20 | 16.96 |
| | avg | **10.80** | **15.36** | **27.60** | **33.20** | **6.01** | **18.08** | **28.56** | **33.63** |
| | max | 28.93 | 29.86 | 51.41 | 52.65 | 19.98 | 26.84 | 52.52 | 44.49 |
| | std | 2.17 | 2.94 | 0.89 | 0.26 | 0.62 | 0.47 | 3.26 | 3.45 |
| Multi-4M | min | 0.22 | 0.55 | 0.74 | 1.10 | 0.58 | 0.82 | 1.03 | 1.35 |
| | avg | **0.56** | **0.84** | **1.17** | **1.75** | **0.75** | **1.02** | **1.28** | **1.75** |
| | max | 0.85 | 1.18 | 1.41 | 2.79 | 0.95 | 1.22 | 1.54 | 2.15 |
| | std | 0.02 | 0.07 | 0.06 | 0.17 | 0.06 | 0.04 | 0.05 | 0.07 |

### *5.6 Conclusions*

Although we showed that the 1-row relaxations can be as bad as the trivial marginal relaxation in the worse case, we also saw some theoretical and computational evidence that it might be a very strong relaxation for many cases. In fact, the 1-row relaxation was weak only on instance classes that were purposely constructed to be problematic, and even then, the 1-row relaxation managed to be strong in many of these instances.

Using a new multi-row extended formulation we also compared the strength of the 1-row and 2-row relaxations computationally. We saw that in the few instances in which the 1-row relaxation was weak the 2-row relaxation could close a large portion of the GAPs showing that 2-row valid inequalities could provide an advantage in some cases.

**Table 29:** 1-row GAP for $d = 4$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.06 | 0.05 |
| | avg | **0.03** | **0.02** | **0.10** | **0.27** | **0.01** | **0.06** | **0.25** | **0.47** |
| | max | 0.30 | 0.10 | 0.34 | 0.76 | 0.08 | 0.27 | 0.58 | 1.39 |
| | std | 0.09 | 0.03 | 0.12 | 0.30 | 0.02 | 0.08 | 0.14 | 0.46 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.03** | **0.05** | **0.03** | **0.03** | **0.03** | **0.05** | **0.04** |
| | max | 0.03 | 0.25 | 0.19 | 0.10 | 0.11 | 0.09 | 0.14 | 0.11 |
| | std | 0.01 | 0.08 | 0.07 | 0.04 | 0.04 | 0.03 | 0.05 | 0.04 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.54 | 0.00 | 0.00 | 0.00 |
| | avg | **0.41** | **1.69** | **0.27** | **0.00** | **1.63** | **3.10** | **0.01** | **0.08** |
| | max | 2.72 | 5.22 | 2.20 | 0.03 | 3.52 | 5.62 | 0.03 | 0.31 |
| | std | 0.87 | 2.07 | 0.68 | 0.01 | 0.92 | 1.80 | 0.01 | 0.09 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.26** | **2.35** | **0.00** | **0.02** | **0.82** | **6.98** | **0.00** | **0.07** |
| | max | 1.01 | 5.87 | 0.00 | 0.09 | 3.07 | 9.56 | 0.03 | 0.31 |
| | std | 0.38 | 1.98 | 0.00 | 0.04 | 0.93 | 2.62 | 0.01 | 0.10 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.17** | **0.03** | **0.30** | **1.86** | **0.00** | **0.53** | **1.02** | **0.60** |
| | max | 1.73 | 0.30 | 1.99 | 7.69 | 0.02 | 2.61 | 3.73 | 3.47 |
| | std | 0.55 | 0.10 | 0.63 | 2.61 | 0.01 | 0.91 | 1.62 | 1.27 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.01** | **0.00** | **0.01** | **0.01** | **0.02** |
| | max | 0.03 | 0.02 | 0.02 | 0.06 | 0.02 | 0.02 | 0.05 | 0.08 |
| | std | 0.01 | 0.01 | 0.01 | 0.02 | 0.01 | 0.01 | 0.01 | 0.03 |

**Table 30:** 2-row GAP for $d = 4$ [%].

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.03 | 0.02 |
| | avg | **0.02** | **0.01** | **0.03** | **0.09** | **0.01** | **0.04** | **0.12** | **0.25** |
| | max | 0.15 | 0.03 | 0.29 | 0.34 | 0.06 | 0.11 | 0.25 | 0.79 |
| | std | 0.05 | 0.01 | 0.09 | 0.14 | 0.02 | 0.05 | 0.06 | 0.26 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.01** | **0.04** | **0.02** | **0.02** | **0.02** | **0.04** | **0.05** |
| | max | 0.03 | 0.08 | 0.17 | 0.09 | 0.09 | 0.08 | 0.12 | 0.11 |
| | std | 0.01 | 0.03 | 0.06 | 0.03 | 0.03 | 0.03 | 0.04 | 0.04 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.04** | **0.37** | **0.03** | **0.00** | **0.17** | **0.34** | **0.00** | **0.04** |
| | max | 0.23 | 1.96 | 0.26 | 0.00 | 0.57 | 1.15 | 0.03 | 0.11 |
| | std | 0.08 | 0.67 | 0.08 | 0.00 | 0.19 | 0.45 | 0.01 | 0.04 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.01** | **0.42** | **0.00** | **0.01** | **0.09** | **1.16** | **0.00** | **0.04** |
| | max | 0.11 | 1.77 | 0.00 | 0.06 | 0.39 | 2.69 | 0.03 | 0.17 |
| | std | 0.04 | 0.60 | 0.00 | 0.02 | 0.15 | 1.12 | 0.01 | 0.06 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.03** | **0.11** | **1.77** | **0.00** | **0.44** | **0.91** | **0.64** |
| | max | 0.00 | 0.30 | 1.06 | 7.45 | 0.00 | 2.45 | 3.73 | 3.53 |
| | std | 0.00 | 0.10 | 0.34 | 2.56 | 0.00 | 0.84 | 1.51 | 1.30 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.00** | **0.00** | **0.01** | **0.00** | **0.00** | **0.00** | **0.01** |
| | max | 0.00 | 0.00 | 0.01 | 0.05 | 0.02 | 0.01 | 0.01 | 0.04 |
| | std | 0.00 | 0.00 | 0.00 | 0.02 | 0.01 | 0.00 | 0.00 | 0.02 |

**Table 31:** Transportation Problem 1-row GAP for $d = 4$ [%]

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.10 |
| | avg | **0.09** | **0.06** | **0.25** | **0.62** | **0.04** | **0.14** | **0.64** | **1.54** |
| | max | 0.71 | 0.30 | 0.83 | 2.73 | 0.25 | 0.40 | 2.21 | 3.88 |
| | std | 0.23 | 0.11 | 0.31 | 0.84 | 0.08 | 0.15 | 0.62 | 1.31 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.09** | **0.29** | **0.17** | **0.06** | **0.05** | **0.12** | **0.06** |
| | max | 0.00 | 0.44 | 1.01 | 0.55 | 0.16 | 0.25 | 0.31 | 0.16 |
| | std | 0.00 | 0.17 | 0.30 | 0.21 | 0.05 | 0.09 | 0.11 | 0.07 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.37 | 0.00 | 0.00 | 0.00 |
| | avg | **1.28** | **4.13** | **0.91** | **0.03** | **4.22** | **6.78** | **0.04** | **0.10** |
| | max | 7.13 | 13.11 | 4.82 | 0.17 | 7.97 | 13.51 | 0.14 | 0.42 |
| | std | 2.30 | 5.14 | 1.55 | 0.05 | 2.23 | 4.18 | 0.05 | 0.13 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.52 | 0.00 | 0.00 | 0.00 |
| | avg | **0.79** | **5.18** | **0.00** | **0.03** | **1.96** | **14.58** | **0.00** | **0.06** |
| | max | 4.42 | 9.55 | 0.01 | 0.15 | 7.59 | 21.11 | 0.01 | 0.20 |
| | std | 1.40 | 3.78 | 0.00 | 0.06 | 2.06 | 5.69 | 0.00 | 0.08 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.67** | **0.36** | **3.69** | **3.53** | **0.00** | **1.80** | **3.77** | **1.91** |
| | max | 3.83 | 2.53 | 19.46 | 15.54 | 0.05 | 6.55 | 19.94 | 9.70 |
| | std | 1.26 | 0.79 | 6.49 | 4.96 | 0.02 | 2.47 | 6.67 | 3.89 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.01** | **0.01** | **0.04** | **0.01** | **0.03** | **0.02** | **0.05** |
| | max | 0.02 | 0.10 | 0.04 | 0.16 | 0.08 | 0.16 | 0.13 | 0.14 |
| | std | 0.01 | 0.03 | 0.01 | 0.06 | 0.02 | 0.05 | 0.04 | 0.05 |

**Table 32:** Transportation Problem 2-row GAP for $d = 4$ [%]

| | | 100 | | | | 300 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.05 | 0.10 | 0.15 | 0.20 | 0.05 | 0.10 | 0.15 | 0.20 |
| Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.05** | **0.03** | **0.05** | **0.19** | **0.03** | **0.06** | **0.29** | **0.92** |
| | max | 0.30 | 0.26 | 0.25 | 0.91 | 0.22 | 0.28 | 1.25 | 2.32 |
| | std | 0.10 | 0.08 | 0.08 | 0.34 | 0.07 | 0.11 | 0.36 | 0.78 |
| Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.04** | **0.24** | **0.12** | **0.04** | **0.03** | **0.10** | **0.07** |
| | max | 0.00 | 0.32 | 0.94 | 0.55 | 0.09 | 0.25 | 0.23 | 0.18 |
| | std | 0.00 | 0.10 | 0.28 | 0.18 | 0.04 | 0.08 | 0.09 | 0.07 |
| Box-Box | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.08** | **1.07** | **0.22** | **0.00** | **0.74** | **0.78** | **0.03** | **0.05** |
| | max | 0.70 | 6.65 | 1.43 | 0.01 | 2.86 | 3.42 | 0.14 | 0.21 |
| | std | 0.22 | 2.28 | 0.47 | 0.00 | 0.81 | 1.15 | 0.05 | 0.08 |
| Box-Circle | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.26** | **1.45** | **0.00** | **0.02** | **0.49** | **3.93** | **0.00** | **0.03** |
| | max | 2.24 | 3.40 | 0.01 | 0.12 | 3.12 | 7.83 | 0.00 | 0.11 |
| | std | 0.71 | 1.42 | 0.00 | 0.05 | 0.97 | 2.45 | 0.00 | 0.04 |
| Multi-Log | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.29** | **0.34** | **2.45** | **3.18** | **0.00** | **1.51** | **3.70** | **1.91** |
| | max | 1.45 | 2.53 | 19.44 | 15.16 | 0.04 | 5.90 | 19.80 | 9.40 |
| | std | 0.61 | 0.80 | 6.08 | 4.85 | 0.01 | 2.15 | 6.64 | 3.94 |
| Multi-4M | min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | avg | **0.00** | **0.01** | **0.00** | **0.03** | **0.01** | **0.01** | **0.00** | **0.04** |
| | max | 0.00 | 0.06 | 0.02 | 0.16 | 0.07 | 0.05 | 0.02 | 0.20 |
| | std | 0.00 | 0.02 | 0.01 | 0.06 | 0.02 | 0.02 | 0.01 | 0.06 |

# REFERENCES


[1] Abhishek, K., Leyffer, S., and Linderoth, J. T., "Filmint: An outer-approximation-based solver for nonlinear mixed integer programs," Preprint ANL/MCS-P1374-0906, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, September 2006.

[2] Aichholzer, O., Aurenhammer, F., Hurtado, F., and Krasser, H., "Towards compatible triangulations," *Theoretical Computer Science*, vol. 296, pp. 3–13, 2003.

[3] Akturk, M. S., Atamturk, A., and Gurel, S., "A strong conic quadratic reformulation for machine-job assignment with controllable processing times," *Operations Research Letters (To appear)*, 2009. doi:10.1016/j.orl.2008.12.009.

[4] Appleget, J. A. and Wood, R. K., "Explicit-constraint branching for solving mixed-integer programs," in *Computing tools for modeling, optimization, and simulation: interfaces in computer science and operations research* (Laguna, M. and González , J. L., eds.), vol. 12 of *Operations research / computer science interfaces series*, pp. 245–261, Kluwer, 2000.

[5] Balakrishnan, A. and Graves, S. C., "A composite algorithm for a concave-cost network flow problem," *Networks*, vol. 19, pp. 175–202, 1989.

[6] Balas, E., "Disjunctive programming," *Annals of Discrete Mathematics*, vol. 5, pp. 3–51, 1979.

[7] Balas, E., "Disjunctive programming and a hierarchy of relaxations for discrete optimization problems," *SIAM Journal on Algebraic and Discrete Methods*, vol. 6, pp. 466–486, 1985.

[8] Balas, E., "On the convex-hull of the union of certain polyhedra," *Operations Research Letters*, vol. 7, pp. 279–283, 1988.

[9] Balas, E., "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Applied Mathematics*, vol. 89, pp. 3–44, 1998.

[10] Balas, E., "Projection, lifting and extended formulation in integer and combinatorial optimization," *Annals of Operations Research*, vol. 140, pp. 125–161, 2005.

[11] Ball, K. M., "An elementary introduction to modern convex geometry," in *Flavors of Geometry* (Levy, S., ed.), vol. 31 of *Mathematical Sciences Research Institute Publications*, pp. 1–58, Cambridge: Cambridge University Press, 1997.

[12] Beale, E. M. L. and Tomlin, J. A., "Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables," in *OR 69: Proceedings of the fifth international conference on operational research* (Lawrence, J., ed.), pp. 447–454, Tavistock Publications, 1970.

[13] Ben-Tal, A. and Nemirovski, A., "Robust solutions of uncertain linear programs," *Operations Research Letters*, vol. 25, pp. 1–13, 1999.

[14] Ben-Tal, A. and Nemirovski, A., *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2001.

[15] Ben-Tal, A. and Nemirovski, A., "On polyhedral approximations of the second-order cone," *Mathematics of Operations Research*, vol. 26, pp. 193–205, 2001.

[16] Beraldi, P. and Ruszczynski, A., "A Branch and Bound Method for Stochastic Integer Problems Under Probabilistic Constraints," *Optimization Methods & Software*, vol. 17, pp. 359–382, 2002.

[17] Beraldi, P. and Ruszczynski, A., "The probabilistic set-covering problem," *Operations Research*, vol. 50, pp. 956–967, 2002.

[18] Bergamini, M. L., Aguirre, P., and Grossmann, I., "Logic-based outer approximation for globally optimal synthesis of process networks," *Computers & Chemical Engineering*, vol. 29, pp. 1914–1933, 2005.

[19] Bergamini, M. L., Grossmann, I., Scenna, N., and Aguirre, P., "An improved piecewise outer-approximation algorithm for the global optimization of minlp models involving concave and bilinear terms," *Computers & Chemical Engineering*, vol. 32, pp. 477–493, 2008.

[20] Bertsimas, D., Darnell, C., and Soucy, R., "Portfolio construction through mixed-integer programming at grantham, mayo, van otterloo and company," *Interfaces*, vol. 29, pp. 49–66, 1999.

[21] Bertsimas, D. and Shioda, R., "Algorithm for cardinality-constrained quadratic optimization," *Computational Optimization and Applications (To appear)*, 2007. doi:10.1007/s10589-007-9126-9.

[22] Bertsimas, D. and Weismantel, R., *Optimization Over Integers*. Dynamic Ideas, 2005.

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

[24] Bixby, R. and Rothberg, E., "Progress in computational mixed integer programming - a look back from the other side of the tipping point," *Annals of Operations Research*, vol. 149, pp. 37–41, 2007.

[25] Bixby, R. E., Fenelon, M., Gu, Z., Rothberg, E., and Wunderling, R., "Mip: Theory and practice - closing the gap," in *System Modelling and Optimization* (Powell, M. J. D. and Scholtes, S., eds.), vol. 174 of *IFIP Conference Proceedings*, pp. 19–50, Kluwer, 1999.

[26] Blair, C., "2 rules for deducing valid inequalities for 0-1 problems," *SIAM Journal on Applied Mathematics*, vol. 31, pp. 614–617, 1976.

[27] Blair, C., "Representation for multiple right-hand sides," *Mathematical Programming*, vol. 49, pp. 1–5, 1990.

[28] Bonami, P., Biegler, L. T., Conn, A. R., Cornuejols, G., Grossmann, I. E., Laird, C. D., Lee, J., Lodi, A., Margot, F., Sawaya, N., and Waechter, A., "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optimization*, vol. 5, pp. 186–204, 2007.

[29] Borchers, B. and Mitchell, J. E., "An improved branch and bound algorithm for mixed integer nonlinear programs," *Computers and Operations Research*, vol. 21, pp. 359–367, 1994.

[30] Carnicer, J. M. and Floater, M. S., "Piecewise linear interpolants to lagrange and hermite convex scattered data," *Numerical Algorithms*, vol. 13, pp. 345–364, 1996.

[31] Ceria, S. and Stubbs, R. A., "Incorporating estimation errors into portfolio selection: Robust portfolio construction," *Journal of Asset Management*, vol. 7, pp. 109–127, 2006.

[32] Chang, T.-J., Meade, N., Beasley, J. E., and Sharaiha, Y. M., "Heuristics for cardinality constrained portfolio optimisation," *Computers & Operations Research*, vol. 27, pp. 1271–1302, 2000.

[33] Christof, T. and Loebel, A., "PORTA – POlyhedron Representation Transformation Algorithm, version 1.3." Available at `http://www.iwr.uni-heidelberg.de/groups/comopt/software/PORTA/` (Date accessed: June/2009).

[34] Conforti, M. and Wolsey, L. A., "Compact formulations as a union of polyhedra," *Mathematical Programming*, vol. 114, pp. 277–289, 2008.

[35] Coppersmith, D. and Lee, J., "Parsimonious binary-encoding in integer programming," *Discrete Optimization*, vol. 2, pp. 190–200, 2005.

[36] Croxton, K. L., Gendron, B., and Magnanti, T. L., "A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems," *Management Science*, vol. 49, pp. 1268–1273, 2003.

[37] Croxton, K. L., Gendron, B., and Magnanti, T. L., "Models and methods for merge-in-transit operations," *Transportation Science*, vol. 37, pp. 1–22, 2003.

[38] Croxton, K. L., Gendron, B., and Magnanti, T. L., "Variable disaggregation in network flow problems with piecewise linear costs," *Operations Research*, vol. 55, pp. 146–157, 2007.

[39] Dantzig, G. B., "Discrete-variable extremum problems," *Operations Research*, vol. 5, pp. 266–277, 1957.

[40] Dantzig, G. B., "On the significance of solving linear-programming problems with some integer variables," *Econometrica*, vol. 28, pp. 30–44, 1960.

[41] Dantzig, G. B., *Linear Programming and Extensions.* Princeton University Press, 1963.

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

[43] de Farias Jr., I. R., Zhao, M., and Zhao, H., "A special ordered set approach for optimizing a discontinuous separable piecewise linear function," *Operations Research Letters*, vol. 36, pp. 234–238, 2008.

[44] Dentcheva, D., Lai, B., and Ruszczynski, A., "Dual methods for probabilistic optimization problems," *Mathematical Methods of Operations Research*, vol. 60, pp. 331–346, 2004.

[45] Dentcheva, D., Prekopa, A., and Ruszczynski, A., "Concavity and efficient points of discrete distributions in probabilistic programming," *Mathematical Programming*, vol. 89, pp. 55–77, 2000.

[46] Dentcheva, D., Prekopa, A., and Ruszczynski, A., "On convex probabilistic programming with discrete distributions," *Nonlinear Analysis-Theory Methods & Applications*, vol. 47, pp. 1997–2009, 2001.

[47] Dietrich, B., "Some of my favorite integer programming applications at IBM," *Annals of Operations Research*, vol. 149, pp. 75–80, 2007.

[48] Dolan, E. D. and Moré, J. J., "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, pp. 201–213, 2002.

[49] Duran, M. A. and Grossmann, I. E., "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Mathematical Programming*, vol. 36, pp. 307–339, 1986.

[50] Fletcher, R. and Leyffer, S., "Solving mixed integer nonlinear programs by outer approximation," *Mathematical Programming*, vol. 66, pp. 327–349, 1994.

[51] Fourer, R., Gay, D. M., and Kernighan, B. W., *AMPL–A Modeling Language for Mathematical Programming*. The Scientific Press, 1993.

[52] Garfinkel, R. S. and Nemhauser, G. L., *Integer Programming*. Wiley, 1972.

[53] Geoffrion, A., "Generalized benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, pp. 237–260, 1972.

[54] Glineur, F., "Computational experiments with a linear approximation of second order cone optimization," Image Technical Report 0001, Service de Mathématique et de Recherche Opérationnelle, Faculté Polytechnique de Mons, Mons, Belgium, November 2000.

[55] Graf, T., Vanhentenryck, P., Pradelleslasserre, C., and Zimmer, L., "Simulation of hybrid circuits in constraint logic programming," *Computers & Mathematics with Applications*, vol. 20, pp. 45–56, 1990.

[56] Grossmann, I. E., "Review of nonlinear mixed-integer and disjunctive programming techniques," *Optimization and Engineering*, vol. 3, pp. 227–252, 2002.

[57] Gryffenberg, I., Lausberg, J., Smith, W., Uys, S., Botha, S., Hofmeyr, F., Nicolay, R., vanderMerwe, W., and Wessels, G., "Guns or butter: Decision support for determining the size and shape of the South African National Defense Force," *Interfaces*, vol. 27, pp. 7–27, 1997.

[58] Guignard-Spielberg, M. and Spielberg, K., "Integer programming: State of the art and recent advances," *Annals of Operations Research*, vol. 139–140, 2005.

[59] Gupta, O. K. and Ravindran, A., "Branch and bound experiments in convex nonlinear integer programming," *Management Science*, vol. 31, pp. 1533–1546, 1985.

[60] Horst, R., Pardalos, P. M., and Thoai, N. V., *Introduction to Global Optimization*, vol. 3 of *Nonconvex optimization and its applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1995.

[61] Ibaraki, T., "Integer programming formulation of combinatorial optimization problems," *Discrete Mathematics*, vol. 16, pp. 39–52, 1976.

[62] ILOG, *Cplex 10: User's Manual and Reference Manual*. ILOG, S.A., 2005.

[63] Jeroslow, R. G., "Cutting plane theory: disjunctive methods," *Annals of Discrete Mathematics*, vol. 1, pp. 293–330, 1977.

[64] Jeroslow, R. G., "Representability in mixed integer programming 1: characterization results," *Discrete Applied Mathematics*, vol. 17, pp. 223–243, 1987.

[65] Jeroslow, R. G., "A simplification for some disjunctive formulations," *European Journal of Operational Research*, vol. 36, pp. 116–121, 1988.

[66] Jeroslow, R. G., "Representability of functions," *Discrete Applied Mathematics*, vol. 23, pp. 125–137, 1989.

[67] Jeroslow, R. G. and Lowe, J. K., "Modeling with integer variables," *Mathematical Programming Study*, vol. 22, pp. 167–184, 1984.

[68] Jeroslow, R. G. and Lowe, J. K., "Experimental results on the new techniques for integer programming formulations," *Journal of the Operational Research Society*, vol. 36, pp. 393–403, 1985.

[69] Johnson, E. L., Nemhauser, G. L., and Savelsbergh, M. W. P., "Progress in linear programming-based algorithms for integer programming: An exposition," *INFORMS Journal on Computing*, vol. 12, pp. 2–23, 2000.

[70] Kannan, R., "Lattice translates of a polytope and the frobenius problem," *Combinatorica*, vol. 12, pp. 161–177, 1992.

[71] Keha, A. B., *A polyhedral study of nonconvex piecewise linear optimization*. PhD thesis, Georgia Institute of Technology, 2003.

[72] Keha, A. B., de Farias, I. R., and Nemhauser, G. L., "Models for representing piecewise linear cost functions," *Operations Research Letters*, vol. 32, pp. 44–48, 2004.

[73] Keha, A. B., de Farias, I. R., and Nemhauser, G. L., "A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization," *Operations Research*, vol. 54, pp. 847–858, 2006.

[74] Kucukyavuz, S., "On mixing sets arising in probabilistic programming," *Optimization Online*, 2009. `http://www.optimization-online.org/DB_HTML/2009/03/2255.html` (Date accessed:June/2009).

[75] Lai, M. and Schumaker, L. L., *Spline functions on triangulations*, vol. 110 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2007.

[76] Land, A. and Powell, S., "A survey of the operational use of ilp models," *Annals of Operations Research*, vol. 149, pp. 147–156, 2007.

[77] Land, A. H. and Doig, A. G., "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, 1960.

[78] Lasdon, L. S. and Waren, A. D., "A survey of nonlinear programming applications," *Operations Research*, vol. 28, pp. 1029–1073, 1980.

[79] Lee, J., "All-different polytopes," *Journal of Combinatorial Optimization*, vol. 6, pp. 335–352, 2002.

[80] Lee, J., "A celebration of 50 years of integer programming," *Optima*, vol. 76, pp. 10–14, 2008.

[81] Lee, J. and Margot, F., "On a binary-encoded ilp coloring formulation," *INFORMS Journal on Computing*, vol. 19, pp. 406–415, 2007.

[82] Lee, J. and Wilson, D., "Polyhedral methods for piecewise-linear functions I: the lambda method," *Discrete Applied Mathematics*, vol. 108, pp. 269–285, 2001.

[83] Lejeune, M. A. and Ruszczynski, A., "An efficient trajectory method for probabilistic production-inventory-distribution problems," *Operations Research*, vol. 55, pp. 378–394, 2007.

[84] Leyffer, S., "Integrating SQP and branch-and-bound for mixed integer nonlinear programming," *Computational Optimization and Applications*, vol. 18, pp. 295–309, 2001.

[85] Lobo, M. S., Fazel, M., and Boyd, S., "Portfolio optimization with linear and fixed transaction costs," *Annals of Operations Research*, vol. 152, pp. 341–365, 2007.

[86] Lobo, M. S., Vandenberghe, L., and Boyd, S., "Applications of second-order cone programming," *Linear Algebra and its Applications*, vol. 284, pp. 193–228, 1998.

[87] Lowe, J. K., *Modelling with Integer Variables*. PhD thesis, Georgia Institute of Technology, 1984.

[88] Luedtke, J. and Ahmed, S., "A Sample Approximation Approach for Optimization with Probabilistic Constraints," *SIAM Journal on Optimization*, vol. 19, pp. 674–699, 2008.

[89] Luedtke, J., Ahmed, S., and Nemhauser, G., "An integer programming approach for linear programs with probabilistic constraints," in *IPCO '07: Proceedings of the 12th international conference on Integer Programming and Combinatorial Optimization*, pp. 410–423, 2007.

[90] Luedtke, J., Ahmed, S., and Nemhauser, G., "An integer programming approach for linear programs with probabilistic constraints," *Mathematical Programming (To appear)*, 2008. doi:10.1007/s10107-008-0247-4.

[91] Lulli, G. and Sen, S., "A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems," *Management Science*, vol. 50, pp. 786–796, 2004.

[92] Magnanti, T. L. and Stratila, D., "Separable concave optimization approximately equals piecewise linear optimization.," in *IPCO* (Bienstock, D. and Nemhauser, G. L., eds.), vol. 3064 of *Lecture Notes in Computer Science*, pp. 234–243, Springer, 2004.

[93] Marchand, H. and Wolsey, L., "Aggregation and mixed integer rounding to solve mips," *Operations Research*, vol. 49, pp. 363–371, 2001.

[94] Maringer, D. and Kellerer, H., "Optimization of cardinality constrained portfolios with a hybrid local search algorithm," *OR Spectrum*, vol. 25, pp. 481–495, 2003.

[95] Markowitz, H. M. and Manne, A. S., "On the solution of discrete programming-problems," *Econometrica*, vol. 25, pp. 84–110, 1957.

[96] Martin, A., Moller, M., and Moritz, S., "Mixed integer models for the stationary case of gas network optimization," *Mathematical Programming*, vol. 105, pp. 563–582, 2006.

[97] Martin, R., "Using separation algorithms to generate mixed integer model reformulations," *Operations Research Letters*, vol. 10, pp. 119–128, 1991.

[98] Meyer, R. R., "On the existence of optimal solutions to integer and mixed-integer programming problems," *Mathematical Programming*, vol. 7, pp. 223–235, 1974.

[99] Meyer, R. R., "Integer and mixed-integer programming models - general properties," *Journal of Optimization Theory Applications*, vol. 16, pp. 191–206, 1975.

[100] Meyer, R. R., "Mixed integer minimization models for piecewise-linear functions of a single variable," *Discrete Mathematics*, vol. 16, pp. 163–171, 1976.

[101] Meyer, R. R., "A theoretical and computational comparison of equivalent mixed-integer formulations," *Naval Research Logistics*, vol. 28, pp. 115–131, 1981.

[102] Mhaskar, H. N. and Pai, D. V., *Fundamentals of approximation theory*. Boca Raton: CRC Press, 2000.

[103] Misener, R., Gounaris, C. E., and Floudas, C. A., "Global optimization of gas lifting operations: A comparative study of piecewise linear formulations," *Industrial & Engineering Chemistry Research (To appear)*, 2008. doi:10.1021/ie8012117.

[104] Nemhauser, G. L. and Wolsey, L. A., *Integer and combinatorial optimization*. Wiley-Interscience, 1988.

[105] Padberg, M., "Approximating separable nonlinear functions via mixed zero-one programs," *Operations Research Letters*, vol. 27, pp. 1–5, 2000.

[106] Padberg, M. W. and Rijal, M. P., *Location, Scheduling, Design, and Integer Programming*. Springer, 1996.

[107] Pochet, Y. and Wolsey, L. A., *Production planning by mixed integer programming*. Springer, 2006.

[108] Pottmann, H., Krasauskas, R., Hamann, B., Joy, K. I., and Seibold, W., "On piecewise linear approximation of quadratic functions," *Journal for Geometry and Graphics*, vol. 4, pp. 9–31, 2000.

[109] Prekopa, A., "Probabilistic programming," in *Stochastic Programming* (Shapiro, A. and Ruszczynski, A., eds.), vol. 10 of *Handbooks in Operations Research and Management Science*, pp. 267–351, Elsevier, 2003.

[110] Prenter, P. M., *Splines and Variational Methods*. Dover, 1975.

[111] Quesada, I. and Grossmann, I., "An lp/nlp based branch and bound algorithm for convex minlp optimization problems," *Computers & Chemical Engineering*, vol. 16, pp. 937–947, 1992.

[112] Rardin, R. L., *Optimization in Operations Research*. Prentice Hall, 1998.

[113] Ruszczynski, A., "Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra," *Mathematical Programming*, vol. 93, pp. 195–215, 2002.

[114] Saxena, A., Goyal, V., and Lejeune, M. A., "MIP reformulations of the probabilistic set covering problem," *Mathematical Programming (To appear)*, 2008. doi:10.1007/s10107-008-0224-y.

[115] Schrijver, A., *Theory of linear and integer programming*. John Wiley & Sons, Inc., 1986.

[116] Sen, S., "Rrelaxations for probabilistically constrained programs with discrete random variables," *Operations Research Letters*, vol. 11, pp. 81–86, 1992.

[117] Shapiro, A., Dentcheva, D., and Ruszczynski, A., *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2009.

[118] Sherali, H. D., "On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions," *Operations Research Letters*, vol. 28, pp. 155–160, 2001.

[119] Sherali, H. D. and Shetty, C. M., *Optimization with Disjunctive Constraints*, vol. 181 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, 1980.

[120] Shields, R. personal communication, 2007.

[121] Stanley, R. P., *Enumerative Combinatorics*, vol. 1. Cambridge University Press, 1997.

[122] Stralberg, D., Applegate, D. L., Phillips, S. J., Herzog, M. P., Nur, N., and Warnock, N., "Optimizing wetland restoration and management for avian communities using a mixed integer programming approach," *Biological Conservation*, vol. 142, pp. 94–109, 2009.

[123] Stubbs, R. A. and Mehrotra, S., "A branch-and-cut method for 0-1 mixed convex programming," *Mathematical Programming*, vol. 86, pp. 515–532, 1999.

[124] Tawarmalani, M. and Sahinidis, N. V., "Global optimization of mixed-integer nonlinear programs: A theoretical and computational study," *Mathematical Programming*, vol. 99, pp. 563–591, 2004.

[125] Todd, M. J., "Union jack triangulations," in *Fixed Points: algorithms and applications* (Karamardian, S., ed.), pp. 315–336, Academic Press, 1977.

[126] Todd, M. J., *The computation of fixed points*, vol. 124 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, 1979.

[127] Tomlin, J. A., "A suggested extension of special ordered sets to non-separable non-convex programming problems," in *Studies on Graphs and Discrete Programming* (Hansen, P., ed.), vol. 11 of *Annals of Discrete Mathematics*, pp. 359–370, North Holland, 1981.

[128] Vajda, S., *Mathematical Programming*. Addison-Wesley, 1964.

[129] Vance, P. H., Barnhart, C., Johnson, E. L., and Nemhauser, G. L., "Airline crew scheduling: A new formulation and decomposition algorithm," *Operations Research*, vol. 45, pp. 188–200, 1995.

[130] Vielma, J. P., Ahmed, S., and Nemhauser, G. L., "Mixed-integer models for nonseparable piecewise linear optimization: Unifying framework and extensions," *Operations Research (To appear)*, 2009.

[131] Vielma, J. P., Keha, A. B., and Nemhauser, G. L., "Nonconvex, lower semi-continuous piecewise linear optimization," *Discrete Optimization*, vol. 5, pp. 467–488, 2008.

[132] Vielma, J. P. and Nemhauser, G. L., "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," in *IPCO* (Lodi, A., Panconesi, A., and Rinaldi, G., eds.), vol. 5035 of *Lecture Notes in Computer Science*, pp. 199–213, Springer, 2008.

[133] Vielma, J. P. and Nemhauser, G. L., "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," *Mathematical Programming (To appear)*, 2008.

[134] Watters, L. J., "Reduction of integer polynomial programming problems to zero-one linear programming problems," *Operations Research*, vol. 15, pp. 1171–1174, 1967.

[135] Weintraub, A., "Integer programming in forestry," *Annals of Operations Research*, vol. 149, pp. 209–216, 2007.

[136] Westerlund, T. and Pettersson, F., "An extended cutting plane method for solving convex minlp problems," *Computers & Chemical Engineering*, vol. 19, pp. S131–S136, 1995.

[137] Westerlund, T., Pettersson, F., and Grossmann, I., "Optimization of pump configurations as a minlp problem," *Computers & Chemical Engineering*, vol. 18, pp. 845–858, 1994.

[138] Wilf., H. S., *Combinatorial algorithms–an update*, vol. 55 of *CBMS-NSF regional conference series in applied mathematics*. Society for Industrial and Applied Mathematics, 1989.

[139] Williams, H. P., *Model building in mathematical programming*. Wiley, 4th ed., 1999.

[140] Wilson, D., *Polyhedral methods for piecewise-linear functions*. PhD thesis, University of Kentucky, 1998.

[141] Wolsey, L. A., *Integer Programming*. Wiley and Sons, 1998.

[142] Ziegler, G. M., *Lectures on Polytopes*. Springer-Verlag, 1995.