NORTHWESTERN UNIVERSITY

Methods for Linear Programs with Complementarity Constraints

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Sciences

By

Francisco I. Jara-Moroni

EVANSTON, ILLINOIS

September 2018

ABSTRACT

Methods for Linear Programs with Complementarity Constraints

Francisco I. Jara-Moroni

This thesis studies three approaches for solving linear programs with complementarity constraints (LPCC). The focus of Chapter 2 lies on difference-of-convex (DC) penalty formulations and the associated difference-of-convex algorithm (DCA) for computing stationary solutions of LPCCs. We concentrate on three such formulations and establish connections between their stationary solutions and those of the LPCC. Additionally, improvements of the DCA are proposed to remedy certain drawbacks in a straightforward adaptation of the DCA to these formulations. Extensive numerical results, including comparisons with an existing nonlinear programming solver and the mixed-integer formulation, are presented to elucidate the effectiveness of the overall DC approach.

On Chapter 3 we present a global optimization algorithm for LPCCs, based on a logical Benders approach introduced by Hu et al. (2008). Complementarity pieces are selected from an ever-updating branch-and-bound tree formed from satisfiability clauses, and then discarded by means of a hybrid cut generating procedure which combines an

ℓ_1-norm formulation with a sequential sparsification process. Numerical results show that the number of iterations decreases as compared with the base method.

Finally, Chapter 4 shows a numerical study for a global-local approach to find good quality local optima for stochastic LPCCs derived from stochastic linear bilevel programs with *here-and-now* and *wait-and-see* outer and inner level decisions, respectively. We approach the stochastic problem via Sample Average Approximation and we analyze how local solvers for LPCC perform when the starting point is the global solution of a subsampled LPCC, in terms of their proximity to the global solution of the stochastic LPCC. Experiments on a Bilevel Network Newsvendor Problem allow us to present a simple technique to improve the solutions provided by the local solvers by taking advantage of degeneracy of the inner level problems.

Dedication

To Bárbara, Amanda and Isabel

To Momy and Argos

To Daniel Rabinovic

Acknowledgements

Finally reaching the end of my PhD life as a student, I would like to express my enormous gratitude to the people who have made this journey an amazing one, although it is impossible to list everyone in just a couple of pages.

I would like to start with my advisor Andreas Wächter, who within these 5 years has become not only my guide, but also my friend. Andreas, thanks for all your support, patience and help, and for accepting me as your student. Additionally, my biggest and sincerest gratitude goes to professors Jorge Nocedal, Jong-Shi Pang and John Mitchell. I learned a lot from the three of you and I really hope our lives cross once again in the future.

I will also take the time to thank CONICYT for their invaluable financial support. It would have been hard to go through these years without it, specially when raising two kids.

Having people visiting us during this period made everything so much easier for us, which is why we are so grateful of all our friends who took their time to come to Evanston and share so many moments with us. This is a long list so be patient...

I will start with my in-laws, Carlos & María. It was wonderful to have you at our home, where you could share the first months of Isabel. Pame, Micky and Andrea and their amazing kids. We really had a blast with you guys, at the Dells and Disney!! A special mention goes, of course, to Aída. Not everyone has the luck to have their grandmother travel all the way to the states at an age older than 90!! Thanks for that visit, with Goyo and Lalo. Continuing with my dearest friends from high school, Mati and Jani, and their lovely girls, Maite and Viole. We really loved having you there, talking and enjoying time as if we had seen each other the week before. Also, Jose, Claudia, Benja and Nico, walking around Chicago with you guys was amazing!

Life at work would not have been the same without the awesome group of wonderful people that joined me in the lab: Alvaro, Ben, Nitish, Albert, Raghu, Ale, Michael, Ruby and Yuchen. I am really grateful for having shared the office with such amazing people. Special mention to my great friend Albert, not only my lab mate, but also team mate. Thanks for being so kind, and good person, and so selfless. And thanks, to you and Ale, for treating my girls so nicely and spoiling them every time.

Of course life outside the academic world, while leaving abroad, was made a lot easier to handle thanks to the fantastic group of chilean friends we made on the way. In order of appearance, Nico, Vale, Marco, Vicky, Yosune and Cristóbal, and, of course, the little ones, Helena, Martina, Cristóbal and Amaia. I would not want to imagine how tough these 5 years would have been had we not crossed paths with you. Thank you all for your invaluable support and kindness.

Approaching the end of this acknowledgements, my biggest and eternal gratitude to my brother Pedro, alongside with his wonderful family, Javi, Bea and Manu, and my lovely parents, Momy and Sergio. We are so thankful for your visits, and for all the help you provided until the very end of our trip. That trip to Boston was one of the major highlights of this adventure and remembering our childhood with all of you will always remain in my heart.

Finally, the most important of all these recognitions, my three favorite girls, Bárbara, Amanda and Isa. The main reason I woke up every day and went to work. It would have been impossible to achieve this without your support. Thanks for joining me in this trip. I love you.

Francisco Jara-Moroni

Evanston, June 11, 2018.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

This thesis investigates the computation of local and global solutions of linear programs with complementarity constraints (LPCCs) of the form

$$\begin{array}{ll} \underset{x,y,w}{\text{minimize}} & c^T x + d^T y \\ \text{subject to} & Ax + By \geq f \\ & Mx + Ny + q = w \\ \text{and} & 0 \leq y \perp w \geq 0, \end{array} \tag{1.1}$$

where $c \in \mathbb{R}^t$, $d, q \in \mathbb{R}^n$, $f \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times t}$, $B \in \mathbb{R}^{m\times n}$, $M \in \mathbb{R}^{n\times t}$ and $N \in \mathbb{R}^{n\times n}$. The expression "$y \perp w$" means that the vectors w and y are orthogonal, i.e., $w^T y = 0$. Beginning with the early work in the integer programming community on what was known then as the "complementary program" (Ibaraki, 1971, 1973; Jeroslow, 1978), and as a special case of a mathematical program with equilibrium constraints (MPECs) (Luo et al., 1996) on which there is an extensive literature to date, the LPCC has grown in importance in applications (Hu et al., 2012b) as complementarity constraints can be used to model many logical, piecewise, and nonconvex conditions (Hu et al., 2012b), even discontinuous ones such as cardinality objective (Feng et al., 2013) and constraints (Burdakov et al., 2016). In addition, the LPCC provides an interesting framework for the study of nonconvex quadratic programs (Hu et al., 2012a). In general, determining the global solution of

LPCCs is NP-hard. While there has been recent research into the global solution of the LPCC (Hu et al., 2008; Yu, 2011) and its extension to a quadratic program with complementarity constraints (Bai et al., 2013), the task of designing a general-purpose procedure with a provable certificate to compute a globally optimal solution to the LPCC efficiently, if it exists, remains practically challenging.

With respect to local solutions, LPCCs still represent a challenging problem since no feasible solution can satisfy all inequalities strictly. Therefore, the Mangasarian-Fromovitz constraint qualification (MFCQ) is violated at every feasible point (Scheel and Scholtes, 2000). This absence of regularity makes traditional non-linear programming (NLP) approaches tend to perform poorly, and the algorithms may converge to spurious stationary points which allow for trivial descent directions over the feasible region of the LPCC (Fang et al., 2012). For example, sequential quadratic programming (SQP) methods can give rise to inconsistent constraint linearizations, while interior point methods present conflicting goals of enforcing complementarity while keeping the variables away from their bounds (Leyffer et al., 2006).

Adding stochasticity to an LPCC gives way to a much broader set of applications, mostly related to hierarchical optimization, in fields such as economic equilibrium, engineering design, logistics, transportation and energy, to name a few. The presence of uncertainty, though, makes the LPCC an even harder problem to tackle, even if the underlying random variables have a finite support, due to large scale nature of its stochastic counterpart. In the presence of continuous random variables, several works have been oriented to prove convergence of stationary points, from discretized deterministic LPCCs, to stationary points of their stochastic counterpart, extending the traditional results of convergence of globally optimal solution and value, from stochastic programming (Shapiro and Xu, 2008; Xu and Jane, 2011).

1.1. Overview

In this thesis we present our research related to local and global optima, and stochastic LPCCs, divided into three chapters.

In Chapter 2 we study the application of the difference-of-convex algorithm (DCA) to penalized versions of the LPCC, with an exact (smooth and non-smooth) penalization term. First we introduce several concepts of stationarity and state their relationships and equivalences, as well as highlight the difficulty of assessing whether a candidate optimal solution falls into any of those definitions. Later we describe the DCA in detail alongside with its convergence properties and how it fits with the presented penalized versions of the LPCC. We also introduce three different decompositions over which we can execute the DCA and state the properties of its limiting points, from the perspective of the LPCC problem. Before presenting the numerical results, we propose two enhancements over the basic DCA: one based on corrected reduced costs, which allows the non-smooth penalized LPCC to escape weakly stationary points, and an active set method which overcomes the inherent slow speed of first order methods for the quadratic decomposition of the smooth penalization. Extensive numerical results for the linear complementarity problems, inverse quadratic problems and problems from the MacMPEC library are presented.

Chapter 3 extends the works of Hu et al. (2008) and Bai et al. (2013) in a logical Benders approach for globally solving LPCCs. The main contribution of this chapter is to present an interpretation of the logical Benders method as a reversed branch-and-bound method, where the whole exploration procedure starts from the leaf nodes, as opposed to the root node, in an enumeration tree. We use this interpretation to provide a new framework over which we can combine master problem and cut generation, as in the traditional Benders algorithm, in a single process. We also present an ℓ_1-norm formulation which makes the cut generation more efficient. Numerical results over the test sets of Hu et al. (2008) are performed and extended to larger complementarity dimensions, exceeding what the state-of-the-art has been able to solve so far.

Finally, we present a numerical study for the Stochastic Linear Bilevel Network Newsvendor problem, by means of its LPCC formulation. We combine both global and local solvers to obtain good approximations of global solutions of a Sampled Average Approximation version of the problem, obtained by drawing a large sample of scenarios. We introduce an alternating weights technique which overcomes the degeneracy of the inner level problem and is capable of finding better solutions. We confirm that in the small instances we are able to find the global solution of the real stochastic problem, while in the larger ones the local solutions obtained overtake, from the point of view of the true stochastic problem, those found from globally solving smaller subsamples. We also highlight that the time taken by combining global and local methods is drastically reduced, as compared to the best performing subsampled global solution, and show that the alternating weights technique manages to find good approximations of global solutions, even if the subsample is not solved to global optimality.

CHAPTER 2

Local Solutions and Stationarity of Linear Programs with Complementarity Constraints

2.1. A Study of the Difference-of-Convex Approach for Solving Linear Programs with Complementarity Constraints

2.1.1. Introduction

This chapter extends the work of two references (Le Thi and Dinh, 2011; Muu et al., 2011) in which problems with complementarity constraints are solved via their reformulations as difference-of-convex (DC) programs; these reformulated DC programs are solved by the difference-of-convex algorithm (DCA) pioneered by Le Thi Hoai An and Pham Dinh Tao (Le Thi et al., 2014; Dinh and Le Thi, 2014; Le Thi and Dinh, 2013; Tao et al., 2005; Tao and An, 1997). In (Le Thi and Dinh, 2011), based on several DC decompositions, the DCA was applied to the standard linear complementarity problem (Cottle et al., 1992); in Muu et al. (2011), the DC approach was embedded in a branch-and-bound scheme for solving mathematical programs with affine variational inequality constraints. Our work is in the spirit of (Le Thi and Dinh, 2011); specifically, we introduce several penalized formulations of the complementarity constraint and employ various DC decompositions of the resulting penalty functions, based on which the DCA is applied to the resulting penalized DC programs and enhancements to this basic algorithm are introduced. This approach is different from a nonlinear programming approach as implemented in the solvers FILTER (Fletcher and Leyffer, 2002; Fletcher et al., 2002) and KNITRO (Byrd et al., 2006), which are applicable to deal with the complementarity constraints (Fletcher* and Leyffer, 2004; Leyffer et al., 2006). Recently, motivated by a sequential LPCC method for solving MPCCs (Leyffer and Munson, 2007), a pivoting method (Fang et al., 2012) is developed that includes an anticycling technique for verifying “B(ouligand)-stationarity”. For a survey of algorithms for solving LPCCs, see (Júdice, 2012).

The organization of the remainder of this chapter is as follows. After formally defining the LPCC and introducing three main stationarity concepts for this problem in the next section, we briefly review the DCA for computing a critical point of a DC program in Section 2.1.3. Three penalty formulations of complementarity constraint are then discussed in detail in Section 2.1.4 that is divided into subsections, one for each penalty formulation. Section 2.2 describes enhancements to the DCA applied to the penalized formulations of the LPCC. Extensive computation results are reported in Section 2.3 to support the practical performance of the enhanced DCA. Overall, this chapter has allowed us to gain a deeper understanding of the DC approach to the local solution of the LPCC.

2.1.2. Concepts of Stationarity

Stationarity concepts for mathematical programs with complementarity constraints have been well studied. For the LPCC, the references (Fletcher et al., 2006, Sections 3 and 4), (Scheel and Scholtes, 2000) and (Fang et al., 2012, Section 2) are most pertinent to the following discussion; in particular, background and details of these stationarity concepts can be found therein. As it is well known, the diversity of these concepts is due to the disjunctive nature of the complementarity constraint. Different ways handling this combinatorial constraint leads to the individual stationarity notions which are made precise in Definition 2.1.1. For concise notation, we define the set of linear constraints in the LPCC (1.1) without the complementarity condition:

$$\Omega \triangleq \left\{ (x, y, w) \in \mathbb{R}^t \times \mathbb{R}^{2n}_+ \mid Ax + By \geq f,\, Mx + Ny + q = w \right\}.$$

Let $(\bar{x}, \bar{y}, \bar{w})$ be a feasible point of the LPCC (1.1). We define the following partition of the variable indices for y and w:

$$\begin{aligned} \mathcal{I}_y(\bar{y}, \bar{w}) &\triangleq \{ i \mid \bar{y}_i = 0 < \bar{w}_i \}, \quad \mathcal{I}_w(\bar{y}, \bar{w}) \triangleq \{ i \mid \bar{y}_i > 0 = \bar{w}_i \}, \\ \mathcal{I}_0(\bar{y}, \bar{w}) &\triangleq \{ i \mid \bar{y}_i = 0 = \bar{w}_i \}, \end{aligned} \tag{2.1}$$

and refer to the indices in $\mathcal{I}_0(\bar{y}, \bar{w})$ as the degenerate (or bi-active) indices, for which strict complementarity fails to hold. Corresponding to these index sets, we define the *relaxed linear program*:

$$\begin{array}{lll} \text{RxLP:} & \underset{(x,y,w)\in\Omega}{\text{minimize}} & c^T x + d^T y \\ & & y_i = 0 \leq w_i, \quad i \in \mathcal{I}_y(\bar{y}, \bar{w}) \\ & & y_i \geq 0 = w_i, \quad i \in \mathcal{I}_w(\bar{y}, \bar{w}) \\ & \text{and} & y_i, w_i \geq 0, \qquad i \in \mathcal{I}_0(\bar{y}, \bar{w}) \end{array}$$

and the *restricted linear program*:

$$\begin{array}{rll} \text{RsLP:} & \underset{(x,y,w)\in\Omega}{\text{minimize}} & c^T x + d^T y \\ & & y_i \,=\, 0 \,\leq\, w_i, \quad i \,\in\, \mathcal{I}_y(\bar{y},\bar{w}) \\ & & y_i \,\geq\, 0 \,=\, w_i, \quad i \,\in\, \mathcal{I}_w(\bar{y},\bar{w}) \\ & \text{and} & y_i, w_i \,=\, 0, \qquad i \,\in\, \mathcal{I}_0(\bar{y},\bar{w}) \end{array}$$

Moreover, for every subset $\mathcal{J} \subseteq \mathcal{I}_0(\bar{y},\bar{w})$, we also define the *piecewise linear program*:

$$\begin{array}{rll} \text{LP}(\mathcal{J})\text{:} & \underset{(x,y,w)\in\Omega}{\text{minimize}} & c^T x + d^T y \\ & & y_i \,=\, 0 \,\leq\, w_i, \quad i \,\in\, \mathcal{I}_y(\bar{y},\bar{w}) \\ & & y_i \,\geq\, 0 \,=\, w_i, \quad i \,\in\, \mathcal{I}_w(\bar{y},\bar{w}) \\ & & y_i \,=\, 0 \,\leq\, w_i, \quad i \,\in\, \mathcal{J} \\ & \text{and} & y_i \,\geq\, 0 \,=\, w_i, \quad i \,\in\, \mathcal{I}_0(\bar{y},\bar{w}) \setminus \mathcal{J}, \end{array}$$

which is a *piece* of the LPCC at the given point $(\bar{x},\bar{y},\bar{w})$. Based on the LPs: (RxLP), (RsLP), and LP($\mathcal{J}$), we have the following definition.

Definition 2.1.1. A feasible triple $(\bar{x},\bar{y},\bar{w})$ of the LPCC (1.1) is said to be

- *strongly stationary* if it is a solution of the relaxed LP (RxLP);
- *weakly stationary* if it is a solution of the restricted LP (RsLP);
- *B(ouligand)-stationary* if it is a solution of LP($\mathcal{J}$) for all $\mathcal{J} \subseteq \mathcal{I}_0(\bar{y},\bar{w})$. □

These stationary concepts can be described in terms of multipliers of the constraints; in particular, using the notation $\circ$ for the Hadamard product of two vectors, we see that a triple $(\bar{x}, \bar{y}, \bar{w}) \in \Omega$ is weakly stationary if and only if it is feasible to (1.1) and there exist multipliers $\lambda \in \mathbb{R}^m$ and μ^y and $\mu^w \in \mathbb{R}^n$, so that

$$\begin{array}{lcl} A^T\lambda + M^T\mu^w & = & c \\ B^T\lambda + N^T\mu^w + \mu^y & = & d \\ 0 \,\leq\, \lambda \,\perp\, A\bar{x} + B\bar{y} - f & \geq & 0 \\ \bar{y} \circ \mu^y \,=\, 0 \;\text{ and }\; \bar{w} \circ \mu^w & = & 0. \end{array} \tag{2.2}$$

Notice that there is no sign restriction on μ_i^y and μ_i^w for $i \in \mathcal{I}_0(\bar{y}, \bar{w})$); if these bi-active multipliers are restricted to be nonnegative, then the resulting conditions are equivalent to the strong stationarity of the given triple $(\bar{x}, \bar{y}, \bar{w})$. In contrast, B-stationarity cannot be described in terms of a single tuple of multipliers due to the multiple LPs that depend on the index subsets $\mathcal{J}$ of $\mathcal{I}_0(\bar{y}, \bar{w})$. Proposition 2.1.2 elucidates the connections among the above stationarity conditions and the local minimizing property of the triple. An example in (Scheel and Scholtes, 2000) shows that a B-stationary point is not necessarily strongly stationary. Related results will be given in the next section when we discuss various penalty formulations of the complementarity constraint.

Proposition 2.1.2. Let $(\bar{x}, \bar{y}, \bar{w})$ be a given feasible solution of (1.1). The following implications hold:

$$\begin{tikzcd} \text{strong stationarity} \arrow[r, Rightarrow] & \text{local minimizing} \arrow[d, Leftrightarrow] & \\ & \text{B-stationarity} \arrow[r, Rightarrow] & \text{weak stationarity.} \end{tikzcd}$$

Proof. It is clear that strong stationarity implies B-stationarity which implies weak stationarity. It remains to show that B-stationarity is equivalent to local minimizing. This is not difficult to see because locally near $(\bar{x}, \bar{y}, \bar{w})$, the feasible region of the LPCC is the union of the feasible sets of the pieces of the LP$(\mathcal{J})$ for all $\mathcal{J} \subseteq \mathcal{I}_0(\bar{y}, \bar{w})$. (This fact was noted in the early work of the MPCC as described in (Luo et al., 1996).) $\square$

With the equivalence of B-stationarity and the locally minimizing property (for the LPCC), we see that this is the sharpest among the other stationarity concepts. One way to compute a B-stationary solution is to apply a global resolution scheme such as the logical Benders approach described in (Hu et al., 2008) or other global methods (Muu et al., 2011; Yu, 2011) that invariably involve some kind of enumeration. Alternatively, it is possible to formulate the LPCC as a difference-of-convex constrained program and apply the (deterministic) algorithm described in (Pang et al., 2016) that is designed to compute a B-stationary solution of the latter program. Such an algorithm also involves some level of enumeration which can be alleviated using a probabilistic choice of subprograms to be solved whose almost sure convergence to a B-stationary solution can be established. This paper does not address either of these two approaches; instead we focus on a penalized DC approach introduced in the next section. As we will see, the notions of strongly and weakly stationary solutions have a relevant role to play in the various DC formulations of the LPCC; see Propositions 2.1.6, 2.1.7 and 2.1.9.

2.1.3. The Difference of Convex Functions Approach

The proposed method replaces the complementarity constraint $0 \leq y \perp w \geq 0$ by a penalization term that is added to the objective function:

$$\underset{(x,y,w)\in\Omega}{\text{minimize}}\ c^T x + d^T y + \rho\,\phi(y,w). \tag{2.3}$$

Here, $\rho > 0$ is a penalty parameter, and the penalty function $\phi : \mathbb{R}^n_+ \times \mathbb{R}^n_+ \longrightarrow \mathbb{R}_+$ is zero if and only if $y \perp w$. Starting from (Luo et al., 1996), penalty formulations have been studied extensively in the literature of the MPCCs. Our focus here is based on the assumption that $\phi(y,w)$ is a difference of two convex functions, i.e.

$$\phi(y,w) \;=\; \phi_+(y,w) - \phi_-(y,w), \tag{2.4}$$

where ϕ_+ and ϕ_- are convex. Writing the objective function in (2.3) as $f_+(x,y,w) - f_-(x,y,w)$ with $f_+(x,y,w) = c^T x + d^T y + \rho\phi_+(y,w)$ and $f_-(x,y,w) = \rho\phi_-(y,w)$, we see that (2.3) is a DC program which has the following general form:

$$\underset{z\in C}{\text{minimize}}\ f(z) \;\triangleq\; f_1(z) - f_2(z), \tag{2.5}$$

where $f_1,\ f_2 : \mathbb{R}^m \longrightarrow \mathbb{R}$ are convex and $C \subseteq \mathbb{R}^m$ is a closed convex set. The basic idea of the DCA (Le Thi and Dinh, 2013; Tao and An, 1997; Tao et al., 2005) for solving (2.5) is as follows: At an iterate z^k, the algorithm overestimates the concave part of the objective function by a linear function, using a subgradient $g^k \in \partial f_2(z^k)$ of f_2 at z^k. An optimal solution of the resulting convex auxiliary problem

$$\min_{z \in C} \; f_1(z) - (g^k)^T(z - z^k) \tag{2.6}$$

provides the new iterate. Needless to say, the choice of the subgradient has an important effect on the practical efficiency of the algorithm; we will discuss more about such choices in specific contexts of the function f_2. The algorithm is formally stated in Algorithm 1.

Algorithm 1 Basic difference-of-convex functions algorithm

```
1: Choose termination tolerance $\varepsilon_{\text{tol}} > 0$.
2: Let $z^1 \in C$ and $k \leftarrow 0$.
3: **repeat**
4:     Set $k \leftarrow k + 1$.
5:     Compute subgradient $g^k \in \partial f_2(z^k)$ of $f_2$ at $z^k$.
6:     Compute the new iterate $z^{k+1}$ as solution of (2.6).
7: **until** $f(z^k) - f(z^{k+1}) \le \varepsilon_{\text{tol}}$
8: Return $z^{k+1}$.
```

Properties of the DCA have been well known in the DC literature, see e.g. (Le Thi et al., 2014; Le Thi and Dinh, 2013). In particular, it has been shown that when $z^* = z^k$ is returned by the algorithm for some finite k or if z^* is a limit point of an infinite sequence $\{z^k\}$ (with $\varepsilon_{\text{tol}} = 0$), then z^* is a critical point for (2.5) in the sense that $\partial f_2(z^*) \cap (\partial f_1(z^*) + \mathcal{N}(z^*; C)) \neq \emptyset$, where $\mathcal{N}(z^*; C)$ is the *normal cone* of C at z^*. Moreover, if C is polyhedral, as in the case of (2.3), and if either f_1 or f_2 is *polyhedral convex* (Rockafellar, 1997), then the algorithm will terminate in a finite number of steps. Subsequently, we show in Proposition 2.1.5 that criticality in a penalized piecewise linear (thus non-differentiable) DC formulation of the LPCC is equivalent to weak stationarity of the LPCC. It has been noted in (Pang et al., 2016) that the condition of criticality is in general weaker than that of d(irectional)-stationarity for a convex constrained DC program. Specifically, $z^* \in C$ is d-stationary for (2.5) if $f'(z^*, z - z^*) \geq 0$ for all $z \in C$, where the prime $'$ notation denotes the directional derivative at z^* in the direction $z - z^*$; the stationarity condition is equivalent to the inclusion $\partial f_2(z^*) \subseteq (\partial f_1(z^*) + \mathcal{N}(z^*; C))$. It is an elementary fact that d-stationarity is a necessary condition of the locally minimizing property for a convex constrained optimization problem with a directionally differentiable objective function.

2.1.4. DC Penalty Functions

We present three different choices for the penalty function ϕ in the following subsections. They correspond to the DC objective functions used in (Le Thi and Dinh, 2011) for the solution of linear complementarity problems and are here extended to the LPCC. This reference presents also a fourth DC function (the "simple constrained quadratic program"); since the numerical results therein show rather poor performance with this formulation, we are not exploring it in this paper. Besides providing the foundation for the penalty-based algorithmic approach for the solution of the LPCC, the discussion in the subsections herein relate the stationarity conditions of the penalty formulation to the various LPCC stationarity conditions introduced in Definition 2.1.1. While connections like these have been discussed to some extent in the MPCC literature (see e.g. Fang et al. (2012); Fletcher et al. (2006)), there are two emphases in our presentation that are not fully transparent in this literature. First, we address the stationarity conditions of the penalty formulations for fixed values of the penalty parameter ρ; second, being nonconvex optimization problems, practically computable solutions to the penalty formulations are typically of the stationary kind; it is therefore important to understand how such computed solutions translate into the LPCC stationarity solutions.

2.1.4.1. A piecewise linear penalty. The first penalty function that we consider is

$$\phi^{\mathrm{PL}}(y,w) \triangleq \sum_{i=1}^{n} \min\,(\,y_i, w_i\,) \tag{2.7}$$

which is clearly a concave, thus DC, function. This corresponds to the "concave separable minimization" formulation in (Le Thi and Dinh, 2011). The authors of this reference identified this as the most effective formulation among the ones they used for solving the standard LCP. Incidentally, this penalty function was used in a successive linearization algorithm (Mangasarian, 1997) that reformulates the LCP as a concave minimization problem. With (2.7) as the penalty function, the problem (2.4) becomes

$$\underset{z \triangleq (x,y,w) \in \Omega}{\text{minimize}} \; \theta^{\mathrm{PL}}(z) \triangleq c^T x + d^T y + \rho \sum_{i=1}^{n} \min(\,y_i, w_i\,). \tag{2.8}$$

In what follows, we examine the stationarity and local minimizing properties of a triple $\bar{z} \triangleq (\bar{x}, \bar{y}, \bar{w}) \in \Omega$ that is feasible for (2.8) but may not be feasible to (1.1). We further show in Proposition 2.1.5 how stationarity in terms of the directional derivatives of the objective function, i.e., d-stationarity, is related to the criticalilty property in terms of the subdifferentials of the two convex components of the DC representation of the min function. We begin with a result showing that $\bar{z} \in \Omega$ is d-stationary for (2.8) if and only if is locally minimizing; we further relate this stationarity property in terms of the LP pieces of the LPCC (1.1), given by $\mathrm{LP}_\rho(\widehat{\mathcal{J}})$

$$\underset{x,y,w\in\Omega}{\text{minimize}}\ c^T x + d^T y + \rho\left[\sum_{i\in\widehat{\mathcal{I}}_y(\bar{y},\bar{w})} y_i + \sum_{i\in\widehat{\mathcal{I}}_w(\bar{y},\bar{w})} w_i + \sum_{i\in\widehat{\mathcal{J}}} y_i + \sum_{i\in\widehat{\mathcal{J}}^c} w_i\right] \tag{2.9}$$

for all subsets $\widehat{\mathcal{J}}$ of $\widehat{\mathcal{I}}_=(\bar{y},\bar{w})$, where

$$\begin{gathered}\widehat{\mathcal{I}}_y(\bar{y},\bar{w}) \triangleq \{\, i \mid \bar{y}_i < \bar{w}_i \,\}, \quad \widehat{\mathcal{I}}_w(\bar{y},\bar{w}) \triangleq \{\, i \mid \bar{w}_i < \bar{y}_i \,\}, \\ \widehat{\mathcal{I}}_=(\bar{y},\bar{w}) \triangleq \{\, i \mid \bar{y}_i = \bar{w}_i \,\}.\end{gathered} \tag{2.10}$$

and the $\widehat{\mathcal{J}}^c$ is taken over $\widehat{\mathcal{I}}_=(\bar{y},\bar{w})$. For easiness of notation, whenever there is no ambiguity, we will drop the term $(\bar{y},\bar{w})$ in $\widehat{\mathcal{I}}_y$, $\widehat{\mathcal{I}}_w$ and $\widehat{\mathcal{I}}_=$.

Proposition 2.1.3. Let $\rho > 0$ and a triple $\bar{z}^\rho \triangleq (\bar{x}^\rho, \bar{y}^\rho, \bar{w}^\rho) \in \Omega$ be given. The following three statements are equivalent.

(a) $\bar{z}^\rho$ is a local minimizer of (2.8);

(b) $\bar{z}^\rho$ is d-stationary for (2.8);

(c) $\bar{z}^\rho$ is a minimizer of the $\mathrm{LP}_\rho(\widehat{\mathcal{J}})$ for every subset $\widehat{\mathcal{J}}$ of $\widehat{\mathcal{I}}_=$.

Proof. (a) $\Rightarrow$ (b). This is trivially true in general for a directionally differentiable objective.

(b) $\Rightarrow$ (c). Suppose that $\bar{z}^\rho$ is d-stationary for (2.8). Let $z = (x, y, w) \in \Omega$ be arbitrary. We then have

$$\begin{aligned}
0 &\leq \theta'(\bar{z}^\rho; z - \bar{z}^\rho) \\
&= c^T(\, x - \bar{x}^\rho\,) + d^T(\, y - \bar{y}^\rho\,) + \\
&\qquad \rho\left[\sum_{i \in \widehat{\mathcal{I}}_y} (\, y_i - \bar{y}_i^\rho\,) + \sum_{i \in \widehat{\mathcal{I}}_w} (\, w_i - \bar{w}_i^\rho\,) + \sum_{i \in \widehat{\mathcal{I}}_=} \min(\, y_i - \bar{y}_i^\rho,\, w_i - \bar{w}_i^\rho\,)\right] \\
&\leq c^T(\, x - \bar{x}^\rho\,) + d^T(\, y - \bar{y}^\rho\,) + \\
&\qquad \rho\left[\sum_{i \in \widehat{\mathcal{I}}_y} (\, y_i - \bar{y}_i^\rho\,) + \sum_{i \in \widehat{\mathcal{I}}_w} (\, w_i - \bar{w}_i^\rho\,) + \sum_{i \in \widehat{\mathcal{J}}} [\, y_i - \bar{y}_i^\rho\,] + \sum_{i \in \widehat{\mathcal{J}}^c} [\, w_i - \bar{w}_i^\rho\,]\right]
\end{aligned}$$

for every $\widehat{\mathcal{J}} \subseteq \widehat{\mathcal{I}}_=$. Thus (c) holds.

(c) $\Rightarrow$ (a). Suppose that $\bar{z}^\rho$ is a local minimizer of the $\mathrm{LP}_\rho(\widehat{\mathcal{J}})$ for every subset $\widehat{\mathcal{J}}$ of $\widehat{\mathcal{I}}_=$. Let $\mathcal{N}_y \triangleq \{(x, y, w) \mid y_i < w_i, i \in \widehat{\mathcal{I}}_y\}$ and $\mathcal{N}_w \triangleq \{(x, y, w) \mid y_i > w_i, i \in \widehat{\mathcal{I}}_w\}$. Take $\mathcal{N} \triangleq \mathcal{N}_y \bigcap \mathcal{N}_w$. Let $(x, y, w) \in \mathcal{N} \cap \Omega$ be arbitrary. By construction of $\mathcal{N}_y$ and $\mathcal{N}_w$ we have

$$\min(y_i, w_i) = \begin{cases} y_i & \text{if } i \in \widehat{\mathcal{I}}_y \\ w_i & \text{if } i \in \widehat{\mathcal{I}}_w. \end{cases}$$

Let $\widehat{\mathcal{J}} \triangleq \left\{ i \in \widehat{\mathcal{I}}_= \mid y_i \leq w_i \right\}$, we then have

$$\min(y_i, w_i) = \begin{cases} y_i & \text{if } i \in \widehat{\mathcal{J}} \\ w_i & \text{if } i \in \widehat{\mathcal{J}}^c. \end{cases}$$

Hence,

$$\begin{aligned}
& c^T \bar{x}^\rho + d^T \bar{y}^\rho + \rho \sum_{i=1}^{n} \min(\, \bar{y}_i^\rho, \bar{w}_i^\rho \,) \\
= \;& c^T \bar{x}^\rho + d^T \bar{y}^\rho + \rho \left[\sum_{i \in \widehat{\mathcal{I}}_y} \bar{y}_i^\rho + \sum_{i \in \widehat{\mathcal{I}}_w} \bar{w}_i^\rho + \sum_{i \in \widehat{\mathcal{J}}} \bar{y}_i^\rho + \sum_{i \in \widehat{\mathcal{J}}^c} \bar{w}_i^\rho \right] \\
\leq \;& c^T x + d^T y + \rho \left[\sum_{i \in \widehat{\mathcal{I}}_y} y_i + \sum_{i \in \widehat{\mathcal{I}}_w} w_i + \sum_{i \in \widehat{\mathcal{J}}} y_i + \sum_{i \in \widehat{\mathcal{J}}^c} w_i \right] \\
= \;& c^T x + d^T y + \rho \sum_{i=1}^{n} \min(\, y_i, w_i \,),
\end{aligned}$$

where the inequality follows from the fact that $\bar{z}^\rho$ is a local minimizer of $\mathrm{LP}_\rho(\widehat{\mathcal{J}})$. $\square$

In the above proposition, the triple $(\bar{x}^\rho, \bar{y}^\rho, \bar{w}^\rho)$ is not necessarily feasible to the LPCC (1.1). If it is, then it is a local minimizer of the LPCC; moreover, $(\bar{x}^\rho, \bar{y}^\rho, \bar{w}^\rho)$ remains locally minimizing for (2.8) for all ρ sufficiently large. The following result clarifies these assertions.

Proposition 2.1.4. Let $\bar{z} \triangleq (\bar{x}, \bar{y}, \bar{w}) \in \Omega$ be given. The following statements are equivalent.

(a): $\bar{z}$ satisfies $\min(\bar{y}, \bar{w}) = 0$ and is a local minimizer of (2.8) for some $\rho > 0$.

(b): $\bar{z}$ is a local minimizer of the LPCC (1.1).

(c): $\bar{z}$ satisfies $\min(\bar{y}, \bar{w}) = 0$ and there exists $\bar{\rho} > 0$ such that $\bar{z}$ is a local minimizer of (2.8) for all $\rho \geq \bar{\rho}$.

Proof. (a) $\Rightarrow$ (b). Let (x, y, w) be an arbitrary triple feasible to (1.1) that is sufficiently close to $(\bar{x}, \bar{y}, \bar{w})$. We then have

$$\begin{aligned} c^T x + d^T y &= c^T x + d^T y + \rho \sum_{i=1}^{n} \min(\, y_i, w_i\,) \\ &\geq c^T \bar{x} + d^T \bar{y} + \rho \sum_{i=1}^{n} \min(\, \bar{y}_i, \bar{w}_i\,) \;\; = c^T \bar{x} + d^T \bar{y}, \end{aligned}$$

where the inequality holds by the local minimizing property of $\bar{z}$ with respect to (2.8).

(b) $\Rightarrow$ (c). By way of contradiction, suppose that $\bar{z}$ is not a local minimizer of (2.8) for a sequence of positive scalars $\{\rho_k\}$ tending to infinity. Proposition 2.1.3 then implies that for each k there exists $\widehat{\mathcal{J}}_k \subseteq \widehat{\mathcal{I}}_=$ so that $\bar{z}$ is not a minimizer of the $\text{LP}_{\rho_k}(\widehat{\mathcal{J}}_k)$. Note that $\widehat{\mathcal{I}}_= = \mathcal{I}_0(\bar{y}, \bar{w})$ since $\bar{z}$ is feasible to the LPCC. Because there are only finitely many subsets of $\mathcal{I}_0(\bar{y}, \bar{w})$, we may assume, without loss of generality, that $\widehat{\mathcal{J}}_k = \mathcal{J}$ for all k for some $\mathcal{J} \subseteq \mathcal{I}_0(\bar{y}, \bar{w})$. Since $\bar{z}$ is a local minimizer of the LPCC, Proposition 2.1.2 implies that $\bar{z}$ is a B-stationary point, and therefore a minimizer of the $\text{LP}(\mathcal{J})$. Note that $\text{LP}_{\rho_k}(\mathcal{J})$ is a standard penalty formulation of $\text{LP}(\mathcal{J})$. Therefore $\bar{z}$ is also a minimizer to $\text{LP}_{\rho_k}(\mathcal{J})$ for sufficiently large ρ_k, leading to a contradiction.

(c) $\Rightarrow$ (a). This is obvious. $\square$

2.1.4.2. Criticality and weak stationarity. Among many DC decompositions (2.5) of the piecewise linear DC objective θ^{PL}, we employ the one that is given by $f_1^{\text{PL}}(z) \triangleq c^T x + d^T y$ and $f_2^{\text{PL}}(z) \triangleq \displaystyle\sum_{i=1}^{n} \max(-y_i, -w_i)$. With this decomposition, we have the following result which re-affirms that criticality defined for this DC formulation is weaker than d-stationarity for the LPCC. Recall that criticality in this context means: $\partial f_2^{\text{PL}}(z^*) \cap \big(\partial f_1^{\text{PL}}(z^*) + \mathcal{N}(z^*; \Omega)\big) \neq \emptyset$.

Proposition 2.1.5. Let $\rho > 0$ and a triple $\bar{z}^\rho \triangleq (\bar{x}^\rho, \bar{y}^\rho, \bar{w}^\rho) \in \Omega$ be given. The following three statements are equivalent with respect to the DC decomposition $(f_1^{\rm PL}, f_2^{\rm PL})$ of $\theta^{\rm PL}$:

(a) $\bar{z}^\rho$ is a critical point of the DC program (2.8);

(b) for every $(x, y, w) \in \Omega$,

$$c^T(\,x - \bar{x}^\rho\,) + d^T(\,y - \bar{y}^\rho\,) +$$
$$\rho\left[\sum_{i \in \widehat{\mathcal{I}}_y} (\,y_i - \bar{y}_i^\rho\,) + \sum_{i \in \widehat{\mathcal{I}}_w} (\,w_i - \bar{w}_i^\rho\,) + \sum_{i \in \widehat{\mathcal{I}}_=} \max(\,y_i - \bar{y}_i^\rho,\, w_i - \bar{w}_i^\rho\,)\right] \geq 0;$$

(c) $\bar{z}^\rho$ is an optimal solution of the following convex piecewise linear program:

$$\underset{(x,y,w)\in\Omega}{\text{minimize}}\; c^T x + d^T y + \rho\left[\sum_{i \in \widehat{\mathcal{I}}_y} y_i + \sum_{i \in \widehat{\mathcal{I}}_w} w_i + \sum_{i \in \widehat{\mathcal{I}}_=} \max(y_i, w_i)\right].$$

Moreover, if $\bar{z}^\rho$ is critical to (2.8) and feasible to the LPCC (1.1), then $\bar{z}^\rho$ is a weakly stationary solution of (1.1).

Proof. By definition, $\bar{z}^\rho$ is a critical point of the DC program (2.8) if and only if there exists $(0, \tilde{a}, \tilde{b})$ in $\partial f_2^{\rm PL}(\bar{z}^\rho)$ such that $(0, \tilde{a}, \tilde{b}) \in \partial f_1^{\rm PL}(\bar{z}^\rho) + \mathcal{N}(\bar{z}^\rho; \Omega)$. With $(a, b) \triangleq \rho^{-1}(\tilde{a}, \tilde{b})$, this implies that $\bar{z}^\rho$ is an optimal solution of the linear program:

$$\underset{(x,y,w)\in\Omega}{\text{minimize}}\; c^T x + d^T y - \rho\,\left[\,a^T y + b^T w\,\right].$$

By the definition of the separable function $f_2^{\rm PL}$, it is then not difficult to see that $\bar{z}^\rho$ is a critical point of (2.8) if and only if there exist $\xi_i \in [0, 1]$ for $i \in \widehat{\mathcal{I}}_=$ such that $\bar{z}^\rho$ is an optimal solution of the linear program:

$$\operatorname*{minimize}_{(x,y,w)\in\Omega} c^T x + d^T y + \rho \left\{ \sum_{i\in\widehat{\mathcal{I}}_y} y_i + \sum_{i\in\widehat{\mathcal{I}}_w} w_i + \sum_{i\in\widehat{\mathcal{I}}_=} [\,\xi_i\, y_i + (\,1-\xi_i\,)\, w_i\,] \right\}.$$

Based on this equivalence, we can now complete the proof of the proposition. Clearly statements (b) and (c) are equivalent. Suppose that (a) holds. Let $\xi_i \in [0,1]$ for $i \in \widehat{\mathcal{I}}_=$ be as given above. We then have for any $(x,y,w) \in \Omega$

$$\begin{aligned}
& c^T x + d^T y + \rho \left[\sum_{i\in\widehat{\mathcal{I}}_y} y_i + \sum_{i\in\widehat{\mathcal{I}}_w} w_i + \sum_{i\in\widehat{\mathcal{I}}_=} \max(y_i, w_i) \right] \\
&\quad \geq c^T x + d^T y + \rho \left\{ \sum_{i\in\widehat{\mathcal{I}}_y} y_i + \sum_{i\in\widehat{\mathcal{I}}_w} w_i + \sum_{i\in\widehat{\mathcal{I}}_=} [\,\xi_i\, y_i + (\,1-\xi_i\,)\, w_i\,] \right\} \\
&\quad \geq c^T \bar{x}^\rho + d^T \bar{y}^\rho + \rho \left\{ \sum_{i\in\widehat{\mathcal{I}}_y} \bar{y}_i^\rho + \sum_{i\in\widehat{\mathcal{I}}_w} \bar{w}_i^\rho + \sum_{i\in\widehat{\mathcal{I}}_=} [\,\xi_i\, \bar{y}_i^\rho + (\,1-\xi_i\,)\, \bar{w}_i^\rho\,] \right\} \\
&\quad = c^T \bar{x}^\rho + d^T \bar{y}^\rho + \rho \left[\sum_{i\in\widehat{\mathcal{I}}_y} \bar{y}_i^\rho + \sum_{i\in\widehat{\mathcal{I}}_w} \bar{w}_i^\rho + \sum_{i\in\widehat{\mathcal{I}}_=} \max(\,\bar{y}_i^\rho, \bar{w}_i^\rho\,) \right].
\end{aligned}$$

Thus (c) holds. Conversely, suppose that (c) holds. If $\bar{\theta}(x,y,w)$ denotes the objective function in the LP in part (c), we then have $0 \in \partial\bar{\theta}(\bar{z}^\rho) + \mathcal{N}(\bar{z}^\rho;\Omega)$. Since $\partial \max(s,t) = \{(\lambda, 1-\lambda) \mid \lambda \in [0,1]\}$ at a pair (s,t) with $s = t$, statement (a) follows readily from the subdifferential characterization of optimality $\bar{z}$ to the LP in part (c). Therefore statements (a), (b), and (c) are equivalent.

To complete the proof of the proposition, suppose that $\bar{z}^\rho$ is critical to (2.8) and feasible to the LPCC (1.1). Let (x,y,w) be a feasible triple to the restricted LP (RsLP) at $\bar{z}^\rho$. We then have from (c) that

$$\begin{aligned} c^Tx + d^Ty &= c^Tx + d^Ty + \rho\left[\sum_{i\in\widehat{\mathcal{I}}_y} y_i + \sum_{i\in\widehat{\mathcal{I}}_w} w_i + \sum_{i\in\widehat{\mathcal{I}}_=} \max(y_i, w_i)\right] \\ &\geq c^T\bar{x}^\rho + d^T\bar{y}^\rho + \rho\left[\sum_{i\in\widehat{\mathcal{I}}_y} \bar{y}_i^\rho + \sum_{i\in\widehat{\mathcal{I}}_w} \bar{w}_i^\rho + \sum_{i\in\widehat{\mathcal{I}}_=} \max(\bar{y}_i^\rho, \bar{w}_i^\rho)\right] \\ &= c^T\bar{x}^\rho + d^T\bar{y}^\rho, \end{aligned}$$

establishing that $\bar{z}^\rho$ is an optimal solution of RsLP. $\square$

2.1.4.3. The DCA applied to (2.8). In the application of the DCA to (2.8), the subproblem (2.6) becomes the following LP, which we will denote as LP(y^k, w^k, ξ^k):

$$\min_{(x,y,w)\in\Omega} c^Tx + d^Ty + \rho\left[\sum_{i\in\widehat{\mathcal{J}}_y^k} y_i + \sum_{i\in\widehat{\mathcal{J}}_w^k} w_i + \sum_{i\in\widehat{\mathcal{J}}_=^k} \left(\xi_i^k y_i + (1-\xi_i^k)w_i\right)\right], \tag{2.11}$$

where

$$\widehat{\mathcal{J}}_y^k \triangleq \widehat{\mathcal{I}}_y(y^k, w^k), \qquad \widehat{\mathcal{J}}_w^k \triangleq \widehat{\mathcal{I}}_w(y^k, w^k), \qquad \widehat{\mathcal{J}}_=^k \triangleq \widehat{\mathcal{I}}_=(y^k, w^k), \tag{2.12}$$

and $\xi_i \in [0,1]$ indicates which particular subgradient of each max-function in $f_2^{\rm PL}$ is chosen at a non-differentiable pair (y_i, w_i). We note that the current iterate (x^k, y^k, w^k) enters only in the partition of the variable indices (2.12), and that the absolute values of the variables do not matter. From this point of view, we may reinterpret the DCA as an active-set method, in which the algorithm solves LPs corresponding to different guesses of the optimal active set, or piece, of the LPCC. If we assume that ξ_i is chosen only from a finite set Ξ, then this observation shows that the DC algorithm will terminate after a finite number of iterations, because there exist only a finite number of possible constraint partitions $\widehat{\mathcal{J}}_y^k$, $\widehat{\mathcal{J}}_w^k$, $\widehat{\mathcal{J}}_=^k$, and the objective function is strictly monotonically decreasing. In our implementation, we restrict the choice of ξ_i's to be in the finite set $\Xi = \left\{0, \frac{1}{2}, 1\right\}$.

The following proposition shows that the final iterates exhibit stationarity properties for the LPCC.

Proposition 2.1.6. Assume that the LPCC is bounded below and that ξ_i^k in (2.11) is chosen from a finite set $\Xi \subseteq [0, 1]$. Then the DC algorithm, with termination tolerance $\varepsilon_{\text{tol}} = 0$, will terminate after a finite number of iterations. If z^k at termination is feasible to the LPCC (1.1), then z^k is a weakly stationary point.

Proof. As a consequence of the DCA, we already argued that the algorithm terminates after a finite number of iterations. The second assertion of the proposition follows from Proposition 2.1.5. □

Transitioning to the next two subsections where the penalty function ϕ is differentiable, we point out that the concept of a critical point of (2.5) coincides with that of a conventional stationary solution when the objective function is differentiable. In this case, we will use the latter terminology which involves elementary derivatives.

2.1.5. A bilinear penalty function in (y, w)

We next consider the bilinear penalty function:

$$\phi^{\text{BL}}(y, w) \triangleq y^T w \tag{2.13}$$

that leads to the following penalized bilinear programming formulation of the LPCC (1.1): for $\rho > 0$,

$$\underset{z \triangleq (x,y,w) \in \Omega}{\text{minimize}} \ \theta^{\text{BL}}(z) \triangleq c^T x + d^T y + \rho \sum_{i=1}^{n} y_i w_i. \tag{2.14}$$

Since the penalty function $\phi^{\text{BL}}(y, w)$ is differentiable, the stationary solutions of the (nonconvex) program (2.14) follow the standard definition; namely, $\bar{z} \in \Omega$ is stationary if and only if $(z - \bar{z})^T \nabla \theta^{\text{BL}}(\bar{z}) \geq 0$ for all $z \in \Omega$. The following result relates such a stationary solution, if feasible to the LPCC (1.1), to a strongly stationary solution of the LPCC.

Proposition 2.1.7. Let $\bar{z} \triangleq (\bar{x}, \bar{y}, \bar{w}) \in \Omega$ be given. The following two statements hold:

(a): If $\bar{z}$ is stationary for (2.14) for some $\rho > 0$ and is feasible to the LPCC (1.1), then $(\bar{x}, \bar{y}, \bar{w})$ is a strongly stationary point for (1.1).

(b): Conversely, if $\bar{z}$ is a strongly stationary point for (1.1), then $\bar{z}$ is a stationary point for (2.14) for all $\rho > 0$ sufficiently large.

Proof. Let $z = (x, y, w) \in \Omega$ be a feasible point for the relaxed LP (RxLP). The stationarity of $\bar{z}$ for (2.14) then yields

$$\begin{aligned}
0 &\leq (z - \bar{z})^T \nabla \theta^{\mathrm{BL}}(\bar{z}) \\
&= c^T(x - \bar{x}) + d^T(y - \bar{y}) + \rho \sum_{i=1}^{n} \left[\bar{w}_i (y_i - \bar{y}_i) + \bar{y}_i (w_i - \bar{w}_i) \right] \\
&= c^T(x - \bar{x}) + d^T(y - \bar{y}) + \rho \sum_{i \in \widehat{\mathcal{I}}_y} \bar{w}_i y_i + \rho \sum_{i \in \widehat{\mathcal{I}}_w} \bar{y}_i w_i, \\
&= c^T(x - \bar{x}) + d^T(y - \bar{y}).
\end{aligned}$$

The second equality is justified since $\bar{y}$ and $\bar{w}$ are complementary. The last equality follows from the fact that $\widehat{\mathcal{I}}_y = \mathcal{I}_y(\bar{y}, \bar{w})$ and $\widehat{\mathcal{I}}_w = \mathcal{I}_w(\bar{y}, \bar{w})$, given their definitions in (2.1) and (2.10), and because (x, y, w) is feasible to the LP (RxLP). Hence $\bar{z}$ is a minimizer of (RxLP) and (a) holds.

To prove (b), suppose $(\bar{x}, \bar{y}, \bar{w})$ is a strongly stationary point for (1.1). Let $(\bar{\lambda}, \mu^y, \mu^w)$ satisfy (2.2) along with the nonnegativity of μ_i^w and μ_i^y for all $i \in \mathcal{I}_0(\bar{y}, \bar{w})$. It remains to show that a multiplier $\bar{\mu}$ exists such that

$$\begin{aligned}
0 &= c - A^T \bar{\lambda} - M^T \bar{\mu} \\
0 &\leq \bar{y} \perp d + \rho \bar{w} - B^T \bar{\lambda} - N^T \bar{\mu} \geq 0 \\
0 &\leq \bar{w} \perp \rho \bar{y} + \bar{\mu} \geq 0 \\
0 &\leq \bar{\lambda} \perp A\bar{x} + B\bar{y} - f \geq 0.
\end{aligned} \tag{2.15}$$

We claim that $\bar{\mu} \triangleq \mu^w$ does the job. Since (2.2) gives $d = B^T\bar{\lambda} + N^T\mu^w + \mu^y$, it suffices to show that $\bar{y} \perp \mu^y + \rho\bar{w} \geq 0$ and $\bar{w} \perp \mu^w + \rho\bar{y} \geq 0$ for all $\rho > 0$ sufficiently large. Defining

$$\bar{\rho} \triangleq \max\left\{0, \left\{-\frac{\mu_i^y}{\bar{w}_i} \mid \bar{w}_i > 0\right\}, \left\{-\frac{\mu_i^w}{\bar{y}_i} \mid \bar{y}_i > 0\right\}\right\}$$

we see that both $\mu^y + \rho\bar{w}$ and $\mu^w + \rho\bar{y}$ are nonnegative for $\rho > \bar{\rho}$. Lastly, the remaining two orthogonality conditions $\bar{y} \perp \mu^y + \rho\bar{w}$ and $\bar{w} \perp \mu^w + \rho\bar{y}$ follow from the strong stationarity conditions of $(\bar{x}, \bar{y}, \bar{w})$. $\square$

While Proposition 2.1.4 has shown that the local minimizers of the penalized piecewise linear program (2.8) that are feasible to (1.1) are the same as the local minimizers of the LPCC (1.1), this equivalence of such minimizers between (2.14) and (1.1) is not addressed by Proposition 2.1.7. Borrowing an example from (Fang et al., 2012; Scheel and Scholtes, 2000) we show that a local (or even global) minimizer of (1.1) may not be obtainable from solutions of the bilinear penalty formulation for any fixed $\rho > 0$; thus, there are solutions to the original LPCC that are "elusive" from the penalized bilinear program.

Example 2.1.8. Consider

$$\begin{array}{ll}
\underset{(x,y,w)\in\mathbb{R}^3}{\text{minimize}} & -x + y + w \\
\text{subject to} & -x + 4y \geq 0 \\
 & -x + 4w \geq 0 \\
\text{and} & 0 \leq y \perp w \geq 0.
\end{array}$$

It is not difficult to see that $(x,y,w) = (0,0,0)$ is a global and therefore local minimizer for this LPCC. Consider the penalized problem

$$\begin{array}{ll} \underset{(x,y,w)\in\mathbb{R}^3}{\text{minimize}} & -x + y + w + \rho\, yw \\ \text{subject to} & -x + 4y \geq 0 \\ & -x + 4w \geq 0 \\ \text{and} & y, w \geq 0 \end{array}$$

for $\rho > 0$. With $y = w = x/4$, the objective function is equal to $-x/2 + \rho x^2/16$ which is negative when $0 < x < 8/\rho$. Thus $(x,y,w) = (0,0,0)$ is not locally minimizing the penalized bilinear program for any $\rho > 0$. $\square$

In (Le Thi and Dinh, 2011), the bilinear function ϕ^{BL} is written as a DC function by adding and subtracting the norm of the vectors y and w:

$$(2.16)\qquad \phi^{\mathrm{BL}}(y,w) \triangleq \underbrace{y^T w + \frac{\gamma}{2}\left(\|y\|_2^2 + \|w\|_2^2\right)}_{\triangleq\ \phi_+^{\mathrm{BL}_1}(y,w)} - \underbrace{\frac{\gamma}{2}\left(\|y\|_2^2 + \|w\|_2^2\right)}_{\triangleq\ \phi_-^{\mathrm{BL}_1}(y,w)},$$

where $\phi_+^{\mathrm{BL}_1}$ is convex if $\gamma > 1$. This results in the DC program (2.5) having $f_1^{\mathrm{BL}_1}(z) \triangleq c^T x + d^T y + \rho\phi_+^{\mathrm{BL}_1}(y,w)$ and $f_2^{\mathrm{BL}_1}(z) \triangleq \rho\phi_-^{\mathrm{BL}_1}(y,w)$. The subproblem (2.6) in the DCA applied to the resulting penalized DC program is a convex quadratic program:

$$(2.17)\qquad \underset{(x,y,w)\in\Omega}{\text{minimize}}\ c^T x + d^T y + \rho\left[y^T w + \frac{\gamma}{2}\left(\|\,y - y^k\,\|_2^2 + \|\,w - w^k\,\|_2^2\right)\right].$$

In addition, we propose an alternative DC representation of the bilinear function ϕ^{BL}:

$$\phi^{\mathrm{BL}}(y,w) \;=\; \underbrace{\frac{1}{4}\,\|y+w\|_2^2}_{\triangleq\,\phi_+^{\mathrm{BL}_2}(y,w)} \;-\; \underbrace{\frac{1}{4}\,\|y-w\|_2^2}_{\triangleq\,\phi_-^{\mathrm{BL}_2}(y,w)}\;. \tag{2.18}$$

This DC decomposition has been used in (Mitchell et al., 2012) albeit not as a DC objective. The advantage of the DC pair $(\phi_+^{\mathrm{BL}_2}, \phi_-^{\mathrm{BL}_2})$ is that it does not depend on a parameter γ that might need to be adjusted for good performance. The subproblem (2.6) in the DCA applied to the resulting penalized DC program is the following convex quadratic program:

$$\underset{(x,y,w)\in\Omega}{\text{minimize}}\; c^T x + d^T y + \frac{\rho}{4}\,\|y+w\|_2^2 - \frac{\rho}{2}\,\left(y^k - w^k\right)^T(y-w)\,, \tag{2.19}$$

which is different from (2.17). From Proposition 2.1.7, we may deduce that the DCA applied to the penalized bilinear formulation (2.8) for a given $\rho > 0$ will compute a strongly stationary solution of the LPCC (1.1) if the computed solution satisfies the complementarity condition.

2.1.6. Quadratic penalty function in (x, y)

An equivalent formulation of the LPCC (1.1) is obtained by eliminating the variable w. This leads to the following formulation:

$$\begin{array}{ll} \underset{x,y}{\text{minimize}} & c^T x + d^T y \\ \text{subject to} & Ax + By \;\geq\; f \\ \text{and} & 0 \;\leq\; y \;\perp\; Mx + Ny + q \;\geq\; 0. \end{array}$$

Using the penalty function

$$\phi^{-w}(x,y) \triangleq y^T(Mx+Ny+q) = q^Ty + \frac{1}{2}\begin{pmatrix} x \\ y \end{pmatrix}^T \underbrace{\begin{bmatrix} 0 & M^T \\ M & N+N^T \end{bmatrix}}_{\triangleq D} \begin{pmatrix} x \\ y \end{pmatrix},$$

where D is a symmetric indefinite matrix, we obtain the formulation

$$\underset{(x,y)\in\Omega_{xy}}{\text{minimize}}\ c^Tx + d^Ty + \rho\,\phi^{-w}(x,y) \tag{2.20}$$

with $\Omega_{xy} \triangleq \{(x,y) \in \mathbb{R}^{t\times n} \mid Ax + By \geq f,\ Mx + Ny + q \geq 0,\ y \geq 0\}$. Similar to Proposition 2.1.7, we have the relationship between stationary points.

Proposition 2.1.9. Let $(\bar{x},\bar{y}) \in \Omega_{xy}$ be given. The following two statements hold:

(a): Suppose $(\bar{x},\bar{y})$ is a stationary point for (2.20) and let $\bar{w} \triangleq q + M\bar{x} + N\bar{y}$. If $(\bar{x},\bar{y},\bar{w})$ is a feasible solution of the LPCC (1.1) for some $\rho > 0$, then $(\bar{x},\bar{y},\bar{w})$ is a strongly stationary solution for (1.1).

(b): Conversely, if $(\bar{x},\bar{y},\bar{w})$ is a strongly stationary point for (1.1), then $(\bar{x},\bar{y})$ is stationary for (2.20) for all $\rho > 0$ sufficiently large.

Proof. (a) If $(\bar{x},\bar{y}) \in \Omega_{xy}$ is a stationary point for (2.20), then constraint multipliers $\bar{\lambda}$ and $\bar{\mu}$ exist such that

$$
\begin{array}{rcl}
0 & = & c - A^T\bar{\lambda} - M^T(\,\bar{\mu} - \rho\,\bar{y}\,) \\
0 & \leq & \bar{y} \perp d + \rho\,[\,q + M\bar{x} + N\bar{y}\,] - B^T\bar{\lambda} - N^T(\,\bar{\mu} - \rho\,\bar{y}\,) \geq 0 \\
0 & \leq & \bar{\lambda} \perp A\bar{x} + B\bar{y} - f \geq 0 \\
0 & \leq & \bar{\mu} \perp M\bar{x} + N\bar{y} + q \geq 0.
\end{array}
\tag{2.21}
$$

By letting $\mu^w \triangleq \bar{\mu} - \rho\bar{y}$ and $\mu^y \triangleq d - B^T\bar{\lambda} - N^T\mu^w$, we now show that (2.2) holds. The first two conditions in (2.21) can be written as

$$
\begin{array}{rcl}
c & = & A^T\bar{\lambda} + M^T\mu^w \\
d & = & B^T\bar{\lambda} + N^T\mu^w + \mu^y \\
0 & \leq & \bar{y} \perp \rho\,\bar{w} + \mu^y \geq 0.
\end{array}
$$

Because $\bar{y}$ and $\bar{w}$ are feasible for the LPCC (1.1), we have $\bar{y} \circ \bar{w} = 0$, and hence $\bar{y} \circ \mu^y = \bar{y} \circ (\rho\,\bar{w} + \mu^y) = 0$. Furthermore, since $\bar{w}_i = 0$ for $i \in \mathcal{I}_0(\bar{y},\bar{w})$, we also have $\mu^y_i \geq 0$ for $i \in \mathcal{I}_0(\bar{y},\bar{w})$ from the last line above. Similarly, because $\bar{y} \circ \bar{w} = 0$ from (1.1), conditions (2.21) imply $\mu^w \circ \bar{w} = \bar{\mu} \circ (\,M\bar{x} + N\bar{y} + q\,) = 0$. Finally, since $\bar{y}_i = 0$ for $i \in \mathcal{I}_0(\bar{y},\bar{w})$, we also have $\mu^w_i = \bar{\mu}_i \geq 0$ for $i \in \mathcal{I}_0(\bar{y},\bar{w})$ by the definition of μ^w.

(b) Conversely, suppose $(\bar{x},\bar{y})$ is strongly stationary for (1.1). Let $(\bar{\lambda},\mu^y,\mu^w)$ satisfy (2.2) along with the nonnegativity of μ^w_i and μ^y_i for all $i \in \mathcal{I}_0(\bar{y},\bar{w})$. We claim that by defining $\bar{\mu} \triangleq \mu^w + \rho\bar{y}$, (2.21) holds for all $\rho > 0$ sufficiently large. In turn, it suffices to show the second and fourth complementarity conditions in (2.21) for all $\rho > 0$ sufficiently large. Since $d = B^T\bar{\lambda} + N^T\mu^w + \mu^y$, we have

$$d + \rho\,(q + M\bar{x} + N\bar{y}) - B^T\bar{\lambda} - N^T(\bar{\mu} - \rho\,\bar{y}) \;=\; \rho\,(q + M\bar{x} + N\bar{y}) + \mu^y =\; \rho\,\bar{w} + \mu^y;$$

hence the second complementarity condition in (2.21) holds for all $\rho > 0$ because $\bar{y}$ and $\bar{w}$ are feasible for the LPCC (1.1) and orthogonality of y and μ^y comes from stationarity. For the fourth complementarity condition to hold, choose $\rho > 0$ sufficiently large so that $\mu^w_i + \rho\bar{y}_i \geq 0$ if $\bar{y}_i > 0$. For the remaining indices with $\bar{y}_i = 0$ we have two cases: (i) If $\bar{w}_i > 0$, then $\bar{\mu}_i = \mu^w_i = 0$ by complementarity; (ii) if $\bar{w}_i = 0$, then $i \in= \mathcal{I}_0(\bar{y}, \bar{w})$ and $\bar{\mu}_i = \mu^w_i \geq 0$, because $(\bar{x}, \bar{y})$ is strongly stationary. $\square$

The penalty function ϕ^{-w} can be rewritten as DC function

$$\phi^{-w}(x,y) = \phi^{-w}_{+}(x,y) - \phi^{-w}_{-}(x,y), \tag{2.22}$$

where

$$\phi^{-w}_{+}(x,y) = \begin{pmatrix} x \\ y \end{pmatrix}^T D \begin{pmatrix} x \\ y \end{pmatrix} + 2q^Ty + \gamma\,\left(\,\|x\|_2^2 + \|y\|_2^2\,\right) \text{ and}$$

$$\phi^{-w}_{-}(x,y) = \frac{\gamma}{2}\,\left(\,\|x\|_2^2 + \|y\|_2^2\,\right).$$

The function ϕ^{-w}_{+} is convex if γ is larger than the absolute value of smallest eigenvalue of the symmetric indefinite matrix D. The resulting subproblems in DCA are of the form:

$$\underset{(x,y)\in\Omega_{xy}}{\text{minimize}}\;\; \left\{ c^Tx + d^Ty + \frac{\rho}{2}\,\phi^{-w}_{+}(x,y) - \,\rho\,\gamma\;\left[\,(\,y^k\,)^Ty + (\,x^k\,)^Tx\,\right] \right\}. \tag{2.23}$$

In summary, while the theoretical properties of the quadratic penalty ϕ^{-w} without the w-variable are the same as those of the bilinear penalty ϕ^{BL}, the respective subproblems in the DCA differ, and a different sequence of iterates is generated.

2.2. Enhancements of the DCA

The straightforward application of the DCA to the penalty formulations exhibits some inefficiencies. In the case of the piecewise linear function (Section 2.1.4.1), the method might terminate at a point that is weakly stationary but not strongly stationary and has an inferior objective value. When a quadratic penalty function (Sections 2.1.5 and 2.1.6) is used, we observed that the method could suffer from the slow (linear) convergence inherent to first-order methods, because the concave part of the DC function is approximated by a linear function. In what follows, we propose modifications to address these specific shortcomings.

2.2.1. Improved DCA for the piecewise linear penalty

In subproblem (2.11) with the piecewise linear penalty function, a parameter $\xi_i^k \in [0,1]$ has to be chosen for $i \in \mathcal{J}_{=}^k$. To avoid any bias for either one of the two complementary variables, we choose $\xi_i^k = \frac{1}{2}$ in the implementation unless otherwise mentioned. However, we noticed that in some instances improving directions existed at the points at which the DCA terminated and that were missed by the method. In the following example, the LP subproblem (2.11) produces a zero step at an iterate that is not a strongly stationary point for the LPCC.

Example 2.2.1. Consider the problem

$$\begin{array}{ll} \underset{x,y,w}{\text{minimize}} & -x + 2y \\ \text{subject to} & -x \ \geq \ -10 \\ & -y \ \geq \ -10 \\ & x - w \ = \ 0 \\ \text{and} & 0 \ \leq \ y \ \perp \ w \ \geq \ 0. \end{array}$$

The unique strongly stationary point of this problem lies in $z^* = (x^*, y^*, w^*) = (10, 0, 10)$. Applying the DCA for the piecewise linear penalty formulation, with starting point $(x, y, w) = (5, 10, 5)$ and penalization parameter $\rho > 2$, the first linear subproblem solved has a negative objective gradient equal to d_0. Therefore, the DCA will reach the point $(0, 0, 0)$ at the first iteration. If, at the second subproblem, the choice is $\xi^2 = (\frac{1}{2}, \frac{1}{2})$, the negative gradient becomes d_1 and, hence, the solution will not move and the method will stop at this weakly stationary point. Figure 2.1 illustrates how these two iterates are produced.

However, if a different element of the subdifferential $\partial\{-\min(y_i, w_i)\}$, corresponding to $\xi^2 = (0, 1)$, is chosen, the DC subproblem produces z^* as the next iterate, and the algorithm terminates at a strongly stationary point. $\square$

The following lemma shows that, by choosing $\xi_i \in \{0, 1\}$ appropriately, the DCA will escape weakly stationary points that are not locally optimal.

Lemma 2.2.2. Suppose the DCA applied to (2.8) terminates in a weakly stationary point $z^* = (x^*, y^*, w^*)$ that is not a local minimum for the LPCC. Then there exists a

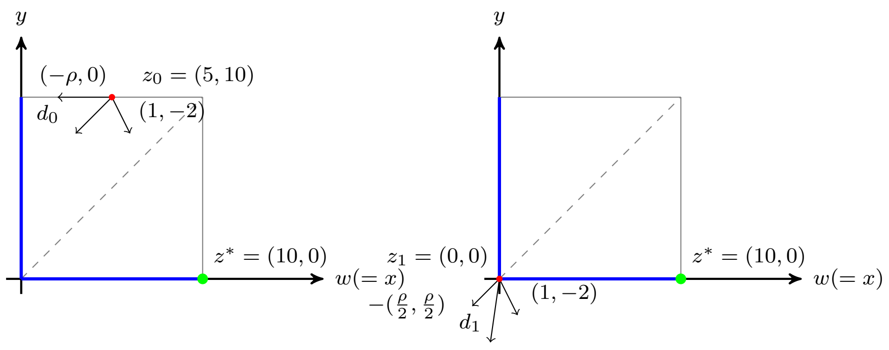


Figure 2.1. DCA for the piecewise linear penalty function converges to a weakly stationary point.

subset $\mathcal{J} \subseteq \mathcal{I}_0(y^*, w^*)$ so that the DCA continues with a non-zero step if it is resumed with $\xi_i = 1$ for $i \in \mathcal{J}$ and $\xi_i = 0$ for $i \in \mathcal{I}_0(y^*, w^*) \setminus \mathcal{J}$.

Proof. If $z^* = (x^*, y^*, w^*)$ is not a local minima for the LPCC then, by Proposition (2.1.2), it is not B-stationary. That is, there exists a subset $\mathcal{J} \subseteq \mathcal{I}_0(y^*, w^*)$ such that z^* is not a solution for LP($\mathcal{J}$). Hence, there exists a decreasing direction $\delta = (\delta^x, \delta^y, \delta^w) \in \Omega_0$ such that $c^T\delta^x + d^T\delta^y < 0$ and

$$\begin{aligned}
\delta_i^y &= 0 & i &\in \mathcal{I}_y(y^*, w^*) \\
\delta_i^w &= 0 & i &\in \mathcal{I}_w(y^*, w^*) \\
\delta_i^y &= 0 \leq \delta_i^w, & i &\in \mathcal{J} \\
\text{and} \quad \delta_i^y &\geq 0 = \delta_i^w, & i &\in \mathcal{I}_0(y^*, w^*) \setminus \mathcal{J},
\end{aligned}$$

where $\Omega_0 = \{(x,y,w) \in \mathbb{R}^t \times \mathbb{R}^{2n} \mid Ax + By \geq 0,\, Mx + Ny = w\}$. Let $\xi_i = 1$ for $i \in \mathcal{J}$ and $\xi_i = 0$ for $i \in \mathcal{I}_0(y^*, w^*) \setminus \mathcal{J}$. The resulting subproblem in the DCA is $\mathrm{LP}_\rho(\mathcal{J})$, defined in (2.9). Since $z^* + \alpha\delta$ is feasible for $\mathrm{LP}_\rho(\mathcal{J})$ for a sufficiently small $\alpha > 0$ and $c^T x^* + d^T y^* + \rho y^{*T} w^* = c^T x^* + d^T y^* > c^T(x^* + \alpha\delta^x) + d^T(y^* + \alpha\delta^y) = c^T(x^* + \alpha\delta^x) + d^T(y^* + \alpha\delta^y) + (y^* + \alpha\delta^y)^T(w^* + \alpha\delta^w)$, we have that z^* is not an optimal solution of $\mathrm{LP}_\rho(\mathcal{J})$. Therefore the DCA reaches a new iterate z^{k+1} with better objective function than z^*, and the method continues. $\square$

Finding the subset $\mathcal{J}$ in Lemma 2.2.2 to escape a non-optimal weakly stationary point is a combinatorial problem with worst-case exponential complexity. To generate a candidate set $\mathcal{J}$ quickly, we employ a heuristic that utilizes the multipliers $\bar{\mu}^{y,k}$ and $\bar{\mu}^{w,k}$ in the LP subproblem (2.11) for the non-negativity restrictions of y and w, respectively. In order to recover the quantities that correspond to the multipliers μ^y and μ^w in the stationarity conditions (2.2) for the LPCC, we need to subtract the influence of the penalty term and compute for each index $i \in \mathcal{I}_0(y^{k+1}, w^{k+1})$

$$
\begin{aligned}
\mu_i^{y,k} &= \bar{\mu}_i^{y,k} - \rho & \mu_i^{w,k} &= \bar{\mu}_i^{w,k} & &\text{if } i \in \widehat{\mathcal{J}}_y^k,\\
\mu_i^{y,k} &= \bar{\mu}_i^{y,k} & \mu_i^{w,k} &= \bar{\mu}_i^{w,k} - \rho & &\text{if } i \in \widehat{\mathcal{J}}_w^k,\\
\mu_i^{y,k} &= \bar{\mu}_i^{y,k} - \rho\xi_i^k & \mu_i^{w,k} &= \bar{\mu}_i^{w,k} - \rho(1 - \xi_i^k) & &\text{if } i \in \widehat{\mathcal{J}}_=^k.
\end{aligned}
$$

If $z^{k+1} = (x^{k+1}, y^{k+1}, w^{k+1})$ is feasible for the LPCC (1.1) and $\mu_i^{y,k} \geq 0$ and $\mu_i^{w,k} \geq 0$ for all $i \in \mathcal{I}_0(y^{k+1}, w^{k+1})$, then z^{k+1} is a strongly stationary point for the LPCC (1.1). Otherwise, we use the adjusted multipliers to choose the subgradient to restart the DCA: For each

$i \in \mathcal{I}_0(y^{k+1}, w^{k+1})$ we set

$$\xi_i^{k+1} = \begin{cases} 1 & \text{if } \mu_i^{y,k} < \mu_i^{w,k} - \varepsilon_\xi \\ \frac{1}{2} & \text{if } \left| \mu_i^{y,k} - \mu_i^{w,k} \right| \leq \varepsilon_\xi \\ 0 & \text{if } \mu_i^{y,k} > \mu_i^{w,k} + \varepsilon_\xi \end{cases} \tag{2.24}$$

and $\xi_i^{k+1} = \frac{1}{2}$ if $i \in \widehat{\mathcal{J}}_{=}^{k+1} \setminus \mathcal{I}_0(y^{k+1}, w^{k+1})$. This choice of ξ_i^{k+1} is within the specification of the DCA and so the convergence result, Proposition 2.1.6, remains valid for this version of the method. The small tolerance ε_ξ (chosen to be 10^{-8} in our experiments) accounts for potential numerical error of the subproblem solution. If the resulting step reduces the objective function, we resume the regular DCA. Otherwise, if the resulting step does not reduce the objective function, in our implementation, we terminate the algorithm at the weakly stationary point instead of trying to generate another candidate set $\mathcal{J}$, based on some other heuristic. However, for the test problems in Section 2.3, we found that the simple heuristic above was always able to escape a non-optimal point.

2.2.2. Improved DCA for the quadratic penalty function in (y, w)

In the standard DCA with the quadratic penalty functions, the nonlinear concave part of the quadratic penalty problem is approximated by a linear function. We observed that this can lead to slow convergence if the approximation is applied straightforwardly to the given quadratic function. To accelerate the method, the improved DCA described next fixes variables to zero for which complementarity has already been achieved, and deletes the penalty terms for those components in the subproblem objective. In the extreme case, when one variable in all complementarities has been fixed, the subproblem becomes an LP, and the next iterate jumps immediately to the optimal solution in that piece. This saves many iterations compared to the original method whose convergence may be slowed by the penalty term. Even when not all complementarities have been fixed, the enhanced version has to resolve fewer complementarities with a linear approximation of the quadratic penalty function than the original version, and faster convergence can be expected.

We note that the subproblems (2.17) and (2.19) can be rewritten in a unified form with each complementarity having its own quadratic penalty function:

$$\min_{(x,y,w)\in\Omega} \; c^T x + d^T y + \rho \sum_{i=1}^{n} \psi_i^k(y_i, w_i).$$

By construction of the DC subproblem (2.6), we have that the gradient of the original DC function and that of its convex approximation coincide at the current iterate; i.e.

$$\nabla \psi_i^k(y_i^k, w_i^k) = \nabla \phi_i(y_i^k, w_i^k) = \begin{pmatrix} w_i^k \\ y_i^k \end{pmatrix} \tag{2.25}$$

where $\phi_i(y_i, w_i) = y_i \cdot w_i$. In our proposed variation of the DCA, we maintain the set $\mathcal{F}_y^k$ of indices i for which y_i^k is fixed at zero in iteration k. The index set $\mathcal{F}_w^k$ is defined analogously for w^k, and we let $\mathcal{F}^k = \mathcal{F}_y^k \cup \mathcal{F}_w^k$. These sets try to predict which sides of the complementarities are active (zero) at the solution. The modified subproblem that delivers the next iterate is then defined as

$$\begin{array}{ll} \underset{(x,y,w)\in\Omega}{\text{minimize}} & c^T x + d^T y + \rho \sum_{i \notin \mathcal{F}^k} \psi_i^k(y_i, w_i) \\ \text{subject to} & y_i = 0 \text{ for } i \in \mathcal{F}_y^k \\ \text{and} & w_i = 0 \text{ for } i \in \mathcal{F}_w^k. \end{array} \tag{2.26}$$

Note that the more variables become fixed in $\mathcal{F}^k$, the fewer free variables there are in the subproblem, leading to a reduction in computation time. The sets $\mathcal{F}_y^k$ and $\mathcal{F}_w^k$ are updated throughout the course of the algorithm. An index i enters the set $\mathcal{F}_y^{k+1}$, if $i \notin \mathcal{F}_y^k$ and $y_i^{k+1} = 0$; i.e., the variable y_i has just become zero. The update rule for w is similar.

We must also account for the possibility that our prediction is incorrect and a variable has to be released because it might be nonzero at the optimal solution. Recall that, ultimately, we want to solve the penalty formulation (2.3) of the LPCC with $\phi(y,w) = \phi^{\text{BL}}(y,w) = y^T w$ defined in (2.13). The first-order optimality conditions for (2.14) are

$$\begin{aligned}
& A^T\lambda^f + M^T\lambda^q && = \; c \\
& B^T\lambda^f + N^T\lambda^q + \mu^y && = \; d + \rho\, w \\
& -\lambda^q + \mu^w && = \; \rho\, y \\
& 0 \le \lambda^f \perp Ax + By - f && \ge \; 0 \\
& 0 \le y \perp \mu^y && \ge \; 0 \\
& 0 \le w \perp \mu^w && \ge \; 0 \\
& (x, y, w) && \in \; \Omega.
\end{aligned} \tag{2.27}$$

On the other hand, the first-order optimality conditions for (2.26) can be written as

$$
\begin{array}{lcl}
A^T\lambda^f + M^T\lambda^q & = & c \\
B^T\lambda^f + N^T\lambda^q + \bar{\mu}^y & = & d + \dfrac{\rho}{2}\displaystyle\sum_{i\notin\mathcal{F}^k} \nabla_y \psi_i^k(y_i, w_i)e_i \\
-\lambda^q + \bar{\mu}^w & = & \dfrac{\rho}{2}\displaystyle\sum_{i\notin\mathcal{F}^k} \nabla_w \psi_i^k(y_i, w_i)e_i \\
0 \le \lambda^f \perp Ax + By - f & \ge & 0 \\
y \circ \bar{\mu}^y & = & 0 \\
w \circ \bar{\mu}^w & = & 0 \\
y_i & = & 0 \text{ for } i \in \mathcal{F}_y^k \\
w_i & = & 0 \text{ for } i \in \mathcal{F}_w^k \\
\bar{\mu}^y & \ge & 0 \text{ for } i \notin \mathcal{F}_y^k \\
\bar{\mu}^w & \ge & 0 \text{ for } i \notin \mathcal{F}_w^k \\
(x, y, w) & \in & \Omega.
\end{array}
\tag{2.28}
$$

where $e_i \in \mathbb{R}^n$ denotes the i^{th} coordinate vector.

Suppose that the new iteratate $(x^{k+1}, y^{k+1}, w^{k+1})$ of our variation of the DCA, computed as the optimal solution of the subproblem (2.26), is identical to the current iterate (x^k, y^k, w^k). Usually, the method would terminate now. To verify that this is indeed a minimizer for the penalty function (2.14) without the restrictions imposed by the sets $\mathcal{F}_y^k$ and $\mathcal{F}_w^k$, let $(x^{k+1}, y^{k+1}, w^{k+1}, \lambda^f, \lambda^q, \bar{\mu}^y, \bar{\mu}^w)$ be a primal-dual solution of the subproblem

(2.26) satisfying the optimality conditions (2.28), and define

$$\mu_i^y = \bar{\mu}_i^y \text{ if } i \notin \mathcal{F}_y^k \qquad\qquad \mu_i^y = \bar{\mu}_i^y + \rho\, w_i^{k+1} \text{ if } i \in \mathcal{F}_y^k$$

$$\mu_i^w = \bar{\mu}_i^w \text{ if } i \notin \mathcal{F}_w^k \qquad\qquad \mu_i^w = \bar{\mu}_i^w + \rho\, y_i^{k+1} \text{ if } i \in \mathcal{F}_w^k.$$

Using observation (2.25), it is easy to see that then $(x^{k+1}, y^{k+1}, w^{k+1}, \lambda^f, \lambda^q, \mu^y, \mu^w)$ satisfies the first-order optimality conditions (2.27) for the penalty problem (2.3), as long as

$$\mu_i^y \geq 0 \text{ if } i \in \mathcal{F}_y^k \quad \text{and} \quad \mu_i^w \geq 0 \text{ if } i \in \mathcal{F}_w^k. \tag{2.29}$$

In that case, our method terminates at a local minimizer of the penalty function. If condition (2.29) does not hold, then there is a good chance that a multiplier for a fixed variable in $\mathcal{F}_y^k$ or $\mathcal{F}_w^k$ would have a negative sign at the solution. The variable should therefore not be forced to be zero. We use this observation to decide when to release a variable from the sets $\mathcal{F}_y^k$ and $\mathcal{F}_w^k$ of fixed components. This idea is implemented in Step 7 of the overall method summarized in Algorithm 2.

2.3. Computational Results

This section explores the practical performance of the DC decompositions described earlier for three different problem sets. The results confirm that the enhancements proposed in Section 2.2 contribute in one of two ways for the basic DCA applied to these decompositions: (a) for the piecewise linear penalty function, solutions with better objective values are found, or (b) for the bilinear penalty function, termination occurs faster, thus speeding up the algorithm. In short, the enhanced DC methods provide a competitive option for solving LPCCs.

Algorithm 2 Accelerated DCA for the quadratic penalty function in (y, w)

```
1: Choose termination tolerance $\varepsilon_{\text{tol}} > 0$.
2: Let $(x^1, y^1, w^1) \in \Omega$, $k \leftarrow 0$, $\mathcal{F}_y^1 = \emptyset$, and $\mathcal{F}_w^1 = \emptyset$.
3: repeat
4:     Set $k \leftarrow k + 1$.
5:     Solve subproblem (2.26).
6:     Let $(x^{k+1}, y^{k+1}, w^{k+1})$ be an optimal solution of (2.26), and let $\bar{\mu}_i^y$ and $\bar{\mu}_i^w$ be the multipliers in (2.28).
7:     Update the set of fixed variables:
       $$\mathcal{F}_y^{k+1} = \left\{i \notin \mathcal{F}_y^k : y_i^{k+1} = 0\right\} \cup \left\{i \in \mathcal{F}_y^k : \bar{\mu}_i^y + \rho\, w_i^{k+1} \geq 0\right\}$$
       $$\mathcal{F}_w^{k+1} = \left\{i \notin \mathcal{F}_w^k : w_i^{k+1} = 0\right\} \cup \left\{i \in \mathcal{F}_w^k : \bar{\mu}_i^w + \rho\, y_i^{k+1} \geq 0\right\}$$
8: until termination criteria satisfied
9: Return $(x^{k+1}, y^{k+1}, w^{k+1})$.
```

The experiments were performed on a Linux workstation with 3.10GHz Xeon processors using 20 cores (40 cores hyperthreaded) and 256GB RAM. The DCA and its enhancements were implemented in MATLAB (R2014a). To compare the performance of our methods with efficient alternatives, we solved all instances also with the FILTER software, a sophisticated implementation of an active-set SQP method for nonlinear programs with special features to solve MPCCs (Fletcher* and Leyffer, 2004; Fletcher and Leyffer, 2002). FILTER was called from AMPL (Fourer et al., 1990), with AMPL's presolve disabled. We have considered the KNITRO nonlinear programming solver which has the capability to handle complementarity constraints with a penalty function (Leyffer et al., 2006); but we do not describe its performance here since it was unable to solve many of the test problems. All these approaches only attempt to find local optima. To assess the solution quality, we also consider the MILP reformulation of the LPCC,

$$
\begin{array}{lrcl}
\underset{x,y,w,z}{\text{minimize}} & c^T x + d^T y & & \\
\text{subject to} & Ax + By & \geq & f \\
& Mx + Ny + q & = & w \\
& 0 \leq y & \leq & u^y \circ z \\
& 0 \leq w & \leq & u^w \circ (1 - z) \\
\text{and} & z \in \{0,1\}^n, & &
\end{array}
\tag{2.30}
$$

that can compute the global optimum of the instances. Here, u^y and u^w are explicit upper bounds for y and w. In the case where such upper bounds were not explicitly available, a large value (1,000) was used. A time limit of 900 seconds (wall clock time) was imposed on the MILP solver, using 32 threads. This corresponds roughly to up to 8 hours of CPU time. If no optimal solution was found after this time, the solver returned the best incumbent. All LPs and MILPs in our experiments were solved with CPLEX 12.6.2 using default settings.

Three sets of problems were tested: (a) Linear Complementarity Problems (LCP) instances from (Le Thi and Dinh, 2011) (Section 2.3.1), (b) linearizations of problems from the MacMPEC library (Leyffer, 2000) (Section 2.3.2), and (c) random instances of the inverse quadratic problem (Section 2.3.3). For easiness of notation, decompositions (2.8), (2.16), (2.18), and (2.22) are abbreviated as PL, BL1, BL2, and B-w, respectively. For each decomposition, two different starting points were tried. The first one, denoted by E, sets $y = w = \mathbf{1}$, and the other, called R, uses the solution to the LP relaxation of the LPCC, which is defined as (2.3) with $\rho = 0$. Note that the choice of the initial value of x is irrelevant, because it does not appear in $f_2(z)$ of the decomposition (2.4) and has no effect on the subproblem (2.6). Finally, enhanced methods are marked with a star (*). For example, BL2-E* refers to the enhanced DC approach based on decomposition (2.18), with starting point on $y = w = \mathbf{1}$. For decomposition (2.16), the values for γ are $\gamma_1 = 1.01$ and $\gamma_2 = 2$. Decomposition (2.22) uses $\gamma = \max\{0, -\lambda_{\min}\} + 0.01$, where $\lambda_{\min}$ is the smallest eigenvalue of matrix D defined in Section 2.1.6.

The implemented strategy for the penalization parameter ρ is given in Algorithm 3. It starts with a small value (namely, 1 in our experiments) and solves the penalty problem (2.3) with a termination tolerance of $\varepsilon_{\text{tol}} = 10^{-8}$. If the solution is already considered complementary, i.e. the infeasibility measure $V_\perp(y, w) \triangleq \max_i \{\min\{y_i, w_i\}\}$ is below a tolerance of $\varepsilon_\perp = 10^{-8}$, increasing ρ would have no effect (see Propositions 2.1.4, 2.1.7, and 2.1.9), and therefore we stop in Step 4. Otherwise, we increase ρ by a constant factor (in our case, 10) and solve the DC problem with the new ρ. If ρ reaches a maximum allowed value of 10^9, the algorithm terminates in Step 9.

We distinguish the following outcomes of the algorithm. Counting one subproblem solution as one iteration, we terminate with "Maximum number of iterations reached" when the iteration limit of 100 subproblem solves is exceeded. The algorithm stops with "Successful termination" if Algorithm 3 terminates because the complementarity tolerance in Step 4 is satisfied. However, there are also cases in which Algorithm 3 terminates because ρ exceeded $\bar{\rho}$ in Step 9, which we label as "Maximum value of penalty parameter reached." The latter case deserves closer examination. It is the expected outcome when

Algorithm 3 Overall algorithm with increasing penalty parameter

```
1: Choose starting penalization parameter $\rho > 0$, increase factor $\pi > 1$, upper bound $\bar{\rho}$,
   termination tolerance $\varepsilon_{\text{tol}} > 0$, and complementarity tolerance $\varepsilon_{\perp} > 0$.
2: repeat
3:     Solve DC problem (2.3) using the DC algorithm with current $\rho$ and tolerance $\varepsilon_{\text{tol}}$.
   Obtain solution $(x, y, w) \in \Omega$.
4:     if $V_{\perp}(y, w) \leq \varepsilon_{\perp}$ then
5:         Exit loop
6:     else
7:         Update $\rho := \pi\rho$
8: until $\rho \geq \bar{\rho}$
9: Return $(x, y, w)$.
```

the problem is (locally) infeasible. However, we frequently also observed the following situation in our experiments: The inner DC algorithm in Step 3 of Algorithm 3 terminates because the penalized objective function is not changing by more than $\varepsilon_{\text{tol}} = 10^{-8}$ between iterations, but the feasibility test in Step 4 is not satisfied, even though $V_{\perp}(y, w)$ is quite small and it appears that the iterates are close to a stationary point. The method then repeatedly increases ρ, taking only one step per penalty parameter value, until ρ reaches its limit. We will examine this situation in more detail in Section 2.3.2. For now, we will count a run as converged, if the method does not run out of iterations, and if the infeasibility measure $V_{\perp}(y, w)$ is at most $\varepsilon_{\text{feas}}$, where $\varepsilon_{\text{feas}} = 10^{-5}$ unless otherwise specified.

2.3.1. Linear complementarity problem instances

The LCP consists in finding a vector $x \in \mathbb{R}^n$ which satisfies

$$0 \leq x \perp \tilde{A}x - \tilde{b} \geq 0,$$

where $\tilde{A} \in \mathbb{R}^{n \times n}$ and $\tilde{b} \in \mathbb{R}^n$ are given. We reformulate the LCP as an LPCC by setting c, d, M, A, B and f to zero and letting $N = \tilde{A}$ and $q = -\tilde{b}$. Notice that this is a feasibility problem, so reaching strongly stationary points here has no special meaning. Therefore, results for the enhanced piecewise linear penalty function are not reported. In (Le Thi and Dinh, 2011), the authors proposed the solution of LCPs with the DC algorithm, decomposing different penalization functions for the complementarity term similar to those discussed here. Their methods DCA1, DCA2, and DCA4, correspond to B-w, PL, and BL1. The reference also contains a further penalty formulation, DCA3, which was shown therein not to perform well and so we did not include it in our approach for solving LPCCs.

In this first set of numerical experiments, we solved each of the twelve LCP instances given in (Le Thi and Dinh, 2011). The first six problems have dimensions $n \in \{2, 3, 4\}$. The remaining instances are scalable, and we chose $n = 1,000$, the largest size reported in this reference. Only the starting point E was used, since the LP relaxation is meaningless when no objective is given.

Similar to Le Thi and Dinh (2011), most methods solve all the instances. The only exception is BL2-E in test case 5, where the final infeasibility is violated only slightly with $1.2 \cdot 10^{-5} > \varepsilon_{\text{feas}}$. Table 2.1 displays the geometric averages of CPU times and iterations for all methods aggregated over the six scalable problems. We used geometric

Table 2.1. Geometric averages of computation times (in seconds) and iterations for the large LCP instances, including the original ("Orig") and enhanced ("Enhcd") variants of the bilinear decompositions.

	Average time		Average iterations	
Method	Orig	Enhcd	Orig	Enhcd
PL-E	0.068	-	2.000	-
BL1-E-γ=1.01	4.906	1.099	8.497	3.533
BL1-E-γ=2	6.539	1.369	11.671	4.451
BL2-E	2.156	0.607	6.094	2.289
B-w-E	0.450	-	2.140	-
MILP	0.164	-	-	-
FILTER	0.281	-	-	-

averages because these are less biased towards the more time-consuming instances than the arithmetic averages. We excluded the small instances because their solution time was considered negligible (less than 0.15 seconds).

The piecewise linear method is clearly the fastest by an order of magnitude, because it requires only two LP solutions per problem. The B-w version also does well, with computation times comparable to the MILP formulation and the FILTER solver. Similar observations were made in (Le Thi and Dinh, 2011). Enhancements on the bilinear penalties reduce time and iterations by 70–78% and 58–63%, respectively. Note that relative decrease in computation time is larger than that in iteration count. This confirms the hypothesis in Section 2.2.2 that the fixing of the variables in the enhancement for the bilinear penalty function not only speeds up convergence, but also leads to smaller subproblems that can be solved more quickly. Nevertheless, the BL variants still require one to two orders of magnitude more computation time than the PL variants for these LCP instances.

2.3.2. MacMPEC library

The MacMPEC library (Leyffer, 2000) contains a collection of MPCCs, from which we constructed LPCC instances by linearizing the objective function and constraints around the origin. The original upper and lower variable bounds were kept, and set to 1,000 where no bounds were defined. A total of 128 instances were created. In order to select instances that are guaranteed to be feasible, we first solved the MILP formulations and identified 96 instances in which the MILP solver found a feasible point within a maximum time limit of 900 wall clock seconds. Thirteen of these 96 instances have complementarity dimension greater than 100.

Table 2.2 shows the number of instances that reach the different outcomes for each method. We note that the piecewise linear decomposition never exceeds the iteration limit, and that the enhancement for the bilinear decompositions helps to reduce the number of iterations and allows more instances to be solved within the limit. First we consider the quality of the returned final iterates in terms of feasibility. As mentioned at the end of the introduction of Section 2.3, the bilinear methods terminate when the progress in the penalty function becomes very small (below $\varepsilon_{\text{tol}} = 10^{-8}$), but this does not imply that complementarity variables that are zero at a solution are very small as well. For example, consider a complementary pair with $y_i = w_i = 10^{-4}$. The corresponding penalty term is $y_i \cdot w_i = 10^{-8}$, but the feasibility violation is $V_{\perp}(y_i, w_i) = 10^{-4}$. Therefore, the choice of the feasibility tolerance $\varepsilon_{\text{feas}}$ is very crucial when deciding which final iterates constitute a successful run of a method in our statistics. In Table 2.2 we see that the number of runs that are counted as successful increases significantly with $\varepsilon_{\text{feas}}$ for the bilinear methods, whereas it has very little effect for the piecewise linear methods. This demonstrates that the bilinear versions are less capable of resolving complementarities to high accuracy. One may argue that a much tighter tolerance ε_{tol} for the DCA should be chosen, but most likely this will lead to numerical difficulties since the subproblems are solved only up to a certain tolerance as well (CPLEX has a default tolerance of 10^{-6}). We also observe, however, that the enhancement for the bilinear versions, which relies on fixing variables to 0 exactly, clearly improves this aspect of the solutions. Given the observations in Table 2.2, we chose the feasibility tolerance $\varepsilon_{\text{feas}} = 10^{-5}$ for all our statistics.

Next we examine the relative performance of the methods in terms of objective function values, first among only the DC methods. Again, we see that the enhancements for the bilinear methods lead to a significant improvement in solution quality. The enhancement for the piecewise linear method has less of an effect. We also observe that, for this problem set, the unbiased E starting point typically leads to better solutions than the one based on the LP relaxation, and that the perturbation parameter $\gamma_1 = 1.01$ appears to be the better choice for the BL1 variants. The B-w variant, missing an enhanced version, does not perform well. Finally, we assess the solution quality compared to the global optima (obtained by the MILP solver) and relative to the FILTER code. The "E*" variants find the global solution in 70-77% of the instances. Overall, among the DC methods, PL-E* and BL2-E* are the more effective variants for this problem set in terms of solution quality. A comparison with the FILTER code shows that this MPCC solver finds a better solution than the best DC variants more often than not.

Table 2.2. Number of problems with specific outcomes. "Max iter": Maximum number of iterations exceeded (excluded from remaining statistics); "$\varepsilon_{\text{feas}}$ =": Returned iterate satisfies $V_\perp \leq \varepsilon_{\text{feas}}$; "Best DC": as good as best DC method; "Global found": Global solution found; "FILTER better": Number of instances in which FILTER objective is better; "FILTER worse": Number of instances in which FILTER objective is worse.

Method	Max iter	$\varepsilon_{\text{feas}} = 10^{-8}$	$\varepsilon_{\text{feas}} = 10^{-5}$	$\varepsilon_{\text{feas}} = 10^{-2}$	Best DC	Global found	FILTER worse	FILTER better
PL-E	0	83	83	85	73	71	12	24
PL-R	0	72	72	78	68	66	7	28
PL-E*	0	83	83	85	76	74	12	21
PL-R*	0	72	72	78	68	66	7	28
BL1-E-γ_1	11	39	58	83	33	32	3	59
BL1-R-γ_1	6	32	47	85	22	21	6	67
BL1-E*-γ_1	2	82	83	91	76	73	8	17
BL1-R*-γ_1	2	75	75	89	66	63	9	28
BL1-E-γ_2	13	24	42	74	30	30	3	64
BL1-R-γ_2	7	21	40	80	19	19	6	71
BL1-E*-γ_2	2	66	76	85	70	67	5	23
BL1-R*-γ_2	3	72	73	85	64	61	9	32
BL2-E	10	45	66	83	43	41	8	55
BL2-R	9	42	62	84	35	33	7	64
BL2-E*	1	81	84	92	77	74	11	17
BL2-R*	2	77	78	91	71	68	12	26
B-w-E	58	28	28	30	27	27	3	65
B-w-R	31	46	46	50	45	45	5	46
FILTER	-	79	92	95	-	76	-	-

Table 2.3 compares the geometric averages of computation time and iterations between original and enhanced methods. We only included the 52 instances where at least one method took over 1 second of wall clock time to solve. For each method, averages are computed over all runs with successful outcomes. Comparing original and improved versions, we again observe that the enhancement for the BL methods leads to a significant reduction in computation time and iteration count. As expected, the PL methods increase the times slightly when the improvement is enabled, because it is based on continuing

the DCA with additional iterations that use different choices of the subgradients. We see that the average computation time for PL methods is slightly higher than that of MILP and FILTER, and the enhanced BL variants take 5-8 times as long as the PL versions. We also see that for these problems, the average number of iteration is around 3 for the PL methods, and 5-7 for the well-performing BL variants.

Table 2.3. Geometric averages of computation times (in seconds) and iterations for MacMPEC instances, including the original ("Orig") and enhanced ("Enhcd") variants of the decompositions. The number of instances with successful outcomes is given in the last columns.

	Average time		Average iterations		Successful	
Method	Orig	Enhcd	Orig	Enhcd	Orig	Enhcd
PL-E	0.031	0.032	2.768	3.268	46	46
PL-R	0.025	0.028	2.631	2.857	41	41
BL1-E-γ_1	0.552	0.140	16.784	5.026	29	45
BL1-R-γ_1	0.436	0.147	14.443	5.653	28	42
BL1-E-γ_2	0.739	0.153	23.449	5.912	21	42
BL1-R-γ_2	0.539	0.202	21.769	7.378	24	40
BL2-E	0.449	0.121	13.721	4.448	33	45
BL2-R	0.356	0.114	11.898	4.610	31	41
MILP	0.022	-	-	-	52	-
FILTER	0.019	-	-	-	52	-

2.3.3. Medium-scale inverse QP instances

Consider the convex quadratic program (QP):

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \tfrac{1}{2} y^T Q y + c^T y \\ \text{subject to} \quad & Ay \geq b, \end{aligned} \tag{2.31}$$

where $Q \in \mathbb{R}^{n \times n}$ is symmetric positive definite, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. For given matrices Q and A, the inverse convex quadratic problem, as described in (Hu et al., 2012b), consists in finding vectors x, b, and c that solve the forward problem (2.31) and are the least deviated from some given vectors $\bar{x}, \bar{b}$, and $\bar{c}$; that is (x, b, c) solves

$$\begin{array}{ll} \underset{x,b,c}{\text{minimize}} & \left\| (x,b,c) - (\bar{x},\bar{b},\bar{c}) \right\|_1 \\ \text{subject to} & \left\{ \begin{array}{lll} x \in & \underset{y}{\arg\min} & \frac{1}{2} y^T Q y + c^T y \\ & \text{subject to} & Ay \geq b. \end{array} \right. \end{array}$$

By stating the KKT conditions in the second-level problem and reformulating the ℓ_1-norm with slack variables, we obtain the LPCC

$$\begin{array}{lllllll} \underset{x,b,c,z^x,z^b,z^c,\lambda}{\text{minimize}} & \multicolumn{6}{l}{\sum_{i=1}^{n} z_i^x + \sum_{j=1}^{m} z_j^b + \sum_{i=1}^{n} z_i^c} \\ \text{subject to} & \multicolumn{6}{l}{Qx + c - A^T \lambda = 0} \\ & -z^x \leq & x - \bar{x} & \leq z^x, & -u^x \leq & x & \leq u^x \\ & -z^b \leq & b - \bar{b} & \leq z^b, & -u^b \leq & b & \leq u^b \\ & -z^c \leq & c - \bar{c} & \leq z^c, & -u^c \leq & c & \leq u^c \\ \text{and} & 0 \leq \lambda \perp & Ax - b & \geq 0, & & \lambda & \leq u^\lambda \end{array} \tag{2.32}$$

For our experiments, feasible random inverse QP instances were generated in MATLAB with the following procedure, given dimensions m and n, and a sparsity level s. The parameter s was chosen so that, on average, every matrix had at most 10 non-zero elements. We include explicit bounds u^x, u^b, u^c, and u^λ to compare with the MILP formulation.

```
1: Generate a sparse random symmetric positive definite matrix $Q \in \mathbb{R}^{n\times n}$ and a sparse matrix $A$ in $\mathbb{R}^{m\times n}$, using the MATLAB commands $\textsc{sprandsym}(n, s, 0.5, 1)$ and $\textsc{sprand}(m, n, s)$.
2: Generate a random vector $x \in \mathbb{R}^n$ with components following a normal distribution $N(0, 1)$.
3: Generate vectors $\widetilde{\lambda}$ and $\widetilde{w} \in \mathbb{R}^m$ uniformly at random between 0 and 10.
4: Generate a random binary vector $v \in \{0, 1\}^m$ and set $\lambda = \widetilde{\lambda} \circ v$ and $w = \widetilde{w} \circ (\mathbf{1} - v)$, so that $\lambda \perp w$.
5: Define $b \triangleq Ax - w$ and $c \triangleq A^T\lambda - Qx$.
6: Perturb $x, b$ and $c$ with normally distributed $N(0, 1)$ noise to obtain vectors $\bar{x}, \bar{b}$ and $\bar{c}$.
7: Set upper bounds $u^x \triangleq 10 \max\{|x_i|\}$, $u^b \triangleq 10 \max\{|b_i|\}$, $u^c \triangleq 10 \max\{|c_i|\}$, $u^\lambda \triangleq 10 \max\{|\lambda_i|\}$.
```

By construction, vectors x, b, c, and λ are feasible to (2.32). A total of 120 instances were generated with complementarity dimension m equal to 10, 25, 50, 100, 250, and 500, and 20 instances per size. Dimension n was chosen as $0.75 \times m$. The experiments are summarized in Table 2.4 and Figure 2.6. In terms of the robustness of the methods, the final columns in Table 2.4 show once again that the enhancements for the BL methods are essential for good performance. We also see that the PL variants are successful in all instances. Since decomposition B-w showed poor results on these instances (solving less than 25% of them), it is not included in these statistics.

Table 2.4. Computation times on Inverse QP instances. For each method and problems size, the arithmetic averages of the computation times (in seconds) are given. The numbers within the parentheses list the number of successfully solved problems over which the averages are taken. The "It" column displays the number of instances in which a method ran out of iterations, and the "Inf" column the number of problems in which the complementarity violation is not below $\varepsilon_\perp = 10^{-5}$.

Method	$m = 10$	$m = 25$	$m = 50$	$m = 100$	$m = 250$	$m = 500$	It	Inf
PL-E	0.03 (20)	0.04 (20)	0.05 (20)	0.1 (20)	0.64 (20)	5.17 (20)	0	0
PL-R	0.03 (20)	0.03 (20)	0.04 (20)	0.08 (20)	0.45 (20)	3.81 (20)	0	0
PL-E*	0.03 (20)	0.05 (20)	0.07 (20)	0.12 (20)	0.72 (20)	6.60 (20)	0	0
PL-R*	0.03 (20)	0.04 (20)	0.05 (20)	0.11 (20)	0.53 (20)	5.87 (20)	0	0
BL1-E-γ_1	1.61 (17)	2.26 (3)	-	-	-	-	95	5
BL1-R-γ_1	0.80 (16)	1.60 (9)	-	-	-	-	87	8
BL1-E*-γ_1	0.18 (20)	0.32 (19)	0.65 (20)	2.28 (18)	7.62 (18)	22.99 (20)	0	5
BL1-R*-γ_1	0.14 (20)	0.24 (19)	0.62 (20)	2.09 (19)	6.33 (18)	20.88 (20)	0	4
BL1-E-γ_2	1.69 (6)	-	-	-	-	-	113	1
BL1-R-γ_2	1.27 (14)	1.70 (4)	-	-	-	-	99	3
BL1-E*-γ_2	0.27 (20)	0.51 (19)	1.06 (20)	3.46 (18)	12.3 (19)	32.63 (17)	3	4
BL1-R*-γ_2	0.16 (20)	0.36 (19)	0.91 (20)	3.09 (19)	9.57 (20)	29.88 (19)	1	2
BL2-E	0.99 (20)	1.60 (11)	2.78 (8)	6.62 (4)	-	-	71	6
BL2-R	0.69 (18)	1.06 (10)	2.50 (8)	-	-	-	75	9
BL2-E*	0.16 (20)	0.28 (19)	0.51 (20)	1.60 (19)	6.58 (20)	18.64 (20)	0	2
BL2-R*	0.11 (20)	0.21 (20)	0.36 (20)	1.43 (18)	4.79 (20)	15.11 (19)	0	3
MILP	0.17 (20)	0.49 (20)	0.87 (20)	147.16 (20)	904.38 (20)	900.58 (20)	0	0
FILTER	0.01 (20)	0.03 (20)	0.23 (20)	1.35 (20)	44.59 (20)	547.96 (20)	0	0

The plots in Figure 2.6 compare solution quality obtained by the various methods. The performance measure per instance is calculated as a ratio between the objective value of each method and the best solution among all methods in Table 2.4. If a method failed on an instance, its ratio is considered infinite. The profiles in the graphs count how many instances have a performance measure below a threshold, given on the x-axis. Note that the x-axis scale differs from one plot to another. Plot 2.2 illustrates that the enhancement for the PL methods shifts the curves upwards, implying that solution quality improves. In fact, the number of cases in which the method terminates at a point that is proven

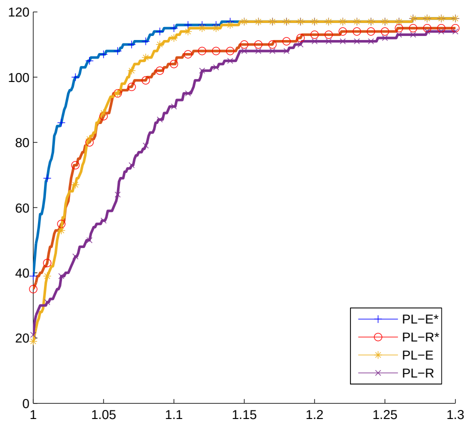


Figure 2.2. PL variants

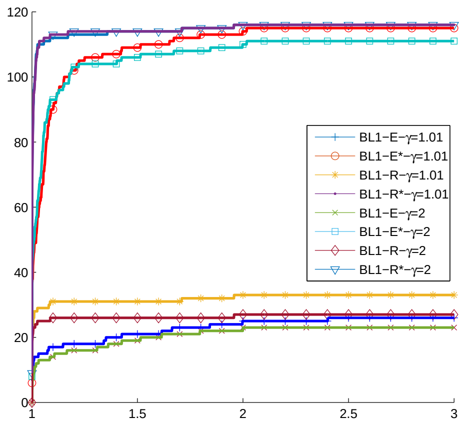


Figure 2.3. BL1 variants

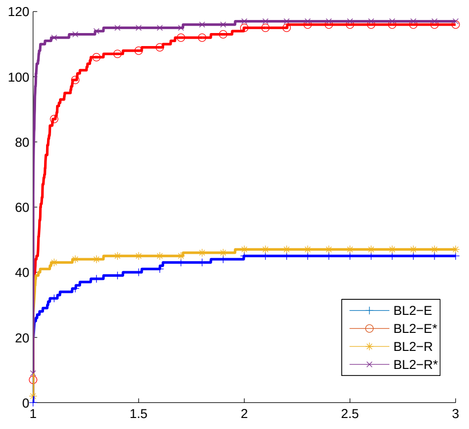


Figure 2.4. BL2 variants

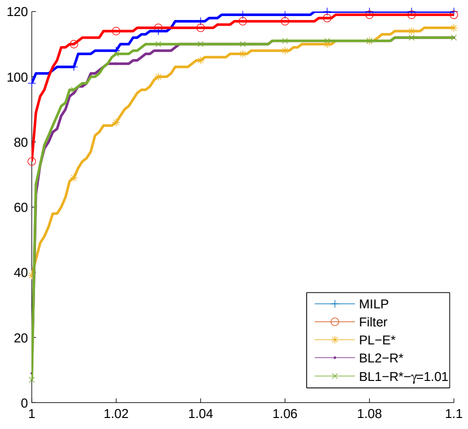


Figure 2.5. Best DC variants vs MILP vs FILTER

Figure 2.6. Inverse QP Performance Profiles.

to be strongly stationary increases from 28 (31) to 116 (115) out of 120 for the E (R) starting point. This demonstrates the effectiveness of the procedure in Section 2.2.1 that aims at escaping non-strongly stationary points by choosing different subgradients. The best performing method is PL-E*, which solves 100 problems with an objective value within 3% of the best solution, and 114 within 10%. Plots 2.3 and 2.4 confirm that the enhancement for the bilinear methods increases robustness. Using the optimal solution of the LP relaxation as starting point provides better objective values. The choice of the perturbation parameter γ appears to be less relevant. Plot 2.5 compares the best methods of the three previous plots with the MILP and FILTER solutions. The former finds the best solution in 98 cases and the latter in 74. FILTER, MILP, BL1-R*-γ_1, BL2-R* and PL-E* solve 110, 103, 96, 95, 69 of the problems within 1% of the best solution, and 117, 119, 110, 110, 107 of the problems within 5%, respectively. This demonstrates the competitiveness of the DC approach in terms of solution quality.

Although the MILP formulation and the FILTER solver provide slightly better solution quality, Table 2.4 considers a different perspective. Even for the largest case $m = 500$, the PL variants solve the instances in an average of 6.6 seconds, and the enhanced BL methods within 32 seconds. In contrast, FILTER requires on average almost 10 minutes. Even more dramatically, the CPLEX solver runs out of the wall clock time limit of 15 min in all instances using 32 cores, while the other approaches run on only a single core. This shows a very clear advantage of the DC approach, particularly since there is only a relatively small loss in solution quality.

2.3.4. Large-scale inverse QP instances

Our final set of experiments considers large-scale instances of the inverse QP problem with up to 5,000 complementarity constraints. We compare the relative performance of the best methods identified in the previous section. The FILTER solver was not able to solve any instance of that size, and the MILP formulation was not attempted because already the $m = 500$ instances ran out of time on 32 cores. Table 2.5 shows the average computation time (in seconds) and solution quality, defined as the ratio of the objective value of a method over the best objective value, taken over the successfully solved instances (given in the "#" columns). Whenever BL2-R* solves an instance, it obtains the best objective value. The largest deviation from the best solution among all instances is 7%, 11%, 6%, and 7% for PL-E*, PL-R*, BL1-R*-γ_1, BL1-R*-γ_2, respectively. Again, only the PL variants solve all of the instances to the desired level of feasbility. Considering solution time, even the largest instances are solved within at most 1 hour on average. The slowest successful run was BL1-R*-γ_2 for one $m = 5,000$ instance with 66 minutes. In contrast to the smaller instances in the previous section, the computation times for the PL variants, which require only LP solves, is no longer shorter than those for the BL methods, which require the solution of QPs. This is likely due to the different scalability of the simplex LP solver and the interior point QP solver, which are the default methods in CPLEX. Overall, variant BL2 based on the newly proposed decomposition of the bilinear penalty function is the most successful option in this test set.

Table 2.5. Computation time and solution quality for large-scale inverse QP instances.

Method	$m = 1,000$			$m = 2,500$			$m = 5,000$		
	Avg time	Avg ratio	#	Avg time	Avg ratio	#	Avg time	Avg ratio	#
PL-E*	23.72	1.02	5	316.88	1.02	5	3101.21	1.01	5
PL-R*	27.85	1.03	5	279.48	1.04	5	2875.57	1.02	5
BL1-R*-γ_1	48.37	1.01	5	384.30	1.02	5	2577.43	1.00	2
BL1-R*-γ_2	77.14	1.02	5	472.40	1.01	2	3452.60	1.00	3
BL2-R*	41.03	1.00	5	315.85	1.00	5	2156.08	1.00	4

2.4. Conclusions and Further Research

We examined four different DC decompositions of penalty function reformulations for LPCCs. Three of these correspond to decompositions previously proposed in (Le Thi and Dinh, 2011) for solving LCPs. We established relationships between the stationary points and local minima of the LPCCs and those of the penalty problems. Numerical experiments demonstrate that the proposed improvement of the piecewise linear penalty variation can significantly improve the solution quality by escaping non-strongly stationary points. Similarly, the enhancement of the bilinear penalty variant is shown to reduce the number of iterations significantly. Overall, the DC algorithm and its enhancements are competitive in terms of objective function values compared to state-of-the-art solvers, and are the only option for the large-scale test problems examined here. With these encouraging results of the DC approach for solving LPCCs, further research is warranted to study the application of this approach to MPCCs with DC objectives and linear complementarity constraints.

CHAPTER 3

Global Optima for Linear Programs with Complementarity Constraints

3.1. Logical Benders Approach for Globally solving Linear Programs with Complementarity Constraints

3.1.1. Introduction

This chapter focuses on the global solution of Linear Programs with Complementarity Constraints (LPCC) in its general form

$$
\begin{array}{llrl}
 & \underset{x,y,w}{\text{minimize}} & g^T x & \\
(3.1) & \text{subject to} & A_I x + B_I y + C_I w & \leq b_I \\
 & & A_E x + B_E y + C_E w & = b_E \\
 & 0 \leq & y \perp w & \geq 0
\end{array}
$$

where $x \in \mathbb{R}^{n_x}$, $y, w \in \mathbb{R}^{n_c}$, $b_I \in \mathbb{R}^{k_I}$ and $b_E \in \mathbb{R}^{k_E}$. Dimensions for matrices A_I, B_I, C_I, A_E, B_E and C_E are defined accordingly. Notice that the objective function only depends on the variable x. This is not restrictive in the formulation since any dependence on y and/or w can be taken into account within the equality constraints.

By global solution of the LPCC we mean certifying the problem is in one of its three possible states: infeasible, unbounded below or having a finite optimal solution. If both y

and w have bounded feasible regions, then it is well known that (3.1) can be reformulated as an MILP, by setting diagonal matrices M_y and M_w and a binary vector $p \in \{0,1\}^{n_c}$ representing either side of the complementarity piece.

$$
\begin{aligned}
\underset{x,y,w}{\text{minimize}} \quad & g^T x \\
\text{subject to} \quad & A_I x + B_I y + C_I w \le b_I \\
& A_E x + B_E y + C_E w = b_E \\
& w \ge M_w p \\
& y \ge M_y (1-p) \\
& y, w \ge 0.
\end{aligned} \tag{3.2}
$$

The main challenge with this formulation is the computation of valid bounds to obtain the diagonal values of matrices M_y and M_w. Furthermore, if either y or w is not bounded then this reformulation cannot be applied.

While there have been significant advances on computational methods based on nonlinear programming (NLP) to solve MPCCs, their main focus is to find some type of stationary point in an efficient manner. Solvers such as FILTER Fletcher et al. (2002) and KNITRO Byrd et al. (2006), based on sequential quadratic programming and interior point methods, respectively, are capable of finding solutions very quickly, but have no guarantees on the quality of the computed solution. Since an LPCC can be interpreted as a disjunctive linear optimization problem, global solution methods are mainly based on enumeration schemes. Therefore, many integer programming based approaches have been tested on

LPCC, combining essentially branch-and-bound and cutting plane methods (see (Bard and Moore, 1990; Jeroslow, 1978; Ibaraki, 1971, 1973; Yu, 2011; Yu et al., 2018)). Based on the works of Hooker and Ottosson (Hooker and Ottosson, 2003), a logical Benders approach was developed for LPCCs (Hu et al., 2008) and later extended to Q(uadratic)PCC (Bai et al., 2013), where the complementarity pieces are described by binary variables, and later discarded for exploration via a logical Benders cut generation scheme. Although originally stated as an extension to these logical Benders based approaches, the method presented in this chapter is also closely related to branch-and-bound as it will be described in the next section. The main contributions of this chapter are providing an interpretation for logical Benders as a reversed version of branch-and-bound methods, and taking advantage of this relationship to develop a more efficient and robust way of selecting and discarding pieces within the logical Benders framework. We also complement the Benders cut generation scheme with an ℓ_1-norm minimization problem to speed up the process.

3.1.2. Problem statement

For the remainder of this chapter, the following set is defined $\Omega_P := \{(x, y, w) | A_I x + B_I y + C_I w \leq b_I;\ A_E x + B_E y + C_E w = b_E;\ y, w \geq 0\}$, which represents the feasible region of (3.1) without the complementarity constraint.

The general idea of the algorithm presented in this chapter follows the one exposed by Hu et al. (2008). A master problem selects different complementarity pieces which are solved to optimality and the corresponding solution is stored as the incumbent if it is the best one found so far. Pieces are discarded by means of a cut generation method in the master problem. The method ends when all pieces have been explored or discarded. In the following subsection we describe the general idea. For further details please refer to (Hu et al., 2008).

3.1.3. Logical Benders Decomposition

Let $\mathcal{N} := \{1, \ldots, n_c\}$. Given a partition $(I^w, I^y) \subseteq \mathcal{N}$, a complementarity piece for problem (3.1) defined by I^w and I^y is a subset of its feasible region with the added restrictions $w_i \leq 0,\ i \in I^w$ and $y_i \leq 0,\ i \in I^y$. Notice that since (I^w, I^y) is a partition, we have $w^T y = 0$. In our setting, these partitions will be described by a binary vector $p \in \{0,1\}^{n_c}$ in the following way:

$$\begin{aligned} \phi_P(p) = \quad & \underset{(x,y,w)\in\Omega_P}{\text{minimize}} \quad g^T x \\ & \text{subject to} \quad w_i \leq 0,\ i : p_i = 0 \quad (\lambda_i^w) \\ & \qquad\qquad\quad\ \ y_j \leq 0,\ j : p_j = 1 \quad (\lambda_j^y) \end{aligned} \tag{3.3}$$

where the symbol in parenthesis represents the dual variables. In this case $I^w = \bar{I}^w(p) := \{i : p_i = 0\}$ and $I^y = \bar{I}^y(p) := \{i : p_i = 1\}$. It is clear that for any vector (x, y, w) feasible in (3.1) there exists a binary vector p such that (x, y, w) is also feasible in (3.3). By convention we assume that $\phi_P(p) = \infty$ if (3.3) is infeasible. Therefore, (3.1) is equivalent to

$$\underset{p\in\{0,1\}^{n_c}}{\text{minimize}} \quad \phi_P(p). \tag{3.4}$$

Instead of exploring all 2^{n_c} possible pieces explicitly, we maintain a master problem that keeps track of all the pieces that still need to be explored. We denote the state of the master problem by a set $\mathcal{C}$ that consists of pairs (I^w, I^y) of disjoint subsets I^w and I^y of $\mathcal{N}$. Then the set of pieces p that still have to be explored is given by

$$\mathcal{F} = \left\{ p \in \{0,1\}^{n_c} : \sum_{i \in I^w} p_i + \sum_{i \in I^y} (1 - p_i) \geq 1 \text{ for all } (I^w, I^y) \in \mathcal{C} \right\}. \tag{3.5}$$

We refer to an inequality in the above definition as a "cut" in the master problem, and use the corresponding sets I^w and I^y to denote the cut. For example, once (3.3) has been solved for a given piece p, we could add the cut defined by $I^w = \bar{I}^w(p)$ and $I^y = \bar{I}^y(p)$ to $\mathcal{C}$. In this way, the piece p is no longer in the set of unexplored pieces $\mathcal{F}$. If $I^w \subseteq \bar{I}^w(p)$ and $I^y \subseteq I^y(p)$ is not a partition of $\mathcal{N}$, also other pieces will be removed by the corresponding cut. Clearly, the fewer elements are in I^w and I^y (i.e., the sparser the cut is), the more elements are removed from $\mathcal{F}$. Section 3.3.2 discusses techniques for computing sparse cuts.

3.1.3.1. Algorithm Outline. In each iteration of the logical Benders algorithm, a piece p is chosen from $\mathcal{F}$. Then the subproblem (3.3) corresponding to p is solved, and based on the outcome a new cut $I^w \subseteq \bar{I}^w(p)$ and $I^y \subseteq \bar{I}^y(p)$ is added to $\mathcal{C}$. This cut will certainly remove the just-explored piece p from $\mathcal{F}$. If (3.3) is feasible, we also obtain a feasible point for the original problem. The algorithm keeps the best feasible point encountered so far as the incumbent, together with its optimal objective value U, which is an upper bound for the optimal objective value of (3.1).

Since there are only finitely many pieces, $\mathcal{F}$ must eventually become empty. All pieces have been (implicitly) explored and the algorithm terminates. The current incumbent is an optimal solution for (3.1). If no incumbent has been found, the original problem is infeasible. And if during any iteration (3.3) is unbounded below, (3.1) is unbounded.

3.1.3.2. Cut generation. For a given piece p, we can define a relaxation indexed by $I^w \subseteq \bar{I}^w(p)$ and $I^y \subseteq \bar{I}^y(p)$.

$$
\begin{aligned}
\phi_P(I^w, I^y) = \quad & \underset{(x,y,w)\in\Omega_P}{\text{minimize}} \quad g^T x \\
& \text{subject to} \quad w_i \leq 0 \text{ for } i \in I^w \quad (\lambda_i^w) \\
& \qquad\qquad\quad\; y_j \leq 0 \text{ for } i \in I^y \quad (\lambda_j^y)
\end{aligned} \tag{3.6}
$$

Again, we define $\phi_P(I^w, I^y) = \infty$ if (3.6) is infeasible. Because (3.6) is a relaxation of (3.3), we have $\phi_P(I^w, I^y) \leq \phi_P(p)$. Note that (3.6) is also a relaxation of (3.3) for any other $\hat{p}$ such that $I^w \subseteq \bar{I}^w(\hat{p})$ and $I^y \subseteq \bar{I}^y(\hat{p})$, and therefore $\phi_P(I^w, I^y) \leq \phi_P(\hat{p})$.

Now suppose that U is the objective value for the current incumbent and therefore an upper bound on the optimal objective function. In our search for a better incumbent, we need to find a piece $\hat{p}$ so that $\phi_P(\hat{p}) < U$. Consequently, if $\phi_P(I^w, I^y) \geq U$, all pieces $\hat{p}$ with $I^w \subseteq \bar{I}^w(\hat{p})$ and $I^y \subseteq \bar{I}^y(\hat{p})$ cannot contain a better incumbent, and we can exclude them all from $\mathcal{F}$. This is done by adding the cut (I^w, I^y) to $\mathcal{C}$.

To find suitable subsets I^w and I^y, let us consider the dual to (3.3):

$$
\begin{aligned}
\phi_D(I^w, I^y) = \quad & \underset{\mu_I,\, \mu_E,\, \lambda^w,\, \lambda^y}{\text{maximize}} \quad && -b_I^T \mu_I + b_E^T \mu_E \\
& \text{subject to} && -A_I^T \mu_I + A_E^T \mu_E && = g \\
& && -B_I^T \mu_I + B_E^T \mu_E - \lambda^y && \leq 0 \\
& && -C_I^T \mu_I + C_E^T \mu_E - \lambda^w && \leq 0 \\
& && \textstyle\sum_{i \notin I^w} \lambda_i^w + \sum_{i \notin I^y} \lambda_i^y && = 0 \\
& && \mu_I,\ \lambda^w,\ \lambda^y && \geq 0,
\end{aligned}
\tag{3.7}
$$

In (3.6), there are constraints on w_i for $i \in I^w$, and the dual should contain only the corresponding multipliers λ_i^w. To simplify the notation, the multipliers for $w \leq 0$ have been extended to a full vector $\lambda^w \in \mathbb{R}^{n_c}$, and λ^y is similarly defined. The constraint $\sum_{i \notin I^w} \lambda_i^w + \sum_{i \notin I^y} \lambda_i^y = 0$ makes sure that the newly introduced components of λ^w and λ^y must be zero, so that (3.7) is indeed the dual of (3.6).

Assuming the primal problem has a finite optimal solution, we have $\phi_P(I^w, I^y) = \phi_D(I^w, I^y)$. It is then clear that (I^w, I^y) defines a valid cut if $\phi_D(I^w, I^y) \geq U$, or, equivalently, the set

$$\Omega_D(I^w, I^y) := \left\{ (\mu_I,\, \mu_E,\, \lambda^w,\, \lambda^y) : \begin{array}{rl} -b_I^T\mu_I + b_E^T\mu_E & \geq U \\ A_I^T\mu_I + A_E^T\mu_E & = g \\ -B_I^T\mu_I + B_E^T\mu_E - \lambda^y & \leq 0 \\ -C_I^T\mu_I + C_E^T\mu_E - \lambda^w & \leq 0 \\ \sum_{i \notin I^w} \lambda_i^w + \sum_{i \notin I^y} \lambda_i^y & = 0 \\ \mu_I,\, \lambda^w,\, \lambda^y & \geq 0 \end{array} \right\} \tag{3.8}$$

is not empty.

In (Hu et al., 2008), the following procedure was suggested to obtain a valid cut: Given a piece p, solve the primal (3.3). If it is feasible, let $(\mu_I,\, \mu_E,\, \lambda^w,\, \lambda^y)$ be a dual optimal solution (with λ^w and λ^y properly extended). Now define $I^w = \{i \in \bar{I}^w(p) : \lambda_i^w > 0\}$ and $I^y = \{i \in \bar{I}^y(p) : \lambda_i^y > 0\}$. Then it is easy to see that $(\mu_I,\, \mu_E,\, \lambda^w,\, \lambda^y) \in \Omega_D(I^w, I^y)$ and therefore (I^w, I^y) defines a valid cut. If the optimal solution in (3.3) is not strictly complementary, then this procedure produces a cut that removes more than just p from $\mathcal{F}$. Note that this implies that the relaxation $\phi_P(I^w, I^y) \geq U$.

Bai et al. (2013) discuss a procedure to sparsify the cut further. They choose subsets $\tilde{I}^w \subseteq I^w$ and $\tilde{I}^y \subseteq I^y$ and solve the relaxed primal (3.6) to get $\phi_P(\tilde{I}^w, \tilde{I}^y)$. If $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq U$, we can set $(I^w, I^y) \leftarrow (\tilde{I}^w, \tilde{I}^y)$, which is also a valid cut. This can be repeated until $\phi_P(\tilde{I}^w, \tilde{I}^y) < U$.

We can proceed similarly if the primal problem is infeasible. In this case, the dual must have an unbounded ray that can be found by solving the homogeneous version of (3.7):

$$
\begin{array}{rlrl}
\phi_{D_0}(I^w, I^y) = & \underset{\mu_I,\, \mu_E,\, \lambda^w,\, \lambda^y}{\text{maximize}} & -b_I^T\mu_I + b_E^T\mu_E & \\
& \text{subject to} & -A_I^T\mu_I + A_E^T\mu_E & = 0 \\
& & -B_I^T\mu_I + B_E^T\mu_E - \lambda^y & \leq 0 \\
& & -C_I^T\mu_I + C_E^T\mu_E - \lambda^w & \leq 0 \\
& & \sum_{i \notin I^w} \lambda_i^w + \sum_{i \notin I^y} \lambda_i^y & = 0 \\
& & \mu_I,\, \lambda^w,\, \lambda^y & \geq 0.
\end{array}
\tag{3.9}
$$

We will denote Ω_{D_0} to the set Ω_D with $g = 0$ and the constraint $-b_I^T\mu_I + b_E^T\mu_E \geq U$ replaced by $-b_I^T\mu_I + b_E^T\mu_E = 1$. As before, the generated cut (I^w, I^y) guarantees that any piece removed by it will also be infeasible.

If the constraints for $\phi_P(I^w, I^y)$ (resp. $\phi_D(I^w, I^y)$) define an infeasible set then it will be understood that $\phi_P(I^w, I^y) = \infty$ (resp. $\phi_D(I^w, I^y) = -\infty$).

If at any point during the algorithm we find that a primal piece is unbounded then (3.1) is also unbounded, so there is no more exploration required.

Initially the master problem feasible set $\mathcal{F}$ is $\{0,1\}^{n_c}$ and the upper bound $U = \infty$. A candidate complementarity piece p is selected. If $\phi_{D_0}(p) = \infty$, a suitable cut obtained from (3.9) is added to $\mathcal{F}$. Otherwise if $\phi_P(p) = -\infty$ then the master problem is unbounded and the method stops. Finally, if $\phi_P(p) < U$ then the bound is updated ($U = \phi_P(p)$) and a cut obtained from (3.7) is added. The process continues until $\mathcal{F}$ becomes infeasible. At

Algorithm 4 Logical Benders on LPCC

```
1: Let $\mathcal{F} = \{0,1\}^{n_c}$, $U = \infty$
2: while $\mathcal{F} \neq \emptyset$ do
3:     Select a piece $p \in \mathcal{F}$. Let $I^w = \bar{I}^w(p)$ and $I^y = \bar{I}^y(p)$
4:     Solve (3.9)
5:     if $\phi_{D_0}(I^w, I^y) = \infty$ then
6:         Add cut $(I^w, I^y)$ to $\mathcal{C}$
7:     else
8:         Solve (3.6)
9:         if $\phi_P(I^w, I^y) = -\infty$ then
10:            STOP; problem unbounded.
11:        else
12:            if $\phi_P(I^w, I^y) < U$ then
13:                Update $U = \phi_P(I^w, I^y)$.
14:                Store $x^* = \hat{x}$, $y^* = \hat{y}$ and $w^* = \hat{w}$.
15:            Add cut $(I^w, I^y)$ to $\mathcal{C}$.
16: Return $(x^*, y^*, w^*)$ as optimal solution or output infeasible or unbounded.
```

that moment the algorithm returns the current incumbent. These steps are summarized in Algorithm 4.

3.1.4. Contributions of this work

There are two key drivers of the performance of this method: The selection of the candidate piece p and the strength (sparsity) of the generated cuts. Notice that both steps are linked, the generated cut depends on the selected piece and this selection depends on cuts that have already been generated.

In the methods that have been proposed in the past (Hu et al., 2008; Bai et al., 2013), the piece selection and the sparsification procedures were independent of each other. In fact, the candidate piece was chosen just as any $p \in \mathcal{F}$, without giving any preferences of one over another.

The contribution of this chapter is that we consider the logical Benders algorithm from a different point of view, namely as a procedure that operates on a branch-and-bound tree (in reverse order compared to regular branch-and-bound methods). We describe this observation in the Section 3.2, and then put everything together in Section 3.3.

3.2. Interpretation within Branch-and-Bound Framework

3.2.1. Branch-and-Bound Trees

One well-known approach for solving MPCCs is based on branch-and-bound (B&B), introduced by Bard and Moore (1990). Efficient implementations of this framework include many enhancements, such as cutting planes and pseudo-cost branching (Yu, 2011; Yu et al., 2018). Here, we describe the basic B&B method with the purpose of interpreting the logical Benders algorithm as one that operates on a B&B tree. This will allow us to derive new piece selection and cut sparsification methods. The B&B method for LPCCs solves a collection of subproblems of the form (3.6), where I^w and I^y are disjoints subsets of $\mathcal{N}$. These subproblems correspond to nodes, denoted as $[I^w, I^y]$, in a binary enumeration tree (In our notation, we distinguish between cuts (I^w, I^y) and nodes $[I^w, I^y]$ by their parenthesis). The root node of such a tree corresponds to the subproblem described by $I^w = I^y = \emptyset$, i.e., none of the components of y and w are required to be complementary. Each node $[I^w, I^y]$ with $I^w \cup I^y \neq \mathcal{N}$ has two children. They are obtained by choosing a complementarity $j \in \mathcal{N} \setminus (I^w \cup I^y)$ that has not yet been fixed and by making $[I^w \cup \{j\}, I^y]$ and $[I^w, I^y \cup \{j\}]$ the index sets defining the children. We call j the branching complementarity. Different trees are obtained by choosing different branching complementarities at the nodes. Overall, there are $\Pi_{i=0}^{n_c}(n_c - i)^{2^i}$ possible trees T corresponding to (3.1) (Although, in the regular B&B method, the full tree never needs to be completely constructed, since subtrees are eliminated from the search once it is clear that they cannot contain the optimal solution). The nodes on the last level with $I^w \cup I^y = \mathcal{N}$ are called leaves. Here, all complementarity constraints have been fixed to one of the two sides. If a leaf is feasible, its optimal solutions are feasible for the original problem (3.1). Therefore, each leaf corresponds to a piece p in (3.1) and vice versa, with $I^w = \bar{I}^w(p)$ and $I^y = \bar{I}^y(p)$.

Since a child node $[\tilde{I}^w, \tilde{I}^y]$ is obtained by fixing a complementarity constraint to one of the two sides, its feasible region cannot be larger than that of its parent $[I^w, I^y]$. As a consequence, $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq \phi_P(I^w, I^y)$. This includes the case in which subproblems are infeasible and a quantity in that relationship is ∞.

Given a node $[I_a^w, I_a^y]$ and one of its descendants $[I_d^w, I_d^y]$ in a particular tree, we denote the path P from $[I_d^w, I_d^y]$ to $[I_a^w, I_a^y]$ by $j_1 \to j_2 \to j_3 \to \ldots \to j_L$, where j_l are the branching complementarities that were added to obtain a child node from its parent. The indices are listed in order from the descendant to the ancestor; for example, j_L represents the branching from node $[I_a^w, I_a^y]$.

3.2.1.1. Branch-and-Bound Algorithm. The B&B method constructs a tree during the course of the algorithm. It maintains a list of open nodes that have yet to be explored. At the beginning, this list is initialized with the root node. The root node corresponds to the subproblem described by $I^w = I^y = \emptyset$, i.e., none of the components of y and w are required to be complementary. Its optimal value $\phi_P(\emptyset, \emptyset)$ provides a lower bound on the optimal objective of the original problem (3.1). The method also stores an incumbent, which is a point that is feasible for the original problem (3.1) with the lowest objective value, call it U, found so far. Clearly, U is an upper bound for the optimal objective value of (3.1). At the beginning, no incumbent is available, and we set $U \leftarrow \infty$.

In each iteration of the rudimentary B&B algorithm, an open node $[I^w, I^y]$ is chosen and the corresponding subproblem (3.6) is solved. There are four possible outcomes that determine the next step of the algorithm:

(1) If (3.6) is feasible and $\phi_P(I^w, I^y) \geq U$, all descendants $[I^w_d, I^y_d]$ of this node must also have $\phi_P(I^w_d, I^y_d) \geq U$. Consequently, no feasible point for (3.1) can be found among the descendants of $[I^w, I^y]$ with a better objective value than U.

(2) If $[I^w, I^y]$ is infeasible, also all its descendants must be infeasible, and again no better incumbent for (3.1) can be found below $[I^w, I^y]$.

(3) If (3.6) is feasible and its computed optimal solution is feasible for the original problem (3.1) and $\phi_P(I^w, I^y) < U$, then the optimal solution of the subproblem, if it is finite, provides a new incumbent. The incumbent and the corresponding upper bound U are updated.

(4) If (3.6) is feasible but its computed optimal solution is not feasible for the original problem (3.1), then the set of complementarities that has not been fixed, $\mathcal{N} \setminus (I^w \cup I^y)$, must be non-empty. The algorithm then chooses a branching complementarity $j \in \mathcal{N} \setminus (I^w \cup I^y)$ and adds the corresponding children $[I^w \cup \{j\}, I^y]$ and $[I^w, I^y \cup \{j\}]$ to the list of open nodes.

In the cases 1–3, there is no need to explore descendants of the current node, since no feasible solution better than the incumbent can be found in that part of the tree. In that case, we call the current node a fathomed node. The B&B algorithm does not explicitly construct the part of the tree below a fathomed node.

The algorithm terminates once the list of open nodes becomes empty. The current incumbent is the optimal solution of (3.1). If no incumbent was found during the search, the original problem is infeasible. Since there is only a finite number of possible open nodes and no node can become an open node twice, the algorithm is guaranteed to terminate in a finite number of iterations.

3.2.1.2. Proving Optimality. For simplicity we will assume in the following that the optimal point is available as incumbent and that the purpose of the algorithm is to confirm its optimality. This is reasonable in a practical setting if we first solve a big-M formulation of (3.1), namely

$$\begin{aligned} \underset{(x,y,w)\in\Omega_P,\, p\in\{0,1\}^{n_c}}{\text{minimize}} \quad & g^T x \\ \text{subject to} \quad & w_j \le M p_j \text{ for } j \in \mathcal{N} \\ & y_j \le M(1-p_j) \text{ for } j \in \mathcal{N}, \end{aligned} \tag{3.10}$$

for a finite $M > 0$ with an MILP algorithm. In this manner, we can use powerful off-the-shelf MILP solver implementations. Ideally, we would like to choose M large enough so that the optimal solution for (3.1) is not excluded. However, often such a value is not known a priori, and choosing a very large value for M renders the MILP formulation more difficult to solve, since its integer relaxation becomes weak.

Often, the optimal solution of (3.10) is optimal for the original problem (3.1), and what remains is to prove that it is indeed optimal. To this end, we follow the approach proposed by Bai et al. (2013) and define an "outer problem" that consists of the original problem (3.1) with the added linear constraint

$$\sum_{j \in \mathcal{N}} w_j + \sum_{j \in \mathcal{N}} y_j \geq M.$$

The union of the feasible set of the outer problem and that of the big-M MILP (3.10) (projected onto the (x, y, w) space) includes the feasible set of the original problem (3.1). Therefore, the best among the optimal solutions of the two problems is guaranteed to be the optimal solution of (3.1).

Solving the outer problem is the context in which the logical Benders algorithm can be applied. For the remainder of this section we assume that the optimal objective function value U for (3.1) is known. In case U is not optimal, the algorithm is still able to determine the optimal solution.

3.2.1.3. Logical Benders within Branch-and-Bound Tree. We now give an interpretation of the logical Benders algorithm using the concept of B&B trees.

Let T be a *fixed* B&B tree as defined in Section 3.2.1, that is, all braching decisions are predetermined all the way to all leaves, and let U be the optimal objective value of (3.1). We assume that the root node relaxation has a value worse than the upper bound, i.e., $\phi_P(\emptyset, \emptyset) < U$. Otherwise, we can terminate the procedure immediately, since $\phi_P(\emptyset, \emptyset)$ is always a lower bound on the optimal objective value. If $\phi_P(\emptyset, \emptyset) \geq U$, we can immediately conclude that U is optimal.

Inspired by the terminology of the basic B&B algorithm described in Section 3.2.1.1, we call a node $[I^w, I^y]$ in T a *fathomed node* if $\phi_P(I^w, I^y) \geq U$ and $\phi_P(\tilde{I}^w, \tilde{I}^y) < U$ for its parent $[\tilde{I}^w, \tilde{I}^y]$. A fathomed node is one for which the basic B&B algorithm would encounter one of the cases 1 or 2. The B&B algorithm would not explore the subtree below a fathomed node. On the other hand, the B&B algorithm would create children for any node $[\hat{I}^w, \hat{I}^y]$ above a fathomed node since no definite conclusion about the subtree below $[\hat{I}^w, \hat{I}^y]$ can be drawn based on the optimal solution of $[\hat{I}^w, \hat{I}^y]$ alone. Therefore, the fathomed nodes in a fixed tree T is the minimal set of nodes that must be solved at some point in order to prove that U is indeed the optimal objective value. Note that fathomed nodes depend exclusively on the structure of the tree T, and are independent of any algorithm.

Note that case 3 cannot occur since we assume that U is the optimal objective value. Also, the tree T with predetermined branching might not correspond to one that would be generated by the B&B algorithm. This is the case, for example, when child nodes are obtained by branching on a complementarity j for which the optimal solution of the parent is already complementary. To avoid unnecessary work, the B&B algorithm only chooses branching complementarities for which the corresponding optimal values of the current node are not complementary, see case 4. In our context of interpreting the logical Benders method using a B&B tree, however, we permit this situation.

As described next, while the B&B algorithm is finding the fathomed nodes "from above" by branching from the root node to the fathomed nodes, we can interpret the logical Benders algorithm as a method that finds the fathomed nodes "from below".

Algorithm 5 Find Fathomed Node

```
 1: Input: Piece $p \in \{0,1\}^{n_c}$, path $P$ from the leaf $[I^w(p), I^y(p)]$ to the root node.
 2: Let $j_1 \leftarrow j_2 \leftarrow \ldots \leftarrow j_{n_c}$ be the nodes along path $P$.
 3: Set $[I^w, I^y] = [I^w(p), I^w(p)]$
 4: for $l = 1, 2, 3, \ldots, n_c - 1$ do
 5:     Set $[\tilde{I}^w, \tilde{I}^y] = [I^w \setminus \{j_l\}, I^y \setminus \{j_l\}]$. ($[\tilde{I}^w, \tilde{I}^y]$ is the parent of $[I^w, I^y]$)
 6:     Solve (3.6) for $[\tilde{I}^w, \tilde{I}^y]$
 7:     if $\phi_P(\tilde{I}^w, \tilde{I}^y) < U$ then
 8:         Break (leave for loop)
 9:     Set $[I^w, I^y] \leftarrow [\tilde{I}^w, \tilde{I}^y]$
10: end for
11: Return $(I^w, I^y)$ as cut that identifies a fathomed node.
```

Consider a fixed tree T. The logical Benders algorithm starts with an empty set of cuts $\mathcal{C} = \emptyset$, and therefore $\mathcal{F} = \{0,1\}^{n_c}$. It then chooses a piece $p \in \mathcal{F}$, which corresponds to a leaf in the B&B tree. We now want to generate a cut (I^w, I^y) so that

$$\sum_{i \in I^w} p_i + \sum_{i \in I^y} (1 - p_i) \geq 1 \tag{3.11}$$

excludes p as well as many other pieces, if possible. In our set notation, (3.11) is satisfied and the leaf $[J^w, J^y] = [I^w(p), I^y(p)]$ corresponding to piece p is removed from $\mathcal{F}$ by the cut (I^w, I^y) if and only if

$$I^w \subseteq J^w \text{ and } I^y \subseteq J^y. \tag{3.12}$$

One way to generate a cut consistent with the tree T is presented in Algorithm 5.

Starting from the leaf corresponding to the piece p, this procedure works itself upwards in the tree. It continues along the path to the root until the parent $[\tilde{I}^w, \tilde{I}^y]$ of the current node $[I^w, I^y]$ has a value $\phi_P(\tilde{I}^w, \tilde{I}^y) < U$. In that case, $[I^w, I^y]$ must be a fathomed node. To express the fact that no solution better than U can be found in any of the leafs (or pieces) below $[I^w, I^y]$, we add the corresponding cut (I^w, I^y) to the master problem, so $\mathcal{C} \leftarrow \mathcal{C} \cup \{(I^w, I^y)\}$.

We now repeat this for the next iteration of the logical Benders decomposition, starting from any piece $p \in \mathcal{F}$ that has not yet been discarded by a cut. In the tree, this corresponds to any leaf that is not below a fathomed node. The algorithm above will again produce a new fathomed node that is added as a cut to the master problem. In this way, we will eventually discover all fathomed nodes in the tree. At that point, the feasible set $\mathcal{F}$ of the master problem will become empty, and the method concludes.

3.2.1.4. Minimal Cuts. In Section 3.1.3.2 we discussed the idea of sparsification. Given a valid cut (I^w, I^y), i.e., $\phi_P(I^w, I^y) \geq U$, it might be possible to find smaller sets $\tilde{I}^w \subseteq I^w$ and $\tilde{I}^y \subseteq I^y$ with $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq U$, so that $(\tilde{I}^w, \tilde{I}^y)$ still defines a valid cut. Such a sparser cut is preferable, since it excludes more pieces from $\mathcal{F}$, as can be seen from (3.11).

The hierarchy of cut sparsification defines a partial order, and we call the induced minimal elements *minimal cuts*. Within the algorithm described in the previous section, we can augment Algorithm 5 so that the procedure continues after the fathomed node has been found, see Algorithm 6. Whenever it turns out that removing a complementarity from the cut results in an invalid cut, it is simply added back, after which the search continues further along the path.

Algorithm 6 Cut Sparsification

```
1: Input: Piece $p \in \{0,1\}^{n_c}$, path $P$ from the leaf $[I^w(p), I^y(p)]$ to the root node.
2: Let $j_1 \leftarrow j_2 \leftarrow \ldots \leftarrow j_{n_c}$ be the nodes along path $P$.
3: Set $[I^w, I^y] = [I^w(p), I^w(p)]$
4: for $l = 1, 2, 3, \ldots, n_c$ do
5:     Set $[\tilde{I}^w, \tilde{I}^y] = [I^w \setminus \{j_l\}, I^y \setminus \{j_l\}]$. ($[\tilde{I}^w, \tilde{I}^y]$ is the parent of $[I^w, I^y]$)
6:     Solve (3.6) for $[\tilde{I}^w, \tilde{I}^y]$
7:     if $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq U$ then
8:         Set $[I^w, I^y] \leftarrow [\tilde{I}^w, \tilde{I}^y]$
9: end for
10: Return $(I^w, I^y)$ as minimal cut.
```

Clearly this procedure results in a minimal cut. It is important to note that the order in which the complementarities are released, i.e., the path (or tree), determines which particular minimal cut is found.

The set of minimal cuts does not depend on the choice of a particular tree, it is defined by the problem statement together with the upper bound U. Consider a minimal cut (I^w, I^y). We can now construct a tree in which $[I^w, I^y]$ is a fathomed node, simply by choosing the complementarities in $I^w \cup I^y$ as the branching decisions from the root node to $[I^w, I^y]$. On the other hand, given the index sets I^w and I^y from the minimal cut, in a different given tree $\tilde{T}$, there might not be a node $[I^w, I^y]$ with the same index sets. In that case, the cut given by (I^w, I^y) might correspond to more than one fathomed node in $\tilde{T}$, each one has a path to the root that includes all of the complementarities $I^w \cup I^y$. In fact, given a set of minimal cuts $\{I_k^w, I_k^y\}_{k=1}^K$, there might not exist a tree T in which those cuts correspond to nodes of the form $[I_k^w, I_k^y]$. We illustrate this observations with an example:

Example 3.2.1. *Consider the following LPCC:*

$$\begin{array}{lrll}
\textit{minimize} & -2x_1 - x_2 - x_3 & & \\
\textit{subject to} & x_i & \leq 4 & i = 1,2,3 \\
& x_i & = y_i & i = 1,2,3 \\
& \sum_{k \neq i} x_k - 3x_i + 6 & = w_i & i = 1,2,3 \\
0 \leq & y \perp w & \geq 0 &
\end{array} \tag{3.13}$$

and the set of cuts $(I_1^w, I_1^y) = (\emptyset, \{1\})$, $(I_2^w, I_2^y) = (\emptyset, \{2\})$ *and* $(I_3^w, I_3^y) = (\{1,2,3\}, \emptyset)$. *One solution to this problem is* $x_1 = x_2 = y_1 = y_2 = 3$ *and* $w_3 = 12$ *with all other variables set to zero. The optimal value of the LPCC is* $U = -9$ *and that of the relaxation is* $\phi_P(\emptyset, \emptyset) = -16$. *We have* $\phi_P(I_1^w, I_1^y) = -6$ *and* $\phi_P(I_2^w, I_2^y) = -9$, *hence* (I_1^w, I_1^y) *and* (I_2^w, I_2^y) *are minimal cuts. We also have* $\phi_P(I_3^w, I_3^y) = \infty$ *(infeasible) and* $\phi_P(\{1,2\}, \emptyset) = -14$, $\phi_P(\{1,3\}, \emptyset) = -14$ *and* $\phi_P(\{2,3\}, \emptyset) = -12$, *so* (I_3^w, I_3^y) *is also minimal.*

Let T *be any B&B tree for this problem. If the first branching complementarity was component* 1, *then the node* $[\emptyset, \{2\}]$ *corresponding to cut* (I_2^w, I_2^y) *does not exist. We can follow the same reasoning for any other first branching complementarity. Therefore, for every tree there exists at least one of these minimal cuts which does not correspond to a node.*

3.2.1.5. Nodes in a Branch-and-Bound Tree. Let T be a fixed tree, and $\mathcal{C} = \{(I_k^w, I_k^y) : k = 1, \ldots, K\}$ be a set of cuts. We can subdivide the nodes of T into three different classes. Let $[J^w, J^y]$ be a node in T.

- We say that node $[J^w, J^y]$ is *discarded* if there exists a cut $(I_k^w, I_k^y) \in \mathcal{C}$ so that $I_k^w \subseteq J^w$ and $I_k^y \subseteq J^y$ with at least one of the two inclusions being strict. In this case, the node $[J^w, J^y]$ lies below a fathomed node identified by (I_k^w, I_k^y). All the leaves in the subtree below $[J^w, J^y]$ correspond to pieces that are already excluded in $\mathcal{F}$.
- We say that node $[J^w, J^y]$ is *explored* if there exists a fathomed node corresponding to some cut $(I_k^w, I_k^y) \in \mathcal{C}$ so that $[J^w, J^y]$ is on the path from the fathomed node to the root node. In particular, fathomed nodes are explored.
- We say that node $[J^w, J^y]$ is *unexplored* if it is not discarded or explored. In this case, $[J^w, J^y]$ does not lie below a fathomed node or on the path to a fathomed node identified by any of the cuts. All the leaves in the subtree below an unexplored node correspond to pieces that are still in the set $\mathcal{F}$. An unexplored node has the potential to be a fathomed node, and finding a minimal cut identifying it would remove all such pieces from $\mathcal{F}$.

Furthermore, we will call an unexplored node an *open* node, if its parent is explored. Among the unexplored nodes, open nodes have the potential to generate cuts that remove the most pieces (leaves) in that part of the tree if they turn out to be fathomed nodes.

Let us demonstrate this in an example.

Example(Cont). *Using the same problem as in Example 3.2.1, consider the tree in Figure 3.1 with the cuts* (I_1^w, I_1^y) *and* (I_2^w, I_2^y)*. The fathomed nodes corresponding to these cuts are 3 and 5, which correspond to* $[J_1^w, J_1^y] = [\emptyset, \{1\}]$ *and* $[J_2^w, J_2^y] = [\{1\}, \{2\}]$*, respectively, according to our notation. Notice that* $[J_1^w, J_1^y] = [I_1^w, I_1^y]$*, but node* $[I_2^w, I_2^y]$ *does not exist in this tree.*

Nodes 8,9,...,15 are discarded nodes, while nodes 1,2,3 and 5 are explored nodes. Node 4,6 and 7 are unexplored nodes and therefore 4 is the only open node.

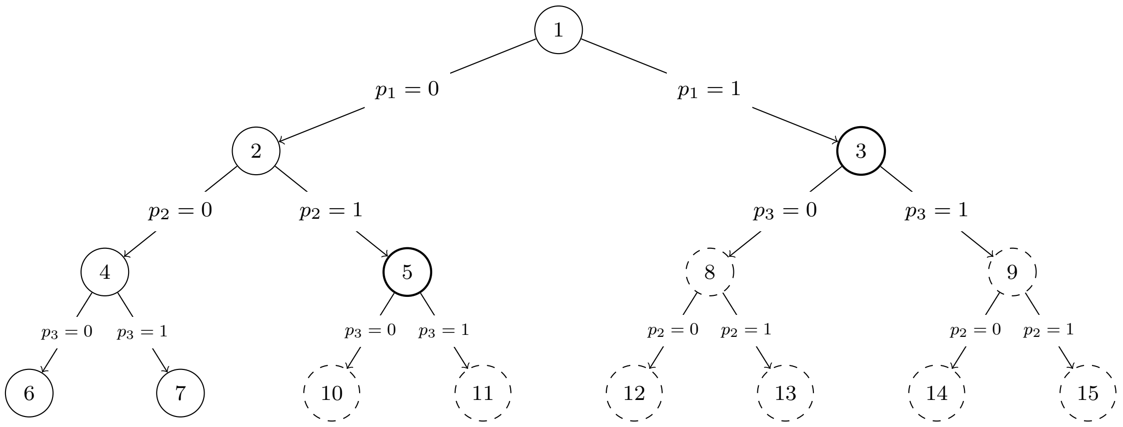


Figure 3.1. Classes of nodes in a B&B tree

Now, if we consider the tree in Figure 3.2 the fathomed nodes correspond to 5, 7, 9 and 13 and the open nodes are 6 and 12.

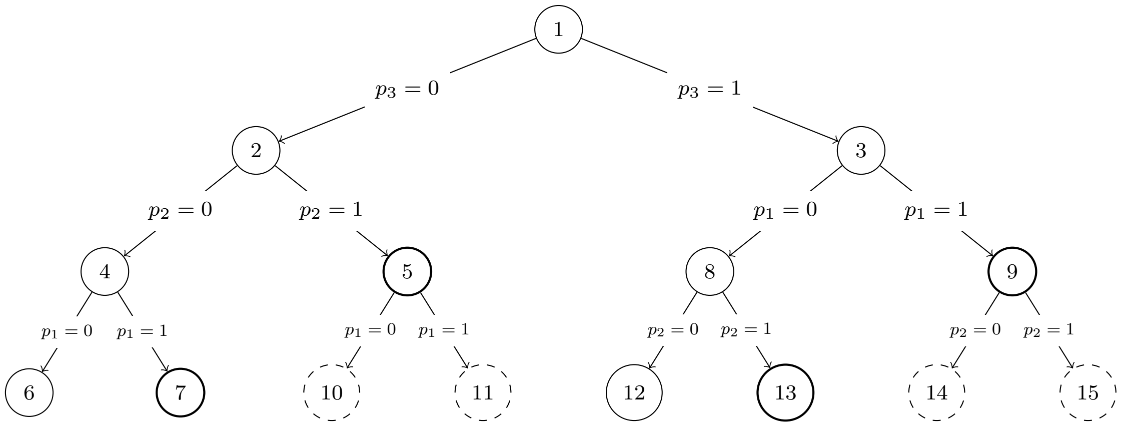


Figure 3.2. Classes of nodes in a B&B tree

3.2.1.6. Constructing a Tree. The initial version of the B&B-based logical Benders algorithm described in Section 3.2.1.3 assumed that the tree T was given and fixed throughout the procedure. Clearly, the amount of work, when measured in the number of (master problem) iterations, depends on the choice of the tree, and in particular on the number of fathomed nodes in that tree, since the algorithm can only terminate when they have all been identified.

We are usually not given a tree a priori that results in good performance for a given problem. The algorithm we propose here is related to the idea of information-based branching introduced in (Karzan et al., 2009). In their paper, the authors partially solve an IP problem via branch-and-bound, until a certain number of nodes are fathomed. Those nodes are identified with logical cuts, in a similar way as explained in Section (3.2.1). Then they resolve the problem, exploring different branching decisions by assigning, and comparing, several types of weights to components appearing in these cuts, which define the priority in the branching. Our approach follows the same spirit, but in a dynamic way. It constructs a "working tree" as a preliminary choice of the final tree in each iteration, taking into account the cuts that have been generated so far. It picks a new piece p, and then executes the cut sparsification Algorithm 6 to generate a new cut. It is important to note that this tree is never assembled explicitly, it is only a conceptual device that helps us to identify the input for Algorithm 6, i.e., the piece p and the path P.

Our algorithm for choosing p and P has two steps. It first (virtually) constructs a working tree and finds an open node in that tree, and then chooses a piece (or leaf) under the open node. For this, the working tree only needs to be defined up to the open and fathomed nodes. Algorithm 7 gives the framework for recursively defining a working tree. The choice of the branching variable j is guided by the following observations.

Algorithm 7 Procedure for generating working tree

$T = \textsc{Tree formation}(\{(I_k^w, I_k^y)\}_{k=1}^K,\ [J^w, J^y])$

INPUT: Cuts $\{(I_k^w, I_k^y)\}_{k=1}^K$. Root node $[J^w, J^y]$ of subtree to be constructed.

```
1: if $K = 0$ then
2:     Label $[J^w, J^y]$ as *open* node. return.
3: if $I_k^w \subseteq J^w$ and $I_k^y \subseteq J^y$ for some $k$ then        ▷ A cut is violated
4:     Label $[J^w, J^y]$ as *fathomed* node. return.
5: Label $[J^w, J^y]$ as *explored* node.
6: $\tilde{\mathcal{N}} = \{i \in \mathcal{N} : i \in (I_k^w \cup I_k^y) \setminus (J^w \cup J^y)$ for some $k = 1, \dots, K\}$.
7: if (Strategy 1) then
8:     $v_i := \#\{(I_k^w, I_k^y) : i \in I_k^w \cup I_k^y$ for some $k = 1, \dots, K\}$ for all $i \in \tilde{\mathcal{N}}$
9:     Select $j \in \operatorname{argmax}_{i \in \tilde{\mathcal{N}}}\{v_i\}$.
10: else (Strategy 2)
11:     Let $s_k = \#\{(I_k^w \cup I_k^y) \setminus (J_k^w \cup J_k^y)\}$ for all $k$.
12:     Let $\bar{s} = \operatorname{argmin}_k\{s_k\}$.
13:     $v_i := \#\{(I_k^w, I_k^y) : i \in I_k^w \cup I_k^y$ and $s_k = \bar{s}$ for some $k = 1, \dots, K\}$ for all $i \in \tilde{\mathcal{N}}$
14:     Select $j \in \operatorname{argmax}_{i \in \tilde{\mathcal{N}}}\{v_i\}$.
15: $\mathcal{I}^w := \{(I_k^w, I_k^y) : j \in I_k^y, k = 1, \dots, K\}$.
16: $\mathcal{I}^y := \{(I_k^w, I_k^y) : j \in I_k^w, k = 1, \dots, K\}$.
17: if $\#\mathcal{I}^w < \#\mathcal{I}^y$ then
18:     Construct subtree for $w$-child: Call TREE FORMATION($\mathcal{I}^w$, $[J^w \cup \{j\}, J^y]$).
19:     Construct subtree for $y$-child: Call TREE FORMATION($\mathcal{I}^y$, $[J^w, J^y \cup \{j\}]$).
20: else
21:     Construct subtree for $y$-child: Call TREE FORMATION($\mathcal{I}^y$, $[J^w, J^y \cup \{j\}]$).
22:     Construct subtree for $w$-child: Call TREE FORMATION($\mathcal{I}^w$, $[J^w \cup \{j\}, J^y]$).
23: return.
```

(1) An open node $[J^w, J^y]$ has the potential to become a fathomed node, namely when $\phi_P(J^w, J^y) \geq U$. In that case, a single iteration of the algorithm suffices to generate a cut to remove all leafs under $[J^y, J^w]$. So, in the most optimistic outcome, the remaining number of iterations is equal to the number of open nodes in the current B&B tree. Therefore, we aim at constructing a working tree with a small number of open nodes.

(2) For every open node there is a fathomed node at the same or lower level since its parent is explored. We therefore would like to generate a tree in which the fathomed nodes are high up in the tree. This way, if the open node can be immediately fathomed, the higher in the tree, the sparser it will be.

(3) The fathomed nodes that are identified by a given cut of cardinality l are at least l levels down. Therefore, in the best case possible, all fathomed nodes should be on the same level as the cardinality of the cut they are represented with.

(4) A complementarity that appears in no cut should not be chosen for branching. Otherwise, all fathomed and open nodes would be pushed down by one level, counteracting the goal described in observation (2) above.

Algorithm 7 proceeds recursively from the root node, calling itself for the generation of the subtrees after branching on a complementarity. At the beginning it is called with the root node and all available cuts, i.e., Tree formation($\{(I_k^w, I_k^y)\}_{k=1}^K$, $[\emptyset, \emptyset]$). The recursion is set up in a way so that only cuts are passed to the next level that are still relevant for the generation of the subtree rooted at the incoming node, in the sense that they could lead to a fathomed node. If no such cuts are available, the current node is marked as open in Step 2. On the other hand, if there is a cut that is violated by the incoming node (see also (3.12)), then the incoming node is marked as fathomed. If neither of those conditions are met, a branching complementarity needs to be chosen among those that have not yet been branched on (in the set $\tilde{\mathcal{N}}$).

We explore two different branching strategies. Strategy 1 chooses a complementarity that appears in most of the cuts that are still relevant. In this way, we hope to keep the level of the deepest fathomed node small, see observation 3 above.

The second strategy is based on the following observation: Suppose that there is a cut that includes only one complementarity that has not yet been branched on. Choosing this complementarity for branching would create one child node that can immediately be fathomed, and we effectively reduced the complexity of the remaining subtree by one complementarity in one shot. This observation motivates us to give priority to the sparsest cuts, i.e., those that include the smallest number $\bar{s}$ of remaining complementarities. Steps 11–12 choose one of the complementarities that appears most often in the sparsest cuts. In both strategies, if more than one complementarity satisfies the criteria above, then we choose the one that appears first in a fixed priority list. In our implementation, this list starts as the components, sorted in a descending order according to their complementarity violation, obtained from solving the root node. The list is updated as the tree becomes constructed along the path from the root to last chosen leaf.

Before calling the algorithm recursively to generate the subtree corresponding to the new child nodes, Steps 15 and 16 remove the cuts that are irrelevant in the respective subtree. Finally, the subtrees are generated.

3.2.1.7. Choosing an open node. Now that we specified how we define the working tree based on the cuts in a given iteration, we need to choose one of its open nodes in order to generate a new cut. We do this by executing a variation of the tree generation algorithm in the previous section.

In Algorithm 7, we set up the recursion in steps 17–22 so that the subtree with the fewest remaining cuts is generated first. While this makes no difference for the final working tree that is formed, it determines the order in which we encounter the open nodes. Since fewer cuts typically lead to subtrees in which the fathomed and open nodes are closer to its root, we are likely to find an open node quickly that is high up in the tree and has the potential to result in a sparse cut. The search procedure executes Algorithm (7) and returns the first open node that it encounters. Since the order of exploration prioritizes smaller subtrees, this depth-first search is likely to find an open node quickly, close to the root.

3.2.1.8. Choosing a piece. Once an open node $[J^w, J^y]$ has been determined, the algorithm requires a piece p and a path P as input for the cut generation procedure in Algorithm 6. To be consistent with the current working tree, the path between $[J^w, J^y]$ and the root node is chosen according to the working tree determined by Algorithm 7. It remains to choose a leaf corresponding to p that lies below $[J^w, J^y]$, and the path between that leaf and $[J^w, J^y]$.

Our choice is guided by the observation that Algorithm 6 removes a complementarity whenever $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq U$, where $(\tilde{I}^w, \tilde{I}^y)$ is the current trial cut in Algorithm 6. This suggests that it is beneficial to choose a leaf p that has a large value of ϕ_P or is infeasible.

To determine such a piece, we solve the relaxation (3.6) corresponding to $\phi_P(J^w, J^y)$. If $\phi_P(J^w, J^y) \geq U$, then the open node $[J^w, J^y]$ is actually a fathomed node, and we can generate a cut corresponding to it. Formally, we choose any piece p below $[J^w, J^y]$ and any path from that piece to $[J^w, J^y]$. The first iterations of Algorithm 6 will then remove all complementarities j_l that are not in $J^w \cup J^y$. In practice, we start Algorithm 6 from $(I^w, I^y) = (J^w, J^y)$ and avoid the unnecessary steps.

If $\phi_P(J^w, J^y) < U$, then the relaxation for $[J^w, J^y]$ must be feasible and has an optimal solution $(\tilde{x}, \tilde{w}, \tilde{y})$. We set, for all $i \in \mathcal{N} \setminus (J^w \cup J^y)$ (i.e., all complementarities that have not yet been branched on) $r_i = \min\{\tilde{w}_i, \tilde{y}_i\}$ and order them in decreasing order. We now choose the complementarities in that order, pick the w-branch toward the leaf if $w_i > y_i$ and the y-branch otherwise. If there is a tie and $w_i = y_i > 0$, we pick from the pre-chosen order, and if there is a tie with $w_i = y_i = 0$, we choose the side with the larger multiplier. This gives us the path and the leaf p.

3.3. Variations of the Logical Benders Algorithm

Recall that the logical Benders algorithm consists of two basic steps: Choose a piece $p \in \mathcal{F}$, and generate a cut (I^w, I^y) that excludes p from $\mathcal{F}$. For each of them, we consider different options, and we will compare their numerical performance in Section 3.4

3.3.1. Piece Selection

In (Bai et al., 2013), the authors found the new piece to explore using a black-box satisfiability problem (SAT) solver, part of the SIMULINK packages available in MATLAB. This approach has the disadvantage that it blindly selects a piece satisfying all cuts without considering the implicit information embedded in these cuts. The method proposed in Section 3.2.1.8 offers an alternative. Because it is based on a working tree it can choose a piece (or leaf) that has the potential to lead to a cut that is quite sparse. More importantly, this procedure generates cuts that are consistent with the existing cuts in the sense that it removes potentially many additional pieces from $\mathcal{F}$ instead of being redundant. We will refer to the piece selection procedure described in Section 3.2.1.8 as the "tree-guided piece selection".

3.3.2. Generation of Sparse Cuts

In this section we describe and discuss three different approaches to obtain sparse cuts which allow us to remove pieces that need not be explored. Cuts are constructed in a way such that any piece p removed by it is guaranteed to satisfy $\phi_P(p) \geq U$. Recall that a cut is an inequality determined by two disjoint subsets $I^w, I^y \subseteq \{1, \dots, n_c\}$ of the form

$$\sum_{i \in I^w} p_i + \sum_{i \in I^y} (1 - p_i) \geq 1, \tag{3.14}$$

hence sparsifying a cut translates into finding smaller sets $\bar{I}^w$ and $\bar{I}^y$ with a certificate that the primal relaxation $\phi_P(\bar{I}^w, \bar{I}^y) \geq U$.

The first approach, already introduced in Section 3.2.1.7, evaluates primal relaxations in a sequential order given by a "virtual" B&B tree. The second one, formulates an ℓ_1-norm minimization problem over the dual feasible region of the explored piece. Finally, we propose a third alternative that combines the advantages of the first two options. A detailed description of the methods follows.

3.3.2.1. Sequential Procedure. The general steps for the second sparsification method have been described in Algorithm 6. Given a piece p and a predefined order of indices $j_1 \to j_2 \to j_3 \to \ldots \to j_{n_c}$, we sequentially solve relaxations of this piece following the order. Let us illustrate this procedure through an example.

Example(Cont). *Using the same problem as in example* (3.2.1)

$$
\begin{array}{llrll}
 & \textit{minimize} & -2x_1 - x_2 - x_3 & & \\
 & \textit{subject to} & x_i & \leq 4 & i = 1,2,3 \\
(3.15) & & x_i & = y_i & i = 1,2,3 \\
 & & \sum_{k \neq i} x_k - 3x_i + 6 & = w_i & i = 1,2,3 \\
 & 0 \leq & y \perp w & \geq 0, &
\end{array}
$$

suppose we are given piece $p = (1,0,0)$*, that is* $I^w = \{2,3\}$ *and* $I^y = \{1\}$*, and the path from leaf to root is given by* $3 \to 2 \to 1$*. We are also given the optimal upper bound* $U = -9$*. Notice that* $\phi_P(p) = -6$*, so the piece could be immediately removed with the cut* (I^w, I^y)*, but this cut will remove only* p *and no other piece. The sequential sparsification procedure checks if we can do better. It first tries the relaxation* $\phi_P(\{2\},\{1\})$*, which has a value of* -6*, so now we found a better cut. It then continues with* $\phi_P(\emptyset,\{1\})$*, again with a value of* -6*, so the cut has been sparsified even further. It ends by trying out* $\phi_P(\emptyset,\emptyset)$ *(i.e., the full relaxation) with a value of* -16*. This means that complementarity* $y_1 = 0$ *cannot be relaxed so index* $i = 1$ *remains on* I^y*. The final cut is therefore* $(\emptyset,\{1\})$*. If we were given the piece* $p = (0,0,1)$*, with same path* $3 \to 2 \to 1$*, the obtained cut will be* $(\emptyset,\{3\})$ *due to symmetry of this example problem. This example is depicted in Figure* 3.3*. Dashed and bold lines represent accepted and rejected relaxations of complementarities, respectively. Bold circles correspond to fathomed nodes. Everything grayed out are nodes and branches not observed yet.*

Clearly, by construction, every cut (I^w, I^y) obtained from this procedure is minimal.

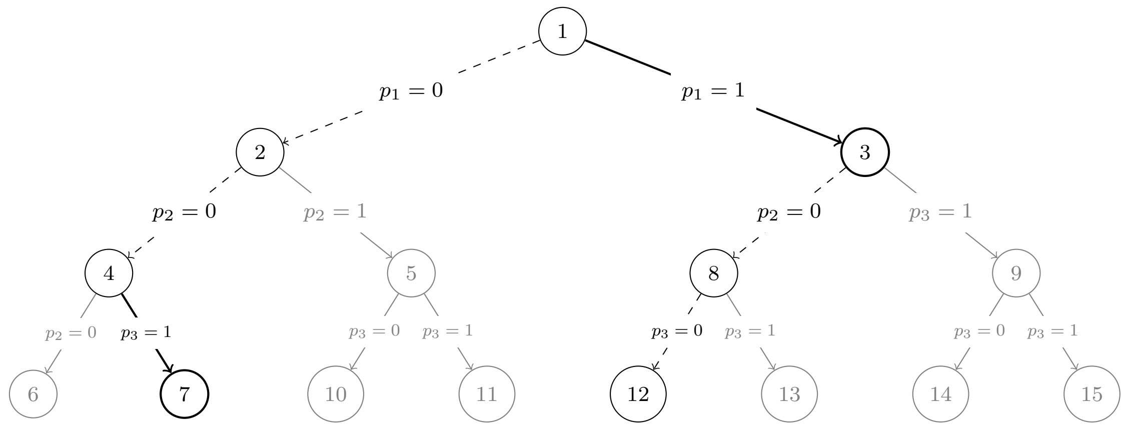


Figure 3.3. Sequential sparsification in a B&B tree

This method can take advantage of a situation in which the tree-guided piece selection finds an open node $[I^w, I^y]$ with $\phi_P(I^w, I^y) \geq U$. It is clear that Algorithm 6 removes all indices along the path from the leaf to $[I^w, I^y]$, and so we save iterations by running Algorithm 6 with an abbreviated path that starts at $[I^w, I^y]$ instead of the leaf.

As mentioned earlier, this method guarantees that the resulting cut (I^w, I^y) is *minimal*, but it could happen that node $[I^w, I^y]$ does not exist in the current working tree, which can be interpreted as if some branching decisions in the tree were not necessary, and hence some fathomed nodes appear in levels lower than desired. This motivates the procedure for the tree to be updated in every iteration, so that fathomed nodes (and consequently open nodes) might be moved higher in the tree.

3.3.2.2. ℓ_1-norm Minimization. Let us first assume that a *feasible* piece p has been selected from $\mathcal{F}$ by the piece selection procedure. As mentioned in Section 3.1.3.2, cuts are closely related to the dual variables λ^w and λ^y in (3.7). A valid cut is obtained with $I^w = \{i : \lambda_i^w > 0\}$ and $I^y = \{i : \lambda_i^y > 0\}$. Hence, and alternative to find sparse cuts is to solve the following linear program:

$$\underset{(\mu_I,\,\mu_E,\,\lambda^w,\,\lambda^y)\in\Omega_D(\bar{I}^w,\bar{I}^y)}{\text{minimize}} \sum_{i\in\bar{I}^w}\lambda_i^w + \sum_{i\in\bar{I}^y}\lambda_i^y, \tag{3.16}$$

where $\bar{I}^w = \bar{I}^w(p)$ and $\bar{I}^y = \bar{I}^y(p)$. A solution $(\hat{\lambda}^y, \hat{\lambda}^w)$ of this problem provides a valid cut $(\hat{I}^w, \hat{I}^y)$ with $I^y := \{i : \hat{\lambda}_i^y > 0\}$ and $I^w := \{i : \hat{\lambda}_i^w > 0\}$, and it corresponding relaxation has a value greater or equal than U. This condition is forced in the $-b_I^T\mu_I + b_E^T\mu_E \geq U$ constraint, in the description of $\Omega_D(I^w, I^y)$. The objective function is interpreted as the ℓ_1-norm of variables λ^w and λ^y which intends to steer their components towards zero. In a strict sense, to find a sparsest solution we should be using the ℓ_0-norm instead in the objective, but then problem (3.16) becomes highly non-convex. Following (Candes et al., 2008), we enhance the solution of (3.16) with an iterative re-weighting scheme. Given a set of initial weights $\omega^0 \in \mathbb{R}_+^{|I^w|}$ and $\gamma^0 \in \mathbb{R}_+^{|I^y|}$, we set the iteration counter $k = 0$ and solve the problem

$$\underset{(\mu_I,\,\mu_E,\,\lambda^w,\,\lambda^y)\in\Omega_D(I^w,I^y)}{\text{minimize}} \sum_{i\in I^w}\omega_i^k\lambda_i^w + \sum_{i\in I^y}\gamma_i^k\lambda_i^y, \tag{3.17}$$

and with the optimal solution $(\hat{\mu}_I^k, \hat{\mu}_E^k, (\hat{\lambda}^w)^k, (\hat{\lambda}^y)^k)$ of (3.17) we update the weights $\omega_i^{k+1} = \frac{1}{\max\{\varepsilon,(\hat{\lambda}^w)_i^k\}}$ and $\gamma_i^{k+1} = \frac{1}{\max\{\varepsilon,(\hat{\lambda}^y)_i^k\}}$, set $k = k+1$ and resolve. The ε parameter serves to ensure the weights are well defined in the circumstance and λ becomes zero In our implementations, ε was set to 10^{-6}. This iterative procedure gives a variable which was close to zero in one iteration a larger weight in the next one and therefore it is more likely to be pushed to zero. The method ends when two consecutive iterates are equal. This procedure can be interpreted, as explained in (Candes et al., 2008), as sequentially minimizing linearizations of $\sum_i \log(\lambda_i)$ over Ω_D and, hence, the closer to zero a variable is, the steepest its slope becomes. In our experiments, this procedure converges very quickly, requiring no more than 6 iterations even in the largest instances.

Under the circumstance that the tree-guided piece selection finds an open node $[I^w, I^y]$ with $\phi_P(I^w, I^y) \geq U$, we start the procedure directly with the index sets I^w and I^y, provided the corresponding relaxation $\Omega_D(I^w, I^y)$ is feasible.

Now consider the case in which the primal of the selected piece $p \in \mathcal{F}$ is infeasible. In case the dual problem (3.7) is feasible, we can still follow the procedure above. However, if (3.7) is infeasible, we formulate the ℓ_1-norm minimization problem over the homogenized set $\Omega_{D_0}(I^w, I^y)$. Once the weighted iterative heuristic finishes in an unbounded ray $(\hat{\mu}_I, \hat{\mu}_E, \hat{\lambda}^w, \hat{\lambda}^y)$, we check whether the corresponding primal relaxation became feasible. If not we set $I^y = \{i : \hat{\lambda}^y_i > 0\}$ and $I^w = \{i : \hat{\lambda}^w_i > 0\}$ as the index sets of the newly found sparse cut. Otherwise, we continue the ℓ_1-norm minimization on the corresponding dual of the relaxed primal.

Although this method has the nice feature that it requires the solutions of just a few LPs, it has no guarantees that it will produce a minimal cut. Consider the following very simple example:

$$\begin{array}{rl} \min_{\lambda} & \lambda_1 + \lambda_2 \\ \text{subject to} & \lambda_1 + 2\lambda_2 \geq 3 \\ & 2\lambda_1 + \lambda_2 \geq 3 \\ & \lambda \geq 0. \end{array} \tag{3.18}$$

If the initial weights are set to one, the procedure will converge in the second iteration to $(1, 1)$, although the sparsest solutions are $(3, 0)$ and $(0, 3)$.

This gives way for a third procedure to generate sparse cuts: the Hybrid method.

3.3.2.3. Hybrid method. Both presented sparsification procedures have their own sets of advantages and downsides. The ℓ_1-norm minimization approach finds cuts quickly by iteratively solving linear programs, which can even be hot-started in a primal simplex algorithm since the feasible region does not change. But there is no guarantee that the resulting cuts are minimal. On the other hand, the sequential procedure finds minimal cuts, but it requires n_c LP solves, making it too costly as the dimension of the complementarities increases, even if solved with the dual simplex method to use hot starts as we did in our implementation.

We can exploit the advantages of both methods in a hybrid sparsification procedure. The approach consists of two steps: (1) the ℓ_1-norm minimization sparsification routine is called to compute a cut (I^w, I^y) given by $I^w := \{i : \hat{\lambda}_i^w > 0\}$ and $I^y := \{i : \hat{\lambda}_i^y > 0\}$, where $\hat{\lambda}^w$ and $\hat{\lambda}^y$ are the outputs of the iterative re-weighting scheme, and (2) the sequential sparsification steps along the path P are executed, assuming that the complementarities in $I^w \cup I^y$ can be directly removed in Algorithm 6 without computations.

The underlying idea is that the ℓ_1-norm minimization already removes many complementarities from the cut, so that the sequential procedure does not need to go through all components from the leaf to the root. A formal outline of the procedure is described in Algorithm 8

Algorithm 8 Hybrid Sparsification

```
1: Input: Piece $p \in \{0,1\}^{n_c}$, a path $P$.
2: Set $(I^w, I^y) = (I^w(p), I^w(p))$
3: Solve problem (3.17) to obtain solution $(\hat{\lambda}^w, \hat{\lambda}^y)$.        ▷ Solving (3.17)
4: Set $I^w := \{i : \hat{\lambda}_i^w > 0\}$ and $I^y := \{i : \hat{\lambda}_i^y > 0\}$.
5: Let $j_1 \leftarrow j_2 \leftarrow \ldots \leftarrow j_L$ be the subset of the path $P$, such that $\{j_i\}_{i=1}^L = I^w \cup I^y$.
6: for $l = 1, 2, 3, \ldots, L$ do
7:     Set $(\tilde{I}^w, \tilde{I}^y) = (I^w \setminus \{j_l\}, I^y \setminus \{j_l\})$. ($(\tilde{I}^w, \tilde{I}^y)$ is the parent of $(I^w, I^y)$)
8:     Solve (3.6) for $(\tilde{I}^w, \tilde{I}^y)$
9:     if $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq U$ then
10:        Set $(I^w, I^y) \leftarrow (\tilde{I}^w, \tilde{I}^y)$
11: end for
12: Return $(I^w, I^y)$ as minimal cut.
```

3.3.3. Remarks

We finish this section by pointing out some observations regarding the power of the presented cut sparsification procedures.

- Throughout most of this chapter, as stated in section 3.2.1.2, we assumed we are given the optimal value U of the LPCC and that the logical Benders algorithm is used to certify optimality. It is only in this setting where we can claim that the cuts generated by the sequential or hybrid methods are actually minimal cuts, where minimality is considered with respect to the actual optimal value of (3.1). This is clear by how these methods were designed. Under the circumstance that the provided incumbent is not the optimal solution, the methods described in the previous sections can be adopted. The only difference is that the upper bound needs to be updated whenever a piece $p \in \mathcal{F}$ is selected such that $\phi_P(p) < U$. Whenever these updates occur the cuts generated so far could potentially be sparsified further, by replacing the newly found incumbent as value to be comparing against, in Ω_D and Algorithm 6. Keep in mind, though, that if resparsification of a cut is called, for example, in the sequential or hybrid method, the path from leaf to root (or subset of the path, to be more precise) to be considered will be from the most recent "virtual" tree and not the one used when the corresponding cut was generated. In this chapter, we do not consider resparsification and its effectiveness remains to be investigated later.

- If the tree T is fixed throughout the whole algorithm, that is, the sequences of branching decisions from every leaf to the root do not change, then the order in which the pieces (leaves) are selected in the procedure is irrelevant. The set of cuts at the end of the method is independent of this selection.

3.4. Numerical Results

This section contains various numerical experiments in order to show the effectiveness and robustness of the proposed method. We solved instances with complementarity dimension n_c ranging from 25 to 1,000 and compared three different variants of the logical Benders methods: (1) selection a piece $p \in \mathcal{F}$ via the black-box SAT solver combined with splitting sparsification ("Splitting"), (2) Tree-guided piece selection with ℓ_1-norm sparsification ("ℓ_1"), and (3) Tree-guided piece selection with hybrid cut generation procedures ("Hybrid").

By "Splitting" sparsification we refer to the method introduced in Bai et al. (2013). An initial cut (I^w, I^y) is obtained from the optimal solution of the dual piece (3.7), as described in section 3.1.3.2. The dual vectors, λ^y and λ^w, are sorted in descending order and the top half of the non-zero components of each vector are selected to form two sets $\tilde{I}^w$ and $\tilde{I}^y$. If $\phi_P(\tilde{I}^w, \tilde{I}^y) \geq U$, then $I^w := \tilde{I}^w$ and $I^y := \tilde{I}^y$ and the sparsification method continues. Otherwise, the returned cut is (I^w, I^y).

As for the tree guided piece selection, it is set up so that if the open node $[I^w, I^y]$ is "fathomable", meaning that its primal value lies above U, we follow Algorithm 5. That is, we keep the cut (I^w, I^y).

The tested instances were generated following the steps described in Hu et al. (2008). In their paper they used a slightly different structured LPCC, namely

$$\begin{array}{llrl} & \underset{x\geq 0, y}{\text{minimize}} & c^Tx + d^Ty & \\ (3.19) & \text{subject to} & Ax + By & >= f \\ & & 0 \leq \; y \perp q + Nx + My & \geq 0. \end{array}$$

The procedure to generate instances is described below. After generating instances of this type, converting to the structure of (3.1) is straightforward.

Instance Generator[1]

INPUT: Dimensions n, m and k, for x, y and f, respectively. Density value s.

OUTPUT: Matrices c, d, A, B, f, M, N and q.

```
1: Generate $x \sim N_n(0,1)$. Set $x = |x|$.
2: Generate $y \sim N_m(0,1)$. Set $y_i = 0$ if $y_i < 0$.
```

```
3:  Generate $c \sim U_n(0,1)$ and $d \sim U_m(1,3)$
4:  Generate $A \sim U_{k\times n}(0,1)$ and $B \sim U_{k\times m}(0,1)$ with density $s$.
5:  Generate $r \sim DU(\{0,1,\ldots,m\})$
6:  Let $s_M = \frac{2000-m}{m^2}$
7:  Generate $E \sim U_{r\times(m-r)}(-1,1)$ with density $s_M$.
8:  Generate $d_1 \sim U_r(0,2)$ and $d_2 \sim U_{m-r}(0,2)$.
9:  Let $D_1 = \mathrm{diag}(d_1)$ and $D_2 = \mathrm{diag}(d_2)$.
10: Let $M = \begin{pmatrix} D_1 & E \\ -E & D_2 \end{pmatrix}$
11: Generate $N \sim U_{m\times n}(-1,1)$.
12: Generate $q \sim U_m(-20,-10)$
13: Let $f = Ax + By - |\varepsilon|$, with $\varepsilon \sim N_k(0,1)$.
```

The dimensions of our test instances follow the same patterns as in Hu et al. (2008): $[n,m,k] = [100,100,90]$, $[300,300,200]$ and $[500,500,450]$. For the largest test set we chose $[n,m,k] = [1000,1000,200]$.

During our experiments we noticed some patterns which seemed to make a problem harder or easier. For example, if coupling constraints were present (i.e. $B \neq 0$), the method required more iterations on average to be solved, for the same dimension numbers. Therefore, as part of our examination, we consider both the cases with and without coupling constraints (by either setting $B \sim U_{k\times m}(0,1)$ or $B = 0$ in the Instance Generator).

[1]Notation: $N_n(0,1)$, refers to n draws from a standard normal distribution; $U_n(a,b)$, n draws from a uniform distribution in the (a,b) interval, and $DU(\{a_1,\ldots,a_m\})$, a uniform draw from the set $\{a_1,\ldots,a_m\}$.

As explained in Section 3.2.1.2, we obtain a candidate optimal solution by solving problem (3.10) first and then attempt to provide a certificate of optimality by solving the "outer problem". In this setting, we can immediately terminate if the solution of (3.10) has an optimal solution lower than the outer relaxation. Therefore, we chose the big-M parameter in (3.10) in a way such that the outer relaxation still provides a strict lower bound to the optimal solution of (3.10).

The algorithms were coded in MATLAB R2016a, with calls to CPLEX 12.6 to solve all MILP and LP problems. The experiments were run on a Windows 10 64-bit machine with Intel(R) Core(TM) @2.7GHz and 16.0 Gb RAM. For a fair comparison, all experiments were run with a single thread.

We compare the number of main iterations (number of times a piece $p \in \mathcal{F}$ is selected) and total CPU time required by the different variants of the logical Benders method. It is important to point out that CPU time must not be taken too seriously, since it is not clear if hot-starts, in the MATLAB/CPLEX inter-phase, works as efficient as it could. We are not reporting the CPU time it takes to solve the big-M formulation (3.10) described in Section 3.2.1.2, since that time is the same for every method. Additionally the total number of simplex iterations is provided. We consider the number of main iterations as a reflection of the quality of both the pieces selected and the cut generated, so it is our main metric to observe. Still, generating strong cuts and finding strong new pieces to explore comes at the expense of CPU time. We complement the tables with charts to give a visual representation of the trade-offs between runtime and number of iterations. We analyze the experiments from smallest to largest. Unless otherwise noted, the instances include coupling constraints.

The two first sets of instances ($n_c = 25$ and $n_c = 50$) are using a big-M value equal to 10, and its metrics are shown in tables 3.1 and 3.2, and figures 3.4 and 3.5. We see that for $n_c = 25$, all instances are solved within 1 second. Notice though that the number of main iterations is always the smallest for the hybrid approach. This is an expected (and desired) behavior among all instances and sizes that we will observe, since the hybrid method is built so that generated cuts are as sparse as they can get, and so they translate into requiring less cuts to explore or discard all pieces.

On instances of size 50, we observed that some instances (instances 3, 8, 9 and 10) did not finish within a 3 hour threshold, for the SAT-Splitting combination, represented by the dashed gray line in the plots. Therefore, we incorporated an alternative method consisting of the tree-guided piece selection replacing the SAT solver, alongside the Splitting sparsification. This method is represented by the continuous yellow line and will be further on referred to as “Splitting” (i.e. we no longer select a piece $p \in \mathcal{F}$ with the SAT solver for larger complementarity dimensions). It can still be seen that Hybrid seems to be the dominating method in most of the instances, in terms of number of main iterations and time.

At a first glance, switching the piece selection method should not make any difference, but it actually does. The main advantage the tree-guided method has over SAT is that it checks whether the recently found open node can be fathomed immediately. In that case, no piece is selected and a cut is immediately generated, removing every single leaf under this node from subsequent piece selections. In the SAT selection case, a piece must be chosen every time, and the sparsification method (any sparsification method whatsoever) might fail to identify the same cut, and could therefore not remove all leaves as in the tree-guided case. This way, the overall algorithm will require more iterations to explore the full set of complementarity pieces. We observed in our experiments that, on instances where the number of iterations of SAT piece selection differed from Tree-guided by a large margin, both with Splitting sparsification, the percentage of iterations where the encountered open node was fathomable was greater than 70%.

We noticed also that instance 3 was a “hard” problem to solve. Apparently the culprit of this problem being hard lies on the fact that the incumbent found while solving the MILP (3.10) was not optimal for the original problem (3.1), meaning that the upper bound had to be updated during the procedure which implied some cuts were not necessarily minimal. This was also observed in the larger instances: whenever the MILP (3.10) did not return the optimal value for (3.1), the Splitting method had trouble solving it. Finally, it is worth mentioning some anomalies that we observed in these experiments. Although SAT-Splitting did not solve 4 out of 10 instances of size $n_c = 50$, in the remaining ones it showed small CPU times. Contrary to our intuition, on instances 1, 4 and 7, it showed higher number of iterations, but less CPU time than the Splitting (with tree-guided piece selection). When we checked the CPU times at a lower level, we noticed that most of the time differences were concentrated in the sparsification procedure, rather than in the piece selection process.

For the next two instances (sizes $n_c = 100$ and $n_c = 300$), we increased the value of the big-M parameter to 100. Here we start observing the actual effectiveness of our proposed cut generation methods compared to the Splitting method. The graphs on Figure 3.6 illustrate how the Splitting method struggles to maintain the number of main iterations low for some instances with 100 complementarities. This becomes more evident in Figure

Table 3.1. Metrics for 25 complementarities

Instance	Main iterations			CPU times (s)			Simplex Iterations		
	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1
1	8	13	12	1.38	0.17	0.82	404	576	245
2	3	3	3	0.33	0.04	0.20	116	141	108
3	1	1	1	0.08	0.01	0.06	77	36	77
4	4	9	6	0.32	0.07	0.30	299	689	174
5	7	6	7	0.59	0.04	0.41	340	246	220
6	14	30	19	1.16	0.23	1.57	1202	3433	1025
7	3	3	3	0.21	0.01	0.21	239	178	181
8	2	5	2	0.13	0.07	0.09	127	309	121
9	2	3	2	0.13	0.02	0.13	113	158	108
10	2	4	2	0.12	0.02	0.11	109	109	100

Figure 3.4. Number of iterations and CPU times for problems with 25 complementarities

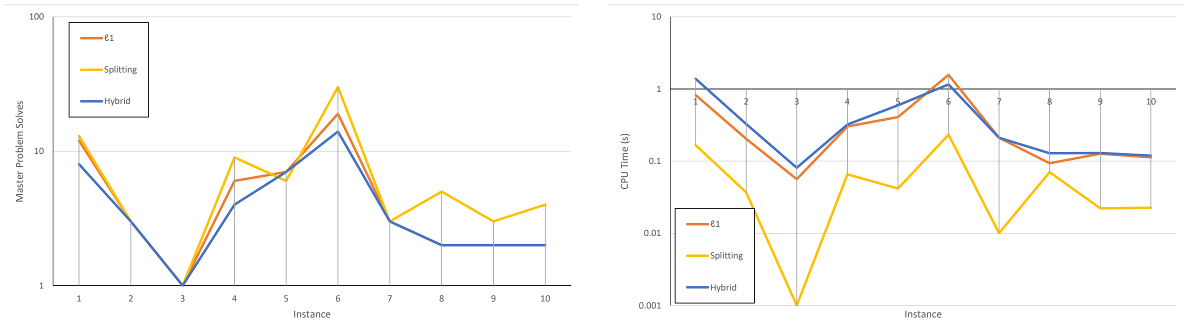
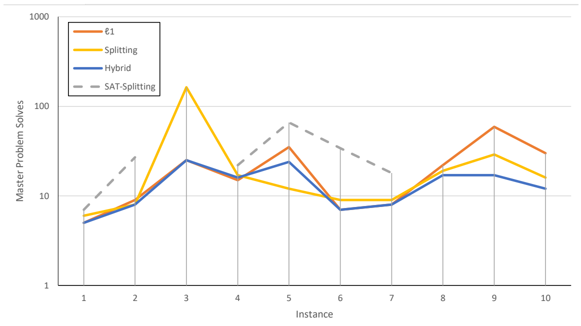
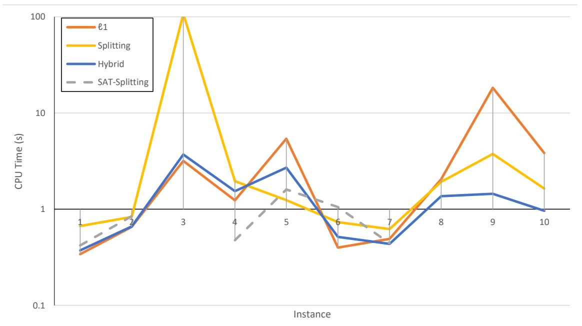


Table 3.2. Metrics for 50 complementarities

Instance	Main iterations			CPU times (s)			Simplex Iterations		
	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1
1	5	6	5	0.37	0.67	0.34	821	408	642
2	8	8	9	0.66	0.83	0.66	1646	780	1172
3	25	163	25	3.70	107.11	3.18	6705	6620	4032
4	16	17	15	1.54	1.96	1.23	2048	1119	1867
5	24	12	35	2.70	1.24	5.41	4341	1329	2801
6	7	9	7	0.52	0.73	0.40	906	606	664
7	8	9	8	0.44	0.62	0.49	771	367	623
8	17	19	22	1.37	1.93	2.05	3330	1620	2363
9	17	29	59	1.45	3.76	18.29	3289	1808	2168
10	12	16	30	0.96	1.63	3.82	2437	1182	1629

Figure 3.5. Number of iterations and CPU times for problems with 50 complementarities





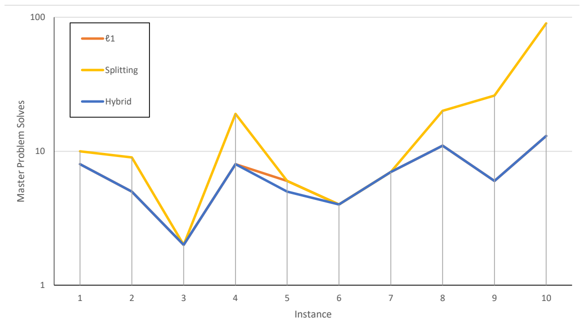
3.7, where the Splitting method fails to certify the optimal solution within 500 iterations in two instances, whereas both Hybrid and ℓ_1 manage to keep this metric below 50. As in the smaller dimensions, the difficult instances were those in which the starting incumbent was not the global optimum. An intuitive reason is that Hybrid is more capable of dealing with non-optimal incumbents, since it treats each complementarity component individually. On the contrary, Splitting rejects a split if it lies below the incumbent (so the higher the incumbent, the more likely the rejection is), and therefore it could fail to find a cut slightly less sparse than the one obtained by accepting the split. We also see that Hybrid does not provide any extra benefit compared to ℓ_1 in terms of iterations as they differ by at most one. This could mean that ℓ_1 already found minimal cuts and, hence, calling the sequential procedure was unnecessary. We verified the generated cuts on each iteration and saw that the difference on sparsity level between Hybrid and ℓ_1 was no larger than 2. With respect to CPU times, on the instances where all methods perform on par (in terms of iterations), Splitting is always the fastest among all three. This is expected, since in the "worst" case it only requires $\frac{\log(n_c)}{\log(2)}$ LP solves, per iteration. [2]

[2]We write "worst" within quotes because if this case happens, the cut would have sparsity at most 2, i.e., very sparse.

Table 3.3. Metrics for 100 complementarities

Instance	Main iterations			CPU times (s)			Simplex Iterations		
	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1
1	8	10	8	0.40	0.30	0.58	3408	2219	2294
2	5	9	5	0.19	0.20	0.21	1539	863	1271
3	2	2	2	0.07	0.09	0.07	542	256	534
4	8	19	8	0.52	0.74	0.42	5066	4128	3663
5	5	6	6	0.18	0.16	0.22	1577	699	1197
6	4	4	4	0.13	0.07	0.10	806	350	617
7	7	7	7	0.19	0.15	0.16	1388	613	1113
8	11	20	11	0.37	0.49	0.34	2669	2048	1946
9	6	26	6	0.30	0.73	0.23	3299	3637	2492
10	13	90	13	0.52	7.08	0.44	4527	4130	2890

Figure 3.6. Number of iterations and CPU times for problems with 100 complementarities



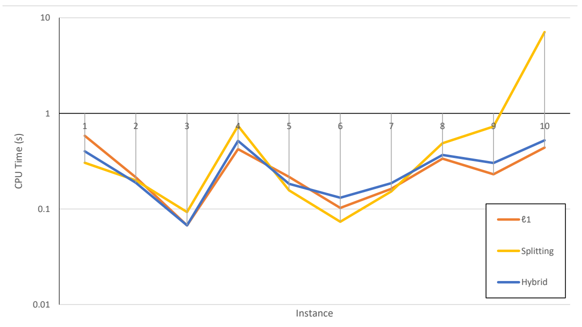
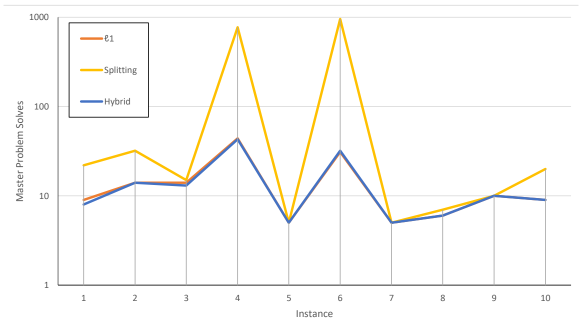
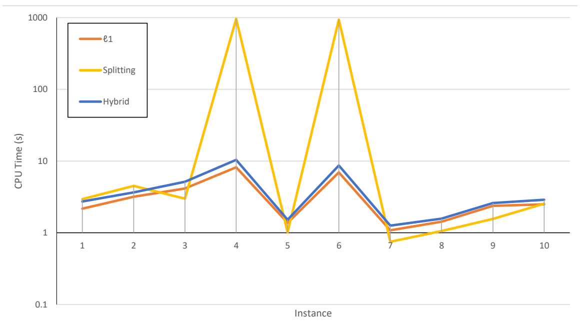


Table 3.4. Metrics for 300 complementarities

Instance	Main iterations			CPU times (s)			Simplex Iterations		
	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1
1	8	22	9	2.72	0.56	2.95	15019	13400	11233
2	14	32	14	3.67	0.91	4.5	18400	20327	12413
3	13	15	14	5.13	0.72	2.99	32256	14742	25223
4	43	774	44	10.39	0.92	963.57	58607	62231	37107
5	5	5	5	1.52	0.44	1.01	8591	3500	6811
6	32	956	31	8.66	0.9	937.55	53802	71741	34494
7	5	5	5	1.26	0.15	0.75	8486	3218	6566
8	6	7	6	1.57	0.45	1.06	8987	4464	6355
9	10	10	10	2.6	0.59	1.56	14715	6100	11599
10	9	20	9	2.88	0.75	2.54	16457	11787	11388

Figure 3.7. Number of iterations and CPU times for problems with 300 complementarities





For the two final sets of instances, the big-M value was set to 1,000. Again we observe the same pattern as before for the $n_c = 500$ case (Table 3.5 and Figure 3.8). That is, instances where the incumbent was not globally optimal lead to a drastic increase of iterations for the Splitting method (instances 1, 4 and 7). Hybrid and ℓ_1 still dominate Splitting across every instance in terms of iterations, and this dominance is reversed when it comes to CPU times, over instances where the solution was found by the MILP 3.10.

The last set of instances, with 1,000 complementarities, did not have coupling constraints. We removed them from the problem, since otherwise no method was able to certify optimality within a 3 hour frame. Once removed, the MILP managed to find the optimal solution on all tested instances. This is reflected in the fact that the worst case in terms of number of iterations does not exceed 100, as opposed to the worst cases of the instances with $n_c = 500$, which went up to 300 or higher. This situation clearly benefits the splitting method since it dominates all but one instance in CPU time. Similar to most of the instances from $n_c = 100$ going up, Hybrid does not seem to provide any benefit over ℓ_1, and both still have the least number of iterations.

Table 3.5. Metrics for 500 complementarities

Instance	Master Problem solves			CPU times (s)			Simplex Iterations		
	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1
1	15	779	19	18.23	666.03	19.87	62248	237100	61179
2	3	3	3	5.71	2.62	4.77	18716	7005	16586
3	24	38	24	31.13	12.17	23.28	87553	33059	73341
4	32	436	36	48.16	176.53	37.79	132792	132965	127914
5	9	39	9	8.74	13.92	6.79	25065	38018	21385
6	5	5	5	7.73	2.91	5.59	21829	8500	21079
7	36	352	36	21.90	145.16	17.44	63170	65090	50046
8	16	16	16	7.69	3.11	6.69	23432	8973	20947
9	16	24	15	25.28	15.00	19.16	87414	50092	68912
10	16	21	16	20.26	10.22	18.95	74530	32813	65303

Figure 3.8. Number of iterations and CPU times for problems with 500 complementarities

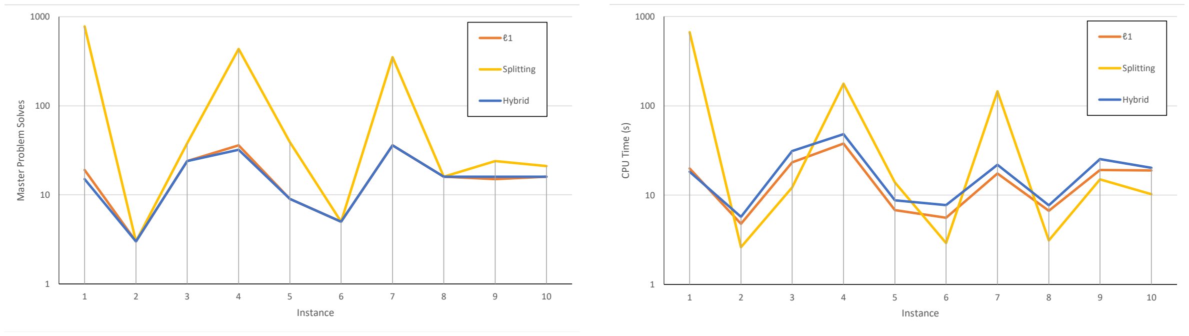
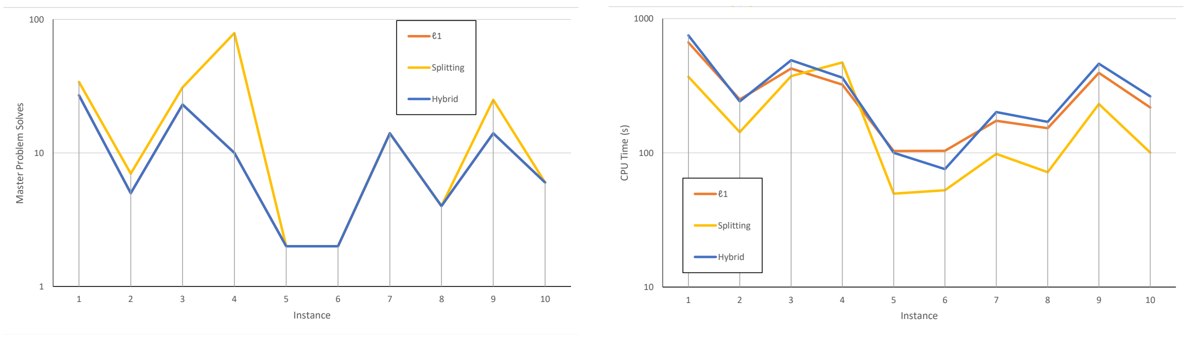


Table 3.6. Metrics for 1,000 complementarities

Instance	Master Problem solves			CPU times (s)			Simplex Iterations		
	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1	Hybrid	Splitting	ℓ_1
1	27	27	34	749.52	368.42	663.94	149942	68631	127612
2	5	5	7	241.65	142.6	249.7	52276	31915	46253
3	23	23	31	490.41	372.29	424.92	93523	61562	78646
4	10	10	79	362.85	471.6	322.66	77212	78384	68079
5	2	2	2	100.01	49.67	103.19	22888	11156	21650
6	2	2	2	75.68	52.62	103.46	22606	10679	21660
7	14	14	14	201.1	98.4	173.3	36638	16783	33025
8	4	4	4	170.42	71.71	152.72	32888	15603	32323
9	14	14	25	461.24	231.38	394.66	90894	46970	78922
10	6	6	6	263.82	100.32	217.81	46178	21848	43533

Figure 3.9. Number of iterations and CPU times for problems with 1,000 complementarities



3.5. Conclusions and Future Research

We introduced a different interpretation to the logical Benders approach for solving LPCCs from the perspective of branch-and-bound methods and exploited this relationship to provide a new way to select pieces and generate cuts. Numerical results showed that this method is more robust in the sense of keeping a consistent number of iterations along instances of the same size, even when the optimal solution is not provided, but at the expense of more time spent in the sparsification process. We also showed that the tree-guided approach for the piece selection outperforms the black-box SAT solver, which indicates that piece selection is indeed an important driver for the performance of logical Benders.

Future research considers the extension of this branch-and-bound framework to Binary Constrained Quadratic Programs with Linear Complementarity Constraints (BCQPCCs). Some preliminary ideas are given next.

3.5.1. Binary Constrained Quadratic Programs with Linear Complementarity Constraints

The straightforward way to convey binary variables via complementarities is by writing a constraint of the form

$$0 \leq z \perp \mathbf{1} - z \geq 0$$

and treat it as an additional set of complementarities, but this is a very inefficient formulation. An alternative to this is explained below.

To keep the same format of (3.3), we split the variable $y = (y^c, y^b)$ and the piece (p^c, p^b) to represent the complementarity and binary pieces. We define the complementarity and binary sets, $I^w := \bar{I}^w(p^c)$, $I^y := \bar{I}^y(p^c)$, $I^0 = \bar{I}^0(p^b) := \{i : p_i^b = 0\}$ and $I^1 = \bar{I}^1(p^b) := \{i : p_i^b = 1\}$, in a similar way: The extension to BCQPCCs of each piece can then be written as follows.

$$
\begin{array}{llrll}
\phi_P(p) = & \underset{x,y,w}{\text{minimize}} & g^T x + \frac{1}{2} x^T Q x & & \\
& \text{subject to} & A_I x + B_I^c y^c + B_I^b y^b + C_I w & \leq b_I & (\mu_I) \\
& & A_E x + B_E^c y^c + B_E^b y^b + C_E w & = b_E & (\mu_E) \\
& & w_i & \leq 0,\ i \in I^w & (\lambda_i^w) \\
& & y_i^c & \leq 0,\ i \in I^y & (\lambda_j^y) \\
& & y^b & \leq \mathbf{1} & (\lambda^b) \\
& & y_i^b & \leq 0,\ i \in I^0 & (\lambda_i^0) \\
& & -y_i^b & \leq -1,\ i \in I^1 & (\lambda_i^1) \\
& & w,\ y & \geq 0. &
\end{array}
\tag{3.20}
$$

With a similar argument as in section 3.1.3 we can construct a cut (I^0, I^1, I^w, I^y) of the form:

$$
\sum_{i \in I^w} p_i^c + \sum_{i \in I^y} (1 - p_i^c) + \sum_{i \in I^0} p_i^b + \sum_{i \in I^1} (1 - p_i^b) \geq 1. \tag{3.21}
$$

If the primal was infeasible, we already discussed that the homogenized dual must be unbounded with λ^w and λ^y extended with zeros as in (3.9). We can replicate this with λ^0

and λ^1.

$$\begin{aligned}
\phi_{D_0}(p) = \quad & \underset{\mu_I,\, \mu_E,\, \lambda^w,\, \lambda^y,\, \lambda^0,\, \lambda^1}{\text{maximize}} && -b_I^T \mu_I + b_E^T \mu_E + \mathbf{1}^T \lambda^1 - \mathbf{1}^T \lambda^b \\
& \text{subject to} && -A_I^T \mu_I + A_E^T \mu_E && = 0 \\
& && -B_I^{cT} \mu_I + {B_E^c}^T \mu_E - \lambda^y && \leq 0 \\
& && -B_I^{bT} \mu_I + {B_E^b}^T \mu_E - \lambda^b - \lambda^0 + \lambda^1 && \leq 0 \\
& && -C_I^T \mu_I + C_E^T \mu_E - \lambda^w && \leq 0 \\
& && (p^c)^T \lambda^w + (\mathbf{1} - p^c)^T \lambda^y && = 0 \\
& && (p^b)^T \lambda^0 + (\mathbf{1} - p^b)^T \lambda^1 && = 0 \\
& && \mu_I,\ \lambda^w,\ \lambda^y,\ \lambda^b,\ \lambda^0,\ \lambda^1 && \geq 0
\end{aligned} \tag{3.22}$$

The sixth constraint implies $\mathbf{1}^T \lambda^1 = (p^b)^T(\lambda^1 - \lambda^0)$, therefore, an unbounded ray $(\hat{\mu}_I, \hat{\mu}_E, \hat{\lambda}^w, \hat{\lambda}^y, \hat{\lambda}^0, \hat{\lambda}^1, \hat{\lambda}^b)$ implies

$$b_E^T \hat{\mu}_E - b_I^T \hat{\mu}_I - \mathbf{1}^T \hat{\lambda}^b + (p^b)^T(\hat{\lambda}^1 - \hat{\lambda}^0) > 0, \tag{3.23}$$

Hence any binary variable p^b which violates

$$b_E^T \hat{\mu}_E - b_I^T \hat{\mu}_I - \mathbf{1}^T \hat{\lambda}^b + (\hat{\lambda}^1 - \hat{\lambda}^0)^T p^b \leq 0 \tag{3.24}$$

will have an infeasible primal problem. Inequality (3.21) can be rearranged in the following way. Let us define

$$\hat{r} := b_E^T \hat{\mu}_E - b_I^T \hat{\mu}_I - \mathbf{1}^T \hat{\lambda}^b \tag{3.25}$$

$$\hat{g} := \max\left\{-\hat{\lambda}^1 + \hat{\lambda}^0, 0\right\} \tag{3.26}$$

$$\hat{h} := \max\left\{\hat{\lambda}^1 - \hat{\lambda}^0, 0\right\} \tag{3.27}$$

So, (3.24) can be rewritten as

$$\hat{r} \le \hat{g}^T p^b - \hat{h}^T p^b \tag{3.28}$$

If we add $\mathbf{1}^T \hat{h}$ on both sides and noticing the resulting left hand side is positive due to (3.23), then (3.24) ends up being

$$1 \le \left(\frac{\hat{g}}{\hat{r} + \mathbf{1}^T \hat{h}}\right)^T p^b + \left(\frac{\hat{h}}{\hat{r} + \mathbf{1}^T \hat{h}}\right)^T (\mathbf{1} - p^b). \tag{3.29}$$

By defining $\bar{g} := \min\left\{\frac{\hat{g}}{\hat{r}+\mathbf{1}^T\hat{h}}, 1\right\}$ and $\bar{h} := \min\left\{\frac{\hat{h}}{\hat{r}+\mathbf{1}^T\hat{h}}, 1\right\}$, this cut can be strengthened further:

$$1 \le \bar{g}^T p^b + \bar{h}^T (\mathbf{1} - p^b). \tag{3.30}$$

Therefore, the combined cut to remove p^c and p^b is

$$\sum_{i \in I^w} p_i^c + \sum_{i \in I^y} (1 - p_i^c) + \sum_{i \in I^0} \bar{g}_i p_i^b + \sum_{i \in I^1} \bar{h}_i (1 - p_i^b) \ge 1. \tag{3.31}$$

A nice feature of this new cut is that the sums keeps the same structure as the original cuts, so sparsification still consists on finding subsets of I^0, I^1, I^w and I^y such that the primal relaxation lies above the current incumbent.

A similar argument and computation can be done when the primal has a finite optimal solution which lies strictly above the upper bound U. We can add a constraint which forces the linearization of the objective to be smaller than U, making the new primal infeasible. We can therefore proceed in the same way as with the homogenized dual to create the cut.

To fit the BCQPCC in our logical Benders/B& B framework two main questions need to be addressed.

(1) How to modify the piece selection method to capture the strengthened cuts. Since these new cuts have weights smaller than one, the relationship between fathomed nodes and cuts is not as clear as in Section 3.2.1.3. It seems that Algorithm 7 should still work with a slight modification on steps 15 and 16, where the index sets are updated.

(2) How to modify the ℓ_1-norm formulation (3.16) to include the constraint which forces the quadratic objective to be greater than the incumbent. Notice that now this formulation will have a quadratic constraint, so now performing the iterative procedure is not as efficient as when the objective was linear. One idea is to force the linearization of the quadratic objective around the optimal primal solution to be greater than the incumbent. Although this constraint is weaker than using the quadratic term, it recovers the ℓ_1-norm formulation with only linear constraints.

Since this linearization makes the primal infeasible, we can always start solving the homogenized formulation of (3.16).

CHAPTER 4

Stochastic Linear Programs with Complementarity Constraints

4.1. A Global-Local Method for Stochastic Linear Bilevel Programs with Applications on the Network Newsvendor Problem

4.1.1. Introduction

This chapter presents a numerical study of a global-local algorithm to solve Stochastic Linear Bilevel Problems (SLBLP) of the form

$$\begin{aligned} \min_{x,y} \quad & c^T x + \mathbb{E}_\xi[d^T y^\xi] \\ \text{s.t.} \quad & x \in \mathcal{X} \\ & y^\xi \in \arg\min_z \{h^T z : z \in \mathcal{C}(x,\xi)\}, \quad \xi \in \Omega, \end{aligned} \tag{4.1}$$

where $x \in \mathbb{R}^s_+$, $y^\xi \in \mathbb{R}^{ds}_+$, for all $\xi \in \Omega$ and Ω represents the sample space of the (continuous) random variable ξ. The sets $\mathcal{X}$ and $\mathcal{C}(x,\xi)$ are assumed to be polyhedra for every x and ξ. We specifically consider the Bilevel Network Newsvendor problem, an extension to the classical inventory problem. By reformulating (4.1) as a Stochastic Mathematical Program with Complementarity Constraints (SMPCC), the structure of the algorithm can be divided in two parts: (1) obtain an initial candidate solution x by globally solving a small, subsampled version of the SMPCC, (2) steer x towards good approximations of global solutions on a much larger sample.

We start this chapter with a review of bilevel programs, mathematical programs with equilibrium constraints, and mathematical programs with complementarity constraints, and their corresponding linear and stochastic versions.

4.1.1.1. Overview. Bilevel programming problems (BLP) are hierarchical mathematical problems with two distinctive variables, x and y, where the vector y is to be chosen as an optimal solution $y = y(x)$ of an optimization problem parametrized in x, that is,

$$\min_{x,y}\{F(x,y) : G(x,y) \le 0, y \in \Psi(x), x \in \mathbb{R}^{n_x}\} \tag{4.2}$$

where

$$\Psi(x) = \arg\min_{y}\{f(x,y) : g(x,y) \le 0, y \in \mathbb{R}^{n_y}\}. \tag{4.3}$$

Problems (4.2) and (4.3) are known as the *leader's* and the *follower's* problems, respectively. We will refer to the problem defining Ψ as the *inner level* problem, and to (4.2) as the *outer level* problem. The feasible region of the inner level problem, for any given x, will be denoted $\mathcal{C}(x) := \{y \in \mathbb{R}^{n_y} : g(x,y) \le 0\}$.

Bilevel programs have many applications in several fields, for example, transportation, economics, energy systems. For a complete review of applications, please refer to (Dempe, 2003). Bilevel programs are a particular case of the more general Mathematical Programs with Equilibrium Constraints (MPEC), where the feasible region for y is expressed through a variational inequality; that is, the point-to-set mapping is defined as $\Psi(x) := \{y \in \mathcal{C}(x), H^T(x,y)(y - \hat{y}) \ge 0, \text{ for all } \hat{y} \in \mathcal{C}(x)\}$. In the situation where the mapping H corresponds to a gradient of some C^1 function, that is, $H = \nabla_y f$, then the MPEC is equivalent to a BLP.

If the inner level problem presents suitable constraint qualifications, we can state the first order optimality conditions which leads to a Mathematical Program with Complementarity Constraints (MPCC). In that case, $\Psi(x) := \{(y,\mu) : \nabla_y f(x,y) + \mu^T \nabla_y g(x,y) = 0, \mu \geq 0, g(x,y) \leq 0, \mu^T g(x,y) = 0\}$. Notice that now $\Psi(x)$ consists of both primal and dual variables y and μ of the inner level problem. It is important to emphasize the need for constraint qualifications when formulating a BLP as an MPCC, since it is not true, opposed to the MPEC case, that BLP is a particular case of MPCC. In (Dempe and Dutta, 2012), the authors show an example where the only (global) solution to a BLP cannot be obtained by solving its MPCC counterpart. The culprit of this discordance is, precisely, the lack of constraint qualifications in the inner level problem at the optimal x. When dealing with linear bilevel problems (LBLP), i.e., F, G, f and g are linear, then LBLP can indeed be formulated as a Linear Program with Complementarity Constraints (LPCC). This relationships between different programs allows us to justify the method to be presented. In what follows, we will assume all functions are affine linear and that H is the gradient mapping of f in the MPEC characterization (its corresponding problem is denoted LPEC).

Allowing uncertainty in the parameters defining functions F, G, f and g, we obtain an SLBLP. The setting of the stochastic version we are interested in is the one expressed in (4.1). The structure of this SLBLP has *here-and-now* outer level and *wait-and-see* inner level decisions. This means, from a leader-follower perspective, that the followers make their decision with full information, i.e., they know the value of both x and ξ, while the leader must make her decision before the realization of the random variable. Any optimal solution (x, y) of this problem is known as an equilibrium, meaning that no agent can benefit by unilaterally changing her decision. From now on, the prefix "S" on every acronym will stand for "Stochastic".

Many of the theoretical results for SLBLP are analyzed through SLPECs. These were first introduced by (Patriksson and Wynter, 1999), where the authors provide a simple chart which illustrates equivalences, generalizations and particular cases, between the problems mentioned above. We replicate this chart below as a visual aid. The relationship $A \subset B$ implies that A is a special case of B.

$$\begin{array}{ccccccc}
 & & & & [\text{LPCC}] & \subset & [\text{SLPCC}] \\
 & & & & \cup & & \cup \\
[\text{SBLP}] & \supset & [\text{BLP}] & \supset & [\text{LBLP}] & \subset & [\text{SLBLP}] \\
\cap & & \cap & & \cap & & \cap \\
[\text{SMPEC}] & \supset & [\text{MPEC}] & \supset & [\text{LPEC}] & \subset & [\text{SLPEC}]
\end{array}$$

In the same paper, conditions for the existence of optimal solutions of SMPECs were stated in the absence of outer level coupling constraints (i.e. $G(x, y) := G(x)$). Later, in Evgrafov and Patriksson (2004), these conditions were generalized to SMPECs with coupling constraints, and imposing measurability assumptions on the uncertain parameters. The standard approach to solve stochastic problems is to perform a finite, but large, number of draws from Ω and replace the (continuous) expectation with a sample average. This is known as the Sample Average Aproximation (SAA). In the works of (Shapiro, 2006; Shapiro and Xu, 2008) and (Xu and Jane, 2011), convergence results were stated with regards to stationary points of the SAA to the set of stationary points of the true SMPEC. In Shapiro (2006) and Xu and Jane (2011), the results are for here-and-now formulations, while in Shapiro and Xu (2008), the authors prove them for the inner level wait-and-see version. In this last paper, results over the rate of convergence of the optimal values are also stated. These convergence results motivate the numerical experiments presented in this chapter. Namely, that it is justified to search for local optima of the SAA formulations since they should be, for a sufficiently large sample, close to some local solution of the original SLPEC. In the following section we describe the standard methods to solve these kinds of problem and introduce the method we use in our experiments.

4.2. Solving SLBLP via SLPCC

For the remaining of this chapter we will assume that $\mathcal{C}(x,\xi) := \{y : Ay \leq b^\xi - Bx, y \geq 0\}$, where $A \in \mathbb{R}^{k\times n_y}$, $B \in \mathbb{R}^{k\times n_x}$, and $b^\xi \in \mathbb{R}^k$ is a vector dependent on the random variable ξ. That is, the outer level variable x and ξ only have an effect on the right-hand side of the inner level polyhedron. Under the circumstance that the inner level problem has multiple solutions, two different formulations are usually implemented: "Optimistic" where the inner level solution chosen is the one that benefits the leader the most, and "Pessimistic", where the worst possible option for the leader is selected. The work presented in this chapter assumes the optimistic setting. We can equivalently write the primal-dual optimality conditions for linear programs on every scenario to obtain the following reformulation as an SLPCC.

$$
\begin{aligned}
\min_{x,y,\mu} \quad & c^T x + \mathbb{E}_\xi[d^T y^\xi] \\
\text{s.t.} \quad & x \in \mathcal{X} \\
& \left.\begin{array}{lll} 0 \leq \mu^\xi & \perp & b^\xi - Ay^\xi - Bx \geq 0 \\ 0 \leq y^\xi & \perp & h + A^T\mu^\xi \geq 0 \end{array}\right\} \xi \in \Omega,
\end{aligned}
\tag{4.4}
$$

where μ^ξ corresponds to the dual vector of the lower level subproblem.

Several efforts have been made towards solving general stochastic MPCCs. In their paper, Lin and Fukushima (2005) present a regularization method for a here-and-now SLPCC, assuming the random variables have a discrete support, where the authors replace the complementarity condition by a smooth approximation parameterized by ε. They prove that as ε approaches zero, the accumulation points of the solutions are feasible for the SLPCC. Moreover, if constraint qualifications are satisfied in the limit point (not considering the complementarity constraints), then it is strongly stationary. Moving to the continuous random variable case, Xu (2006) provides convergence results for methods which combine smoothings of NCP functions and an implicit function $y(x)$ to represent the complementarities as a system of non-linear equations. They show that the solutions to these smooth reformulations converge to a stationary point of the SMPCC when the smoothing paramenter goes to zero. Birbil et al. (2006) outline a set of sufficient conditions for which (weakly, C, strongly)-stationary points of the SAA version of the MPCC gets almost-sure convergence to (weakly, C, strongly)-stationary points of its stochastic counterpart. In (Lin et al., 2009), the authors extend the previously mentioned results, for SAA convergence and implicit smoothing, to wait-and-see formulations. A detailed survey of these findings can be found on (Lin and Fukushima, 2010).

When derived from an LBLP, the SLPCC can be reformulated as a mixed-integer LP (MILP), by incorporating a binary variable z_i for each complementarity component i and a sufficiently large parameter M. Usually the value of M is not known before-hand, so the selection of this big-M needs to be done carefully. It must be large enough so it does not rule out optimal solutions of the inner level problem, but not too large so it may lead to computational difficulties. Although solving the MILP formulation guarantees finding a global solution to the SLPCC, doing so is prohibitively expensive when the number of scenarios is even moderately large. Hence, the alternative is to look for good local optimal solutions which is already a challenging problem due to the highly non-convex structure of the feasible region.

Besides the implicit smoothing techniques mentioned earlier, several other methods for finding local solutions of MPCCs have been developed. To mention a few, we can identify those which solve the problem directly with a sequential quadratic programming (SQP) approach (Fletcher* and Leyffer, 2004; Fletcher et al., 2006). There are also some works which attempt to solve the LPCC via interior point methods, such as (Leyffer et al., 2006), which are generally more efficient for large scale problems. Drifting from these two classical approaches for non-linear programming, we can identify: (1) (Leyffer and Munson, 2007), which combines a linearization of the MPCC to search for descent directions, and a filter method. The authors prove the method to be globally convergent to B-stationary points that, as seen in Chapter 2, Proposition 2.1.2, coincide with local optima in LPCCs, (2) (Fang et al., 2012), a simplex-like method generalized to LPCCs, where the pivoting conditions guarantee that the iterates always remain feasible. They also show a way to deal with degeneracy and present a procedure to avoid cycling when the iterate is B-stationary, but not strongly stationary. Finally, Jara-Moroni et al. (2016) solve the LPCC by means of the difference-of-convex algorithm (DCA) and propose enhancements to each presented decomposition, as described in Chapter 2. One important feature of the enhancement of the piece-wise linear decomposition is the clever selection of subgradients of the concave part when non-strict complementarities are present. This selection allowed the method to escape weakly stationary points efficiently.

4.2.1. A Global-Local Method

As mentioned in Section 4.1.1.1, the standard approach to solve SLPCC is via the SAA, that is, a large, finite sample $\Omega^N := \{\xi_1, \ldots, \xi_N\}$ is drawn from Ω, and the discretized SLPCC is solved,

$$
\begin{aligned}
\min_{x,y,\mu} \quad & c^T x + \frac{1}{N}\sum_{i=1}^{N}[d^T y^{\xi_i}] \\
\text{s.t.} \quad & x \in \mathcal{X} \\
& \left.\begin{aligned} 0 \le \mu^{\xi_i} \quad \perp \quad & b^{\xi_i} - Ay^{\xi_i} - Bx \ge 0 \\ 0 \le y^{\xi_i} \quad \perp \quad & h + A^T \mu^{\xi_i} \ge 0 \end{aligned}\right\} i = 1, \ldots, N.
\end{aligned}
\tag{4.5}
$$

We will refer to this problem as the SAA-LPCC. It was also mentioned that stationary points of this SAA-LPCC get closer to the stationary points of the original stochastic problem as the sample size N grows larger, but solving such a large problem is hard. Our global-local method consists of drawing a subsample $\Omega^n \subset \Omega^N$, of size $n << N$ sufficiently small so that we can solve to global optimality a smaller SLPCC discretized according to the subsample Ω^n. This smaller SLPCC will be referred to as SS-LPCC (SS, as in subsample). It provides an outer level candidate decision x, for which we can compute optimal solutions (y^ξ, μ^ξ), of the inner level problem, for every scenario $\xi \in \Omega^N$. This vector $(x, (y^\xi, \mu^\xi)_{\xi \in \Omega^N})$ can be used as the starting point for the local solver (e.g. the DCA) for (4.5), with the intention that it finds a good approximation for global optima.

Under the circumstance that the big-M of the MILP formulation is not known, we could use any of several other methods to compute global solutions. Bard and Moore (1990) suggest a branch-and-bound procedure, where the branching is performed on the complementarities. This idea is later extended in Yu (2011) and Yu et al. (2018) to a branch-and-cut scheme. The authors also provide an extensive review of different complementarity branching strategies, based on traditional strategies from the IP literature. Other methods rely on cut generation techniques, for example, Hu et al. (2008) and its extension described in Chapter 3, both based in a logical Benders framework. This approach has also been studied for Q(uadratic)PCCs in Bai et al. (2013).

4.3. The Network Newsvendor Problem

In this section we introduce the problem used for our numerical experiments. It is an extension of the classical inventory problem, known as the newsvendor problem, to multiple vendors and sources of demand. Later in this chapter we present the "alternating weights" technique tailored for this problem.

Consider a set of s store nodes, S, and a set of d demand nodes, D, spread in a 2-dimensional plane. For each store the inventory level must be chosen for a specific product for which demand is uncertain. Once the inventory is decided, demand reveals itself at the demand nodes and must be satisfied either by the supply on any of the s stores, or by a competing company with unlimited supply. This problem will be denoted as the Network Newsvendor Problem (NNP). It can be viewed as a second step of the well known capacitated facility location problem where a set of locations has already been chosen and the remaining decision is the capacity on each of these selected locations. As in the traditional newsvendor problem, there is a cost associated to purchasing a unit of the product (adding it to the inventory) and a price related to the selling of it.

In the bilevel formulation of this problem, the leader acts as the agent making the decision about the inventory level of each store. The follower is a fictitious aggregate customer who solves a transportation problem between customers and facilities to satisfy demand at a minimum cost. Each unit of capacity added to a store $j \in S$ incurs a cost of c_j. If a unit of the product is sold from facility j, the leader gains a unit revenue of r_j. The shipping cost to send one unit from facility j to customer $i \in D$ is given by c_{ij} and the demand of customer i, which we will assume to be stochastic, is given by ξ_i. We distinguish the unit capacity cost c_j from the shipping cost c_{ij} by the number of subscripts utilized. We assume that the values of ξ_i are independent and uniformly distributed within a closed bounded interval Ω_i, and hence $\Omega = \prod_{i \in D} \Omega_i$. The decision of the inventory level on location j is represented by a non-negative continuous variable x_j. The number of products that customer i buys in store j is a non-negative continuous variable y_{ij}. The fictitious customer at demand location i also has the option to buy from competing supplier at a cost ρ_i. The corresponding amount is denoted w_i. The mathematical formulation of the inner problem for fixed x and scenario ξ is therefore:

$$
\begin{aligned}
\min_{y,w} \quad & \sum_{\substack{i \in D \\ j \in S}} (c_{ij} + r_j) y_{ij} + \sum_{i \in D} \rho_i w_i \\
\text{s.t.} \quad & \sum_{j \in S} y_{ij} + w_i \geq \xi_i, \quad i \in D \\
& \sum_{i \in D} y_{ij} \leq x_j, \quad j \in S \\
& y, w \geq 0.
\end{aligned}
\tag{4.6}
$$

The overall bilevel NNP formulation is as follows:

$$\begin{aligned}
\min_{x,y,w} \quad & \sum_{j\in S} c_j x_j - \mathbb{E}_\xi \left[\sum_{\substack{i\in D\\ j\in S}} r_j y^\xi_{ij} \right] \\
\text{s.t.} \quad & 0 \le x_j, \quad j \in S
\end{aligned}$$

$$\tag{4.7} \left.\begin{aligned}
y^\xi \in \arg\min\nolimits_{y,w} \quad & \sum_{\substack{i\in D\\ j\in S}} (c_{ij} + r_j) y_{ij} + \sum_{i\in D} \rho_i w_i \\
\text{s.t.} \quad & \sum_{j\in S} y_{ij} + w_i \ge \xi_i, \quad i \in D \\
& \sum_{i\in D} y_{ij} \le x_j, \quad j \in S \\
& y, w \ge 0
\end{aligned}\right\} \xi \in \Omega.$$

The first constraint of the inner problem requires demand to be satisfied for all customers, and the second ensures that a facility cannot sell more units than what is in its inventory. Notice that adding w as a variable makes the subproblem feasible regardless of the upper level decision x and the scenario ξ. Also, since the competing supplier has unlimited supply, the variable y_{ij} will always be zero whenever $\rho_i < c_{ij} + r_j$, for candidate stores $j \in S$. Writing the KKT conditions on the lower level problems leads to the following

SLPCC:

$$\min_{x,y,w,z} \quad \sum_{j\in S} c_j x_j - \mathbb{E}_\xi\left[\sum_{\substack{i\in D\\ j\in S}} r_j y_{ij}^\xi\right]$$

$$\text{s.t.} \qquad 0 \le x_j, \quad j \in S$$

$$\left.\begin{array}{llll} 0 \le \sum_{j\in S} y_{ij}^\xi - \xi_i + w_i^\xi & \perp & \lambda_i^\xi \ge 0 & i \in D \\ 0 \le x_j - \sum_{i\in D} y_{ij}^\xi & \perp & \mu_j^\xi \ge 0 & j \in S \\ 0 \le c_{ij} + r_j - \lambda_i^\xi + \mu_j^\xi & \perp & y_{ij}^\xi \ge 0 & i \in D, j \in S \\ 0 \le \rho_i - \lambda_i^\xi & \perp & w_i^\xi \ge 0 & i \in D \end{array}\right\} \xi \in \Omega. \tag{4.8}$$

We can reduce the number of complementarities by observing, via a simple computation, that any feasible point to (4.8) necessarily satisfies $\sum_{j\in B} y_{ij}^\xi + w_i^\xi = \xi_i$. Therefore, we can replace the first block of complementarity constraints by these equations, for all i, keeping the $\lambda_i^\xi \ge 0$ constraint.

Suppose we have drawn a large sample Ω^N from Ω and that we are now solving this deterministic, but large LPCC. We have also drawn a subsample $\Omega^n \subset \Omega^N$ for which the corresponding LPCC is solved to global optimality. As mentioned in the previous section, this problem can be reformulated as an MILP by incorporating a sufficiently large parameter M, but numerical issues may arise if it is set to high. Fortunately, the NNP allows for explicit bounds when the support Ω is bounded. It is easy to show that the lower level problem always admits at least one optimal solution within the following bounds:

- $\lambda_i^\xi \le \rho_i$, for all $i \in D$.
- $y_{ij}^\xi, w_i^\xi \le M_\Omega := \sum_{i\in D} (\max_{\xi\in\Omega_i} \xi)$, for all $i \in D$.

- $\mu_j^\xi \leq M_\rho := \max_{i \in D} \rho_i$, for all $j \in B$.

In fact, the first bound corresponds to the last constraint in (4.8). The second bound is direct from the fact that the customers will never purchase beyond the maximum possible demand scenario. The third one can be derived by looking at the third and fourth constraints of (4.8). These bounds are not necessarily tight, except for the first one which is achieved whenever $w_i^\xi > 0$.

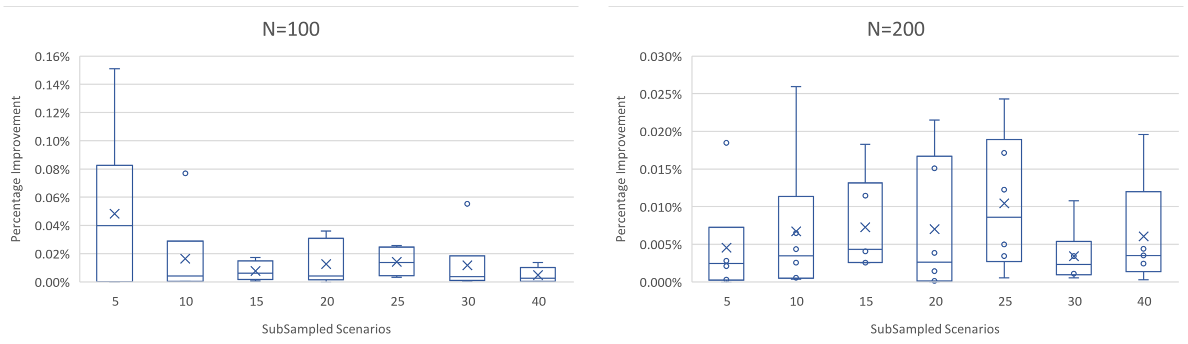
After obtaining the upper level decision x, from the global solution of the SS-LPCC, we construct the full vector $\left(y^\xi, w^\xi, \lambda^\xi, \mu^\xi\right)_{\xi \in \Omega^N}$, by solving the transportation problem (4.6) for every scenario $\xi \in \Omega^N$. By construction, it is clear that the vector $\left(x, (y^\xi, w^\xi, \lambda^\xi, \mu^\xi)_{\xi \in \Omega^N}\right)$ is feasible for 4.8. This full vector becomes the starting point for the local solver.

We created two instances of the NNP, with $d = 30$ and $d = 50$. Both instances have $b = 4$. Nodes are distributed in a $[0, 1] \times [0, 1]$ box by first partitioning the space into 4 equal squares, forming a 2×2 grid. Store nodes are located in the center of each square. Demand nodes are distributed randomly following a Latin Hypercube Sampling. Costs c_{ij} represent the euclidean distance between demand i and store j. Upper level costs and revenues are generated uniformly at random in the $[3.5, 4.5]$ and $[4.8, 5.2]$ intervals, respectively. The parameter ρ_i is selected such that every store has at least one demand node willing to buy from them, that is, for every $j \in S$, there exists $i \in D$ such that $c_{ij} + r_j \leq \rho_i$. When solving the SS-LPCC we choose the corresponding big-M as defined above. The means θ_i of the demand on each node i are drawn uniformly at random between 50 and 100. The scenarios are then generated by drawing $\xi_i \sim U((1-\alpha)\theta_i, (1+\alpha)\theta_i)$, independently, for each i, where α equals 0.1 in the $d = 30$ instance and 0.25 when $d = 50$.

The global solutions of the subsamples were solved in CPLEX and the local solver used was the DCA method with the piece-wise linear penalty function, with the reduced costs enhancement, as described in Chapter 2, Section 2.2.1. Experiments were run on a Linux Cluster with five 20-core 2.4GHz Intel Xeon processors and 4 x 256GB RAM. All runs used a single thread, and we set a time limit of 5 hours for the global solves and 2 hours for the DCA solves.

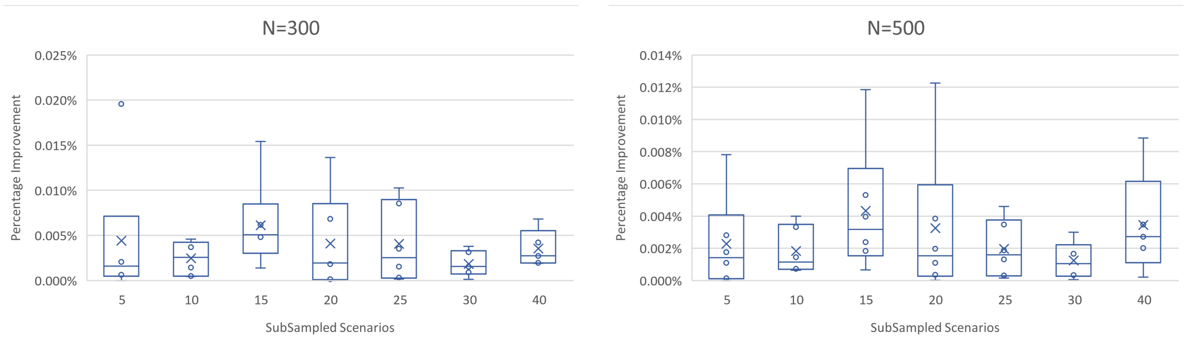
The box plots in Figures 4.1 and 4.2 show the percentage improvement of the solution obtained with the DCA, for different SAA sample sizes $N = 100, 200, 300$ and 500, as compared to the objective value of the starting points, on the SAA-LPCC, found by globally solving the SS-LPCCs of sizes $n = 5, 10, 15, 20, 30, 40$, over 10 runs for each size. We can see that there is very little improvement, if any, on all experiments.

Figure 4.1. Box plot for percentage improvement, $N = 100$ and $N = 200$



The reason is that the starting solution is already very close to some strongly stationary point, which then the DCA method finds and is unable to escape from. The key observation which motivated the enhancement presented in the next section is that the inner level problem tends to show degeneracy in their solutions for some scenarios, that is, the number of non-zero allocations, on variables y and w, is less than $s + d - 1$. If those solutions

Figure 4.2. Box plot for percentage improvement, $N = 300$ and $N = 500$



were unique[1], it implies that the corresponding dual has multiple solutions. Therefore, there is a chance the strongly stationary point found by the DCA had the "incorrect" dual variables chosen. We start the presentation of the "alternating weights" technique with an example to explain what we mean by labeling a solution as incorrect.

4.3.1. Alternating Weights Technique

Consider the following toy example of an NNP that we will use to motivate the "alternating weights" technique. There are 4 demand nodes and 2 stores, with shipping costs c_{ij} set as shown in Figure 4.3. The unit cost c_j and revenue r_j are 0.2 and 0.5, respectively. The threshold parameter ρ_i is the same for all demand nodes and equal to 3.5. Hence, the arcs not shown in the graph can be interpreted as satisfying $c_{ij} + r_j > \rho_i$ and will never be used by the customers.

Suppose only two scenarios are possible: $\xi^1 = (20, 20, 10, 30)$ and $\xi^2 = (5, 5, 30, 20)$ and assume scenario 2 is drawn in the SS-LPCC to be solved to global optimality. In this scenario, the global optimum for the upper level decision maker is to set $x_1 = 40$

[1]This might still happen if the solution is not unique, but is no longer a necessary condition

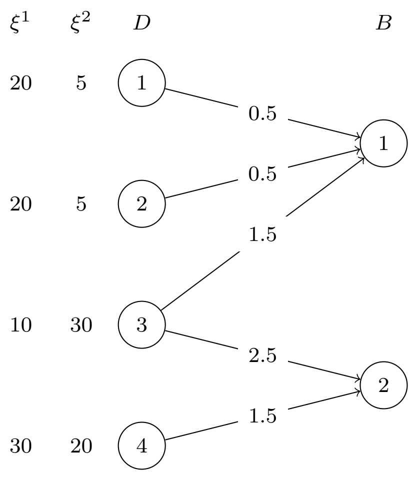


Figure 4.3. Toy example for NNP

and $x_2 = 20$ with the optimal assignments of the inner transportation problem being $y^2_{11} = y^2_{21} = 5$, $y^2_{31} = 30$ and $y^2_{42} = 20$. Keeping x fixed, the optimal assignment under scenario 1 is $y^1_{11} = y^1_{21} = y^1_{42} = 20$ and $w_3 = w_4 = 10$. This gives an objective value of -18. Giving this solution as a starting point for the local solver (DCA), we obtain a different upper level point, namely $x = (40, 30)$. This decision, together with the corresponding optimal lower level assignment gives an objective value of -18.5. Under this circumstance, demand from scenario 2 is always fulfilled by the upper level supply, but in scenario 1 some part of the demand, more precisely 10 units from node 3, is purchased from the competing supplier because the upper level agent does not have sufficient supply. Notice that the optimal solution of each scenario is degenerate, since the number of non-zero optimal variables is $4 < 5 = s + d - 1$. Also, these solutions are unique. These two properties combined imply that the dual of the inner level problem, for each scenario, has multiple solutions. Furthermore, this solution is a local optima in the SAA-LPCC problem.

The global optimal solutions of the SAA-LPCC, in the outer level variables, is the convex hull of the vectors $(50, 30)$ and $(40, 40)$. If either $x_1 = 40$ or $x_2 = 30$ is increased to capture more demand, then the overall cost would decrease, leading to a better solution.

Let us recall the optimality conditions of the inner problem

$$\sum_{j \in S} y_{ij}^{\xi} + w_i^{\xi} = \xi_i \qquad i \in D \tag{4.9a}$$

$$\lambda_i^{\xi} \geq 0 \qquad i \in D \tag{4.9b}$$

$$0 \leq x_j - \sum_{i \in D} y_{ij}^{\xi} \perp \mu_j^{\xi} \geq 0 \qquad j \in S \tag{4.9c}$$

$$0 \leq c_{ij} + r_j - \lambda_i^{\xi} + \mu_j^{\xi} \perp y_{ij}^{\xi} \geq 0 \qquad i \in D, j \in S \tag{4.9d}$$

$$0 \leq \rho_i - \lambda_i^{\xi} \perp w_i^{\xi} \geq 0 \qquad i \in D. \tag{4.9e}$$

In order to give incentives to increase, for example x_2, from 30 to 40, two things should happen so the new solution is still feasible: (1) the scenarios where x_2 will be overstocking should have the corresponding $\mu_2 = 0$, since the left-hand side of complementarity (4.9c) would be inactive, and (2) the left-hand side of (4.9d) should be set to zero, to allow y_{32} to increase.

A similar argument may be done if the desire is to reduce inventory x_j in some store j. In this case, the idea would be to increase some λ_i to activate the left-hand side of 4.9e so w_i may be released (reducing supply can be viewed as allowing part of the demand, on some scenarios, to be purchased from the competition).

The alternating weights technique consists on iteratively solving weighted modifications of (4.8) which encourages dual variables to shift up or down, resulting in non-strict complementaries that permit the DCA method to choose the corresponding piece where the LPCC can benefit the most. In the general NNP case, let $(x^0, y^0, w^0, \lambda^0, \mu^0)$ be the solution found by the local solver and $\mathcal{K}$ be the feasible region of (4.8). The alternating weights method creates a new set of iterates by sequentially solving

$$(x^{k+1}, y^{k+1}, w^{k+1}, \lambda^{k+1}, \mu^{k+1}) \in \underset{(x,y,w,\lambda,\mu)\in\mathcal{K}}{\arg\min} \sum_{j\in S} c_j x_j - \mathbb{E}_\xi \left[\sum_{\substack{i\in D \\ j\in S}} r_j y_{ij}^\xi \right] + f_k(\mu, \lambda), \tag{4.10}$$

where $f_k(\mu, \lambda)$ is a penalty term defined as

$$f_k(\mu, \lambda) = \begin{cases} \zeta \sum_{\substack{\xi\in\Omega^N \\ j\in S}} \mu_j^\xi & \text{if } k \text{ is odd} \\ -\zeta \sum_{\substack{\xi\in\Omega^N \\ i\in D}} \lambda_i^\xi & \text{if } k \text{ is even} \end{cases}, \tag{4.11}$$

with the penalty parameter ζ set to a value big enough to steer μ towards zero (or λ to ρ). In theory, any value for ζ would work, but in practice it needs to be larger than the LP solver tolerance. On the other hand, ζ must also be small enough so that it does not counteract the actual cost parameters of the problem, c and r. It is understood that the solution of problem (4.10) uses $(x^k, y^k, w^k, \lambda^k, \mu^k)$ as the starting point for the DCA and that the argmin refers to the local solution of the modified problem, provided by the DCA. The method stops when for some k, $(x^k, y^k) = (x^{k+1}, y^{k+1})$.

In the toy example, when setting $\zeta = 10^{-5}$, the dual variable of the second store on scenario 1, μ_2^1, is pushed to zero, allowing x_2, in complementarity (4.9c), to increase up to 40. The variables λ^1 for demand nodes 3 and 4 adjust themselves to maintain complementarity of (4.9d). This change in λ is not surprising. Recall that the dual objective function of the inner level problem is $\sum_{i\in D} \xi_i \lambda_i - \sum_{j\in S} x_j \mu_j$, hence, in order to keep the objective value constant, because we are looking for the "correct" dual optimal solution, a decrease in μ_j implies a decrease in some λ's (provided that $x_j \neq 0$).

In the second iteration, when penalizing λ in f_k in (4.10), the dual variables change again, but the primal remains the same so the method stops. We end up in the globally optimal solution $x = (40, 40)$.

4.3.2. Numerical Results

4.3.2.1. Case d=30, s=4. We now present the numerical experiments of this chapter. We start with the instance where $d = 30$ and we set the parameter α to 0.1. We globally solved SS-LPCCs with subsample size $n = 5, 10, 15, 20, 25, 30$ and 40. We will denote the optimal solution and value of these problems as x_g^n and z_g^n. In a similar manner, we will denote by x_n^N and z_n^N the local solution and objective value, respectively, found by the DCA with the alternating weights technique described in Section 4.3.1, in the SAA-LPCC with N scenarios, starting from x_g^n. The selected sizes for N are 100, 200, 300 and 500. Finally, (and) since we are interested in the "real" objective value for the (continuous) stochastic bilevel program, we compute a proxy for this objective value by drawing 40,000 samples, solving the inner level problem on each scenario and computing the sample mean (essentially evaluating the objective value on an SAA-LPCC with $N = 40,000$). In our experiments 40,000 samples was large enough so that the obtained objective value had a very small variance as computed over 10 runs. We will denote by $\bar{z}^n$ and $\bar{z}_n^N$ the proxy for the stochastic objective value of x_g^n and x_n^N, respectively. We also compute a "nominal" value obtained by solving the single scenario deterministic LBLP where the demand for node i is given by the mean demand, θ_i. This value is labeled z_{nom}.

Notice that since we are solving for the optimistic formulation, to evaluate the true stochastic objective value the inner level problem for each scenario is solved twice. First to obtain an optimal value z^ξ for scenario ξ, and then to choose the solution which benefits the leader the most. That is, adding a restriction which forces the inner objective value to be less or equal to z^ξ, and replacing the inner objective function, with the outer objective which depends on y (i.e. $\sum_{\substack{i\in D\\ j\in S}} r_j y_{ij}$).

Table 4.1 shows the average and standard deviation, across 20 runs, of three metrics related to the computation of the local solutions x_n^N for all combinations of n and N: (1) Obj, which represents the percentage improvement of $\bar{z}_n^N$ over z_{nom}. It is computed as $\left|\frac{\bar{z}_n^N - z_{nom}}{z_{nom}}\right|$. (2) Iters, which represents the number of "major" iterations of the alternating weights method. This means, the number of alternations between adding weights to μ and λ, and (3) total CPU time taken from the subsampled solution to the SAA-LPCC solution. That is, the time taken for the MILP solver is not considered in this table. We observe that CPU times and Iters increase with N. This is expected, since the dimension of the complementarities is $O(dsN)$ and, hence, grows linearly with the number of scenarios. Also, the larger the subsample size n, the better the prediction for an optimal outer level decision, which makes the DCA require less effort to find a good local solution, and explains why Iters and CPU time decrease as n increases. For the same reason, the percentage improvement gets better as n increases. A final interesting observation is that, although solved only locally, the percentage improvement also increases with N. That means the DCA makes efficient use of the larger samples to reach better candidate solutions to the stochastic problem.

Table 4.1. Average and standard deviation (in parenthesis) of major iterations, CPU time and percentage improvement of z_n^Nover nominal value

n	Avg (Std Dev)	N			
		100	200	300	500
5	Obj (%)	3.03 (0.22)	3.05 (0.22)	3.06 (0.22)	3.06 (0.22)
	Iters	9.40 (4.51)	11.20 (5.28)	14.10 (6.77)	16.80 (7.12)
	CPU time (s)	15.00 (7.34)	64.25 (27.55)	160.81 (53.53)	597.21 (246.83)
10	Obj (%)	3.09 (0.21)	3.12 (0.20)	3.13 (0.20)	3.13 (0.20)
	Iters	9.30 (4.36)	12.10 (5.10)	14.25 (4.68)	17.70 (5.89)
	CPU time (s)	13.90 (5.14)	63.10 (16.25)	157.32 (47.89)	564.71 (178.18)
15	Obj (%)	3.15 (0.18)	3.18 (0.18)	3.19 (0.18)	3.19 (0.18)
	Iters	9.40 (3.25)	12.30 (4.51)	14.15 (4.62)	17.55 (5.82)
	CPU time (s)	12.71 (4.37)	58.55 (18.45)	146.85 (42.44)	525.15 (186.64)
20	Obj (%)	3.20 (0.13)	3.22 (0.13)	3.23 (0.13)	3.24 (0.13)
	Iters	8.05 (3.02)	12.15 (4.82)	14.45 (5.24)	17.35 (5.01)
	CPU time (s)	11.37 (4.47)	54.08 (17.91)	135.81 (39.72)	479.35 (137.29)
25	Obj (%)	3.22 (0.10)	3.25 (0.09)	3.26 (0.09)	3.26 (0.09)
	Iters	6.65 (2.57)	10.80 (3.43)	12.90 (4.56)	14.85 (5.14)
	CPU time (s)	9.49 (4.20)	47.73 (14.87)	117.52 (44.57)	405.98 (151.35)
30	Obj (%)	3.20 (0.13)	3.23 (0.13)	3.23 (0.13)	3.24 (0.13)
	Iters	6.15 (2.48)	9.40 (3.57)	11.30 (4.57)	13.85 (5.62)
	CPU time (s)	8.48 (4.32)	40.27 (19.29)	99.81 (45.76)	353.49 (165.34)
40	Obj (%)	3.24 (0.04)	3.27 (0.02)	3.28 (0.02)	3.28 (0.01)
	Iters	5.95 (3.06)	10.60 (4.04)	11.10 (4.97)	14.50 (5.17)
	CPU time (s)	6.56 (4.60)	40.45 (19.69)	96.82 (47.88)	355.62 (169.47)

Table 4.1 suggests that in order to achieve a larger improvement to nominal percentage gap, we should be globally solving for larger subsamples (increases in this metric are much more significant by moving vertically in the table, rather than horizontally), but increasing n also results in increasing CPU time for the subsampled LPCC. Table 4.2 shows the mean, median, minimum, maximum and standard deviation for CPU time taken to solve the subsampled LPCC for different number of scenarios n. To speed up this MILP solver, we first found the global solution for $n = 5$ and then provided that solution as an incumbent for the remaining scenario sizes. We see that the time to solve up to 40 subsamples (which

leads to $\sim 3.27\%$ improvement) could take almost up to two hours, while solving for half the number of subsamples (20), it would take less than a tenth of that time (and would lead to a respectable 3.23% improvement).

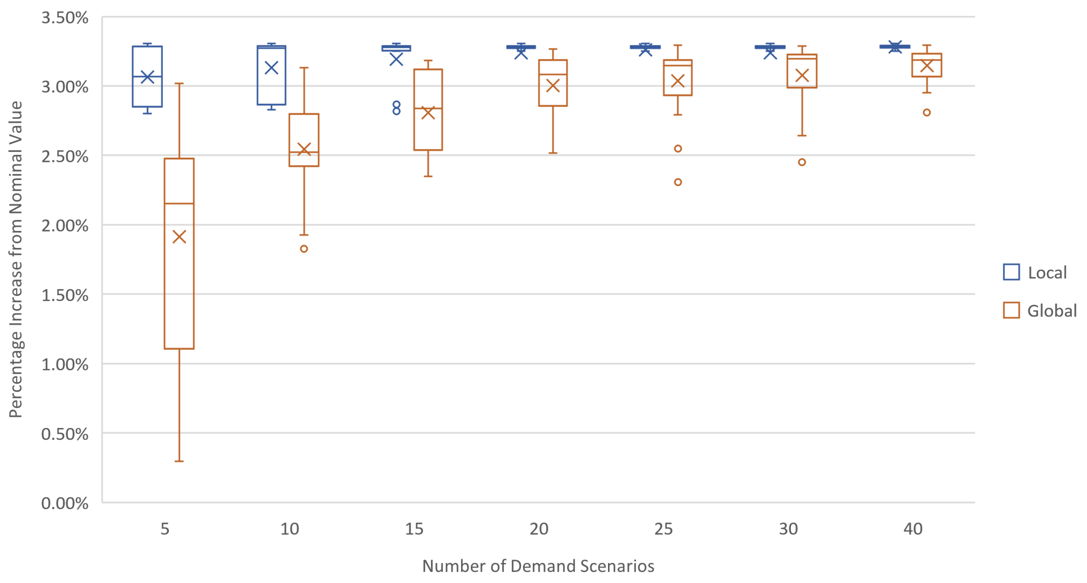
Table 4.2. Mean, median and standard deviation of CPU time to find x_g^n

CPU Time (s)	5	10	15	20	25	30	40
Mean	2.92	19.59	74.62	173.90	426.33	666.51	1930.50
Median	2.69	19.76	63.60	151.95	310.16	469.92	1350.53
Std Dev	2.36	10.44	44.99	103.82	375.53	598.78	1681.32
Min	0.36	1.23	8.52	56.10	74.41	143.05	186.97
Max	11.65	38.52	188.17	401.29	1615.09	2580.44	6303.59

We also want to assess how the subsampled solution x_g^n compares to our SAA solution x_n^N in the stochastic LPCC. Figure 4.4 superimposes two box plots, representing $\bar{z}^n$ (orange box) and $\bar{z}_n^{500}$ (blue box). We can see that starting from a subsample of size 20, the value of $\bar{z}_n^{500}$ already settles. This could be an indication that we actually reached the global optimum, or that starting from a subsample of size 20, we always reach a same local solution. To get a sense out of this last statement, we run the MILP solver to globally solve an LPCC with a sample of 500 scenarios. Since we were only interested in the objective value, we let it run for 12 hours on 4 threads, providing x_{20}^{500} and z_{20}^{500} as an incumbent. We found out that whenever the MILP did not find a global solution within this time limit, the percentage improvement over nominal for the best lower bound found in the process, it barely represented a 0.001% difference with the ones shown in the plot. Therefore, we did actually find a (close to) global solution.

4.3.2.2. Case d=50, s=4. We now focus on the larger instance, with 50 demand nodes and 4 stores. As we said earlier, the number of complementarities is of the order of dsN, hence we are aiming to find close to global solutions of LPCCs up to a considerably large

Figure 4.4. Global vs Local solutions for different subsamples ($d = 30$)



scale. To get a sense of this, when solving for $N = 500$, this means around $100,000$ complementarities.

For this experiment we set α to 0.25, resulting in a significant increase of the variance of the random demand. We report similar tables as in the $d = 30$ case, starting with the MILP solving times. Again, we first solved globally for a subsample $n = 5$ and provided the solution as an incumbent for the larger subsamples. In this occasion, we ran the MILP solver with 4 threads over a time period of 12 hours (an equivalent 2 days if single-threaded). Table 4.3 shows the same statistics as before, but only among the runs, out of 20, which managed to find (or, more precisely, certify) a global optimum. We add a last line to the table with the information about how many runs, per subsample, did not run out of time. We observe that for subsample size 30 and larger very few of the runs actually found (with a certificate) a global solution. In these situations, we still continue the experiment with the incumbent found when the time limit was reached.

Table 4.3. Statistics for CPU time to find x_g^n

CPU Time	5	10	15	20	25	30	40
Mean	75.05	252.90	1,593.80	8,677.84	22,037.93	22,633.69	36,656.62
Std Dev	55.88	460.29	2,364.44	6,988.59	13,236.34	5,751.74	-
Min	17.04	9.72	0,064.28	1,197.11	2,252.78	14,194.45	36,656.62
Max	265.07	1,868.84	9,240.53	24,838.74	38,554.15	30,059.78	36,656.62
Solved	20	20	20	18	14	6	1

As in the previous instance, Table 4.4 shows that the metrics Iters and CPU time increase with N and decrease with n. This is still expected, regardless of whether the subsample was solved to global optimality or not. The metric that indeed gets affected by not globally solving the subsampled problem within the time limit is the percentage improvement from the nominal value (Obj). What we observe is that this metric increases up to $n = 20$ and then decreases drastically. It even drops below the percentage improvement obtained from x_5^N. This is an interesting observation considering that x_g^5 was the initial incumbent to solve larger subsample sizes. This could be due to the larger subsample not being solved to global optimality. We also notice that there is barely any improvement from z_n^{500} to z_n^{800}, for any n, which basically implies that a stable local solution was found, in the sense that more samples do not allow it to escape.

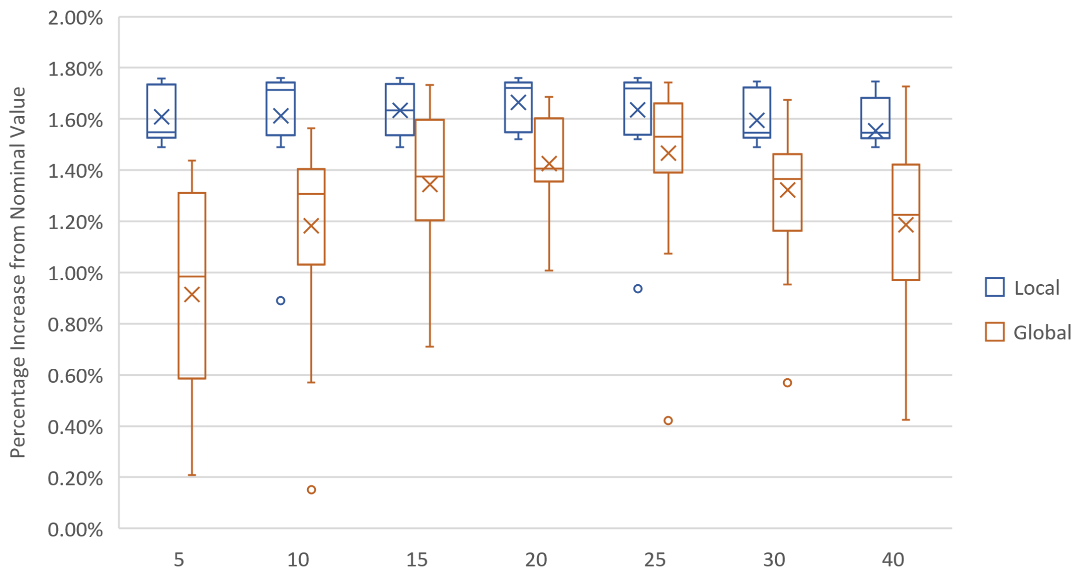
Now we move to the box plot comparisons, $\bar{z}^n$ versus $\bar{z}_n^{500}$ (as compared to the nominal value), illustrated in Figure 4.5. Here we see that the local solutions found, opposed to the $d = 30$ case, show a wide and similar spread regardless of the subsample size. To verify if this was due to $N = 500$ being still too small, we also computed the local solutions for $N = 800$, and saw that the results were practically the same as in $N = 500$. This can

Table 4.4. Average and Standard Deviation of major iterations, CPU time and percentage improvement of $\bar{z}_n^N$ over nominal value

n	Avg (Std Dev)	N			
		200	300	500	800
5	Obj (%)	1.57 (0.11)	1.59 (0.11)	1.60 (0.10)	1.61 (0.10)
	Iters	17.55 (11.02)	21.10 (13.01)	21.85 (11.84)	28.26 (15.03)
	CPU time (s)	221.02 (123.01)	567.47 (316.48)	1,347.21 (952.84)	3,915.09 (1,615.50)
10	Obj (%)	1.60 (0.19)	1.60 (0.19)	1.61 (0.19)	1.61 (0.19)
	Iters	17.20 (9.90)	19.95 (11.10)	22.15 (10.27)	27.05 (11.39)
	CPU time (s)	157.54 (56.84)	377.22 (135.50)	954.77 (418.65)	3,218.03 (1,267.77)
15	Obj (%)	1.61 (0.11)	1.62 (0.11)	1.63 (0.10)	1.63 (0.10)
	Iters	13.15 (4.49)	16.15 (8.45)	20.75 (8.61)	25.60 (9.36)
	CPU time (s)	108.32 (55.36)	273.40 (147.50)	678.83 (431.53)	2,490.90 (1,236.79)
20	Obj (%)	1.63 (0.11)	1.65 (0.10)	1.66 (0.09)	1.66 (0.09)
	Iters	16.15 (8.76)	19.95 (9.27)	22.35 (8.18)	25.45 (11.72)
	CPU time (s)	126.24 (64.51)	334.56 (183.62)	884.33 (518.60)	2,821.07 (1,459.08)
25	Obj (%)	1.62 (0.18)	1.63 (0.18)	1.63 (0.18)	1.63 (0.18)
	Iters	15.80 (7.99)	18.25 (10.50)	19.20 (7.33)	20.65 (8.42)
	CPU time (s)	91.43 (46.62)	220.88 (114.95)	561.87 (283.65)	1,847.44 (837.18)
30	Obj (%)	1.57 (0.10)	1.58 (0.11)	1.59 (0.09)	1.59 (0.09)
	Iters	15.40 (9.73)	16.50 (8.45)	18.60 (8.77)	23.25 (12.20)
	CPU time (s)	99.70 (48.99)	225.94 (112.85)	528.87 (306.21)	1,825.84 (827.60)
40	Obj (%)	1.55 (0.09)	1.57 (0.09)	1.55 (0.17)	1.55 (0.17)
	Iters	19.60 (14.45)	20.50 (12.97)	21.70 (10.67)	29.15 (15.84)
	CPU time (s)	93.73 (106.16)	197.02 (205.28)	525.71 (392.24)	1,467.14 (1,192.52)

be confirmed from Table 4.3 as well. We come to the conclusion that depending on the subsample, the method ended in different local solutions, and therefore a global solution is not reached. This, again, occurs as opposed to Table 4.4 where the local solutions stabilized very close to the global optimum. Another interesting observation is that even for subsamples that were not solved to global optimality, the local method managed to bring the percentage improvement close to or even surpassing the 1.6% threshold, in average. Similar to $d = 30$, results indicate that a safe approach would be to solve globally for 10-15 subsamples and then local for 500, in order to balance total CPU time and percentage improvement.

Figure 4.5. Global vs Local solutions for different subsamples ($d = 50$)



We finish this section by mentioning that we tried the alternating weights technique starting with λ, instead of μ, but saw no significant differences and therefore those findings are not reported.

4.3.3. Conclusions and Future Research

We presented an efficient method to find good approximations of global solutions for the bilevel network newsvendor problem, which takes advantage of primal degeneracy in the lower level optimal solutions. The method relied on a good guess for the outer level decision variable provided by a subsampled LPCC which was solved to global optimality. We used the DCA approach on the presented alternating weights technique to drive the given starting point to improved local optima in a larger sample. With this approach we were able to find better solutions of the actual stochastic problem in considerably less time than its global solver counterparts.

As part of a future research, we intend to incorporate the location of the stores as an upper level decision. We describe the bilevel capacitated facility location problem below, and outline initial findings and observations.

4.3.3.1. From Network Newsvendor to Facility Location. A natural extension we would like to explore is the bilevel formulation of the Capacitated Facility Location Problem (CFLP), where now the upper level agent is also involved in the decision of which stores to open, given a set of potential locations, under a fixed cost. Once the stores are open, she needs to decide how much inventory to put on each store in order to minimize total costs generated from the fixed and unit costs, considering that revenue comes from a lower level transportation problem, just like in the Network Newsvendor problem.

The essential difference between NNP and CFLP is the existence of a binary variable $z \in \{0,1\}^s$ which represents whether a store should be open on any location $j \in S$. The objective function of the CFLP is the same as in 4.7 with an extra term $\sum_{j \in S} q_j z_j$, where q_j is the fixed cost incurred if location j is chosen to open a store. In terms of the constraints, all constraints of 4.7 remain the same, but we need to add a constraint to represent that no inventory should be put in a non-selected location, that is, $x_j \leq M z_j$, where M is a sufficiently large number, similar to M_Ω described before.

Adding the binary variables represents an extra challenge, since the method presented in this chapter only considers continuous variables (in fact, the DCA approach requires that the feasible set of each subproblem is convex). Therefore, we have the need to reformulate the problem accordingly. One straight-forward way to do this is to add, for each $j \in S$, the following complementarity constraint

$$0 \leq z_j \perp 1 - z_j \geq 0, \tag{4.12}$$

but this would not fit into our setting, since it does not allow for non-strict complementarities. In particular, the piece-wise linear penalty function of the DCA approach would be incapable of switching values for z, without becoming infeasible. We can instead treat the cost function for each location $j \in S$ as a complementarity. Notice that there is a clear relationship between x_j and z_j, namely $x_j > 0 \iff z_j = 1$. Therefore, the cost $f_j(x_j)$ is given by

$$f_j(x_j) = \begin{cases} 0 & \text{if } x_j = 0 \\ q_j + c_j x_j & \text{if } x_j > 0. \end{cases} \tag{4.13}$$

If we plot this function we obtain the graph on the left of Figure 4.6. Alternatively, we can represent this relationship as "either $x_j = 0$ or $q_j + c_j x_j - f_j = 0$" which is depicted on the right graph of Figure 4.6 (considering also the non-negativity of x_j and that $f_j \leq q_j + c_j x_j$). The final reformulation for the full sampled LPCC is then

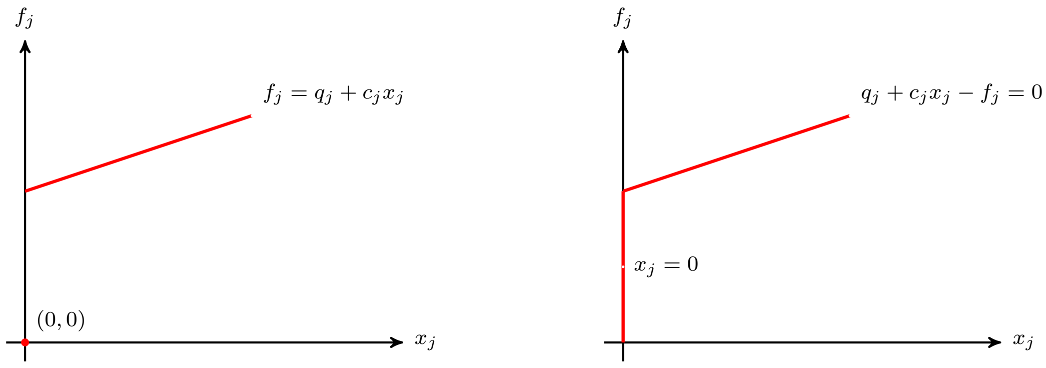


Figure 4.6. Cost representation: Left, f_j as a function of x_j; Right, as a complementarity

$$
\begin{aligned}
\min_{x,y,w,f} \quad & \sum_{j\in S} f_j - \mathbb{E}_\xi \left[\sum_{\substack{i\in D\\ j\in S}} r_j y_{ij}^\xi \right] \\
\text{s.t.} \quad & 0 \le q_j + c_j x_j - f_j \perp x_j \ge 0 \quad j \in S
\end{aligned}
$$

$$
\tag{4.14}
\left.
\begin{array}{llll}
0 = \sum_{j\in S} y_{ij}^\xi - \xi_i + w_i^\xi, & \lambda_i^\xi \ge 0 & & i \in D \\
0 \le x_j - \sum_{i\in D} y_{ij}^\xi & \perp & \mu_j^\xi \ge 0 & j \in S \\
0 \le c_{ij} + r_j - \lambda_i^\xi + \mu_j^\xi & \perp & y_{ij}^\xi \ge 0 & i \in D, j \in S \\
0 \le \rho_i - \lambda_i^\xi & \perp & w_i^\xi \ge 0 & i \in D
\end{array}
\right\} \xi \in \Omega.
$$

Some observations are worth mentioning for this formulation. Notice that if a feasible point of (4.14) satisfies $x_j = 0$ for some $j \in S$, then the objective function guarantees that $f_j = 0$ as well. Hence, this complementarity representation of $f_j(x_j)$ coincides with (4.13) and, therefore, the set of local minima in both formulations is the same.

The main concern with regards to the bilevel CFLP is whether the alternating weights technique is able to capture, for instance, the need of opening a closed facility or vice-versa.

4.3.3.2. Degeneracy on Inner Level Problems. The alternating weights technique works nicely in the NNP since the lower level problem presented multiple dual solutions (primal degeneracy) in many of its scenarios, a common situation when dealing with network problems. This leads us to believe that in general bilevel problems with inner level network problems, the method should perform well. As an extended research project we would like to test this method on further bilevel formulations for applications related to transportation or energy systems, where network models fit naturally.

References

Bai, Lijie, John E Mitchell, Jong-Shi Pang. 2013. On convex quadratic programs with linear complementarity constraints. *Computational Optimization and Applications* **54**(3) 517–554.

Bard, Jonathan F, James T Moore. 1990. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing* **11**(2) 281–292.

Birbil, Ş İlker, Gül Gürkan, Ovidiu Listeş. 2006. Solving stochastic mathematical programs with complementarity constraints using simulation. *Mathematics of Operations Research* **31**(4) 739–760.

Burdakov, Oleg P, Christian Kanzow, Alexandra Schwartz. 2016. Mathematical programs with cardinality constraints: Reformulation by complementarity-type conditions and a regularization method. *SIAM Journal on Optimization* **26**(1) 397–425.

Byrd, RH, J Nocedal, RA Waltz. 2006. An integrated package for nonlinear optimization. *Large-scale nonlinear optimization* 35–59.

Candes, Emmanuel J, Michael B Wakin, Stephen P Boyd. 2008. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications* **14**(5-6) 877–905.

Cottle, RW, JS Pang, RE Stone. 1992. The linear complementarity problem. 1992. *AP, New York* .

Dempe, Stephan. 2003. *Bilevel programming: A survey*. Dekan der Fak. für Mathematik und Informatik.

Dempe, Stephan, Joydeep Dutta. 2012. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Mathematical programming* **131**(1-2) 37–48.

Dinh, Tao Pham, Hoai An Le Thi. 2014. Recent advances in DC programming and DCA. *Transactions on computational intelligence XIII*. Springer, 1–37.

Evgrafov, Anton, Michael Patriksson. 2004. On the existence of solutions to stochastic mathematical programs with equilibrium constraints. *Journal of Optimization Theory and Applications* **121**(1) 65–76.

Fang, Haw-ren, Sven Leyffer, Todd Munson. 2012. A pivoting algorithm for linear programming with linear complementarity constraints. *Optimization Methods and Software* **27**(1) 89–114.

Feng, Mingbin, John E Mitchell, Jong-Shi Pang, Xin Shen, Andreas Wächter. 2013. Complementarity formulations of ℓ_0-norm optimization problems. *Industrial Engineering and Management Sciences. Technical Report. Northwestern University, Evanston, IL, USA* .

Fletcher, Roger, Sven Leyffer. 2002. Nonlinear programming without a penalty function. *Mathematical programming* **91**(2) 239–269.

Fletcher*, Roger, Sven Leyffer. 2004. Solving mathematical programs with complementarity constraints as nonlinear programs. *Optimization Methods and Software* **19**(1) 15–40.

Fletcher, Roger, Sven Leyffer, Danny Ralph, Stefan Scholtes. 2006. Local convergence of SQP methods for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization* **17**(1) 259–286.

Fletcher, Roger, Sven Leyffer, Philippe L Toint. 2002. On the global convergence of a filter–SQP algorithm. *SIAM Journal on Optimization* **13**(1) 44–59.

Fourer, Robert, David M Gay, Brian W Kernighan. 1990. A modeling language for mathematical programming. *Management Science* **36**(5) 519–554.

Hooker, John N, Greger Ottosson. 2003. Logic-based Benders decomposition. *Mathematical Programming* **96**(1) 33–60.

Hu, Jing, John E Mitchell, Jong-Shi Pang. 2012a. An LPCC approach to nonconvex quadratic programs. *Mathematical programming* **133**(1-2) 243–277.

Hu, Jing, John E Mitchell, Jong-Shi Pang, Kristin P Bennett, Gautam Kunapuli. 2008. On the global solution of linear programs with linear complementarity constraints. *SIAM Journal on Optimization* **19**(1) 445–471.

Hu, Jing, John E Mitchell, Jong-Shi Pang, Bin Yu. 2012b. On linear programs with linear complementarity constraints. *Journal of Global Optimization* **53**(1) 29–51.

Ibaraki, Toshihide. 1971. Complementary programming. *Operations Research* **19**(6) 1523–1529.

Ibaraki, Toshihide. 1973. The use of cuts in complementary programming. *Operations Research* **21**(1) 353–359.

Jara-Moroni, Francisco, Jong-Shi Pang, Andreas Wächter. 2016. A study of the difference-of-convex approach for solving linear programs with complementarity constraints. *Mathematical Programming* 1–34.

Jeroslow, Robert G. 1978. Cutting-planes for complementarity constraints. *SIAM Journal on Control and Optimization* **16**(1) 56–62.

Júdice, Joaquim J. 2012. Algorithms for linear programming with linear complementarity constraints. *Top* **20**(1) 4–25.

Karzan, Fatma Kılınç, George L Nemhauser, Martin WP Savelsbergh. 2009. Information-based branching schemes for binary linear mixed integer problems. *Mathematical Programming Computation* **1**(4) 249–293.

Le Thi, Hoai An, Tao Pham Dinh. 2011. On solving linear complementarity problems by DC programming and DCA. *Computational optimization and applications* **50**(3) 507–524.

Le Thi, Hoai An, Tao Pham Dinh. 2013. The state of the art in DC programming and DCA. *Research Report (60 pages), Lorraine University* .

Le Thi, Hoai An, Tao Pham Dinh, et al. 2014. DC programming and DCA for general DC programs. *Advanced Computational Methods for Knowledge Engineering*. Springer, 15–35.

Leyffer, Sven. 2000. MacMPEC: AMPL collection of MPECs.

Leyffer, Sven, Gabriel López-Calva, Jorge Nocedal. 2006. Interior methods for mathematical programs with complementarity constraints. *SIAM Journal on Optimization* **17**(1) 52–77.

Leyffer, Sven, Todd S Munson. 2007. A globally convergent filter method for MPECs. *Preprint ANL/MCS-P1457-0907, Argonne National Laboratory, Mathematics and Computer Science Division* .

Lin, Gui-Hua, Xiaojun Chen, Masao Fukushima. 2009. Solving stochastic mathematical programs with equilibrium constraints via approximation and smoothing implicit programming with penalization. *Mathematical Programming* **116**(1-2) 343–368.

Lin, Gui-Hua, Masao Fukushima. 2005. Regularization method for stochastic mathematical programs with complementarity constraints. *ESAIM: Control, Optimisation and Calculus of Variations* **11**(2) 252–265.

Lin, Gui-Hua, Masao Fukushima. 2010. Stochastic equilibrium problems and stochastic mathematical programs with equilibrium constraints: A survey. *Pacific Journal of Optimization* **6**(3) 455–482.

Luo, Zhi-Quan, Jong-Shi Pang, Daniel Ralph. 1996. *Mathematical programs with equilibrium constraints*. Cambridge University Press.

Mangasarian, Olvi L. 1997. Solution of general linear complementarity problems via nondifferentiable concave minimization. *Acta Mathematica Vietnamica* **22**(1) 199–205.

Mitchell, John E, Jong-Shi Pang, Bin Yu. 2012. Obtaining tighter relaxations of mathematical programs with complementarity constraints. *Modeling and Optimization: Theory and Applications*. Springer, 1–23.

Muu, Le Dung, Tran Dinh Quoc, Le Thi Hoai An, Pham Dinh Tao. 2011. Decomposition algorithms for globally solving mathematical programs with affine equilibrium constraints. *arXiv preprint arXiv:1105.3343* .

Pang, Jong-Shi, Meisam Razaviyayn, Alberth Alvarado. 2016. Computing B-stationary points of nonsmooth DC programs. *Mathematics of Operations Research* **42**(1) 95–118.

Patriksson, Michael, Laura Wynter. 1999. Stochastic mathematical programs with equilibrium constraints. *Operations research letters* **25**(4) 159–167.

Rockafellar, R Tyrrell. 1997. Convex Analysis. Princeton landmarks in mathematics.

Scheel, Holger, Stefan Scholtes. 2000. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research* **25**(1) 1–22.

Shapiro, Alexander. 2006. Stochastic programming with equilibrium constraints. *Journal of Optimization Theory and Applications* **128**(1) 221–243.

Shapiro, Alexander, Huifu Xu. 2008. Stochastic mathematical programs with equilibrium constraints, modeling and sample average approximation. *Optimization* **57**(3) 395–418.

Tao, Pham Dinh, Le Thi Hoai An. 1997. Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica* **22**(1) 289–355.

Tao, Pham Dinh, et al. 2005. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of operations research* **133**(1-4) 23–46.

Xu, Huifu. 2006. An implicit programming approach for a class of stochastic mathematical programs with complementarity constraints. *SIAM Journal on Optimization* **16**(3) 670–696.

Xu, Huifu, J Ye Jane. 2011. Approximating stationary points of stochastic mathematical programs with equilibrium constraints via sample averaging. *Set-Valued and Variational Analysis* **19**(2) 283–309.

Yu, Bin. 2011. A branch and cut approach to linear programs with linear complementarity constraints. Ph.D. thesis, Rensselaer Polytechnic Institute.

Yu, Bin, John E Mitchell, Jong-Shi Pang. 2018. Solving linear programs with complementarity constraints using branch-and-cut. *arXiv preprint arXiv:1802.02941* .