

**INTEGER LINEAR PROGRAMMING METHODS TO SOLVE SOME
MIXED-INTEGER NONLINEAR PROGRAMS**

by

Mahdi Hamzeei

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy
(Industrial and Systems Engineering)

at the
UNIVERSITY OF WISCONSIN–MADISON
2014

Date of final oral examination: 12/05/2014

The dissertation is approved by the following members of the Final Oral Committee:

James Luedtke, Associate Professor, Industrial and Systems Engineering

Vicki Bier, Professor, Industrial and Systems Engineering

Jeffrey Linderoth, Professor, Industrial and Systems Engineering

Christos Maravelios, Professor, Chemical and Biological Engineering

Laura McLay, Associate Professor, Industrial and Systems Engineering

To my daughter, Nika,

To my wife, Narges,

And to my parents

ACKNOWLEDGMENTS

First and foremost, I must express my deepest and sincerest thanks to my advisor, James Luedtke, who supported me throughout my thesis. I believe this thesis would not have been written without his knowledge and expertise. I am also very grateful to Jeff Lindereth, the first director of the MINLP group in the department, for all his help and support, and also for serving on my dissertation committee. Thanks also to the remaining members of the committee, Prof. Vicki Bier, Prof. Christos Maravelias and Prof. Laura McLay. I would also like to thank the MINOTAUR group, especially my friend Ashutosh Mahajan with his invaluable help. I had amazing time with the LOL (Luedtke Optimization Lab) group (in chronological order): Yongjia Song, Merve Bodur, Thai Dinh and Eli Towle. Thanks for everything, for discussing assignments and class projects, for helping me figure out coding tricks, for giving me feedback on my presentation, and especially tolerating me as their office mate.

For all the love and support, last but not least, I would like to deeply and wholeheartedly thank my wife, Narges, who has been the best and greatest source of inspiration and motivation for me.

DISCARD THIS PAGE

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT	vii
1 Introduction	1
1.1 Thesis overview	1
1.2 Background	4
1.2.1 Relaxation and valid inequality	5
1.2.2 Branch-and-bound	6
1.2.3 Concepts in theory of nonlinear programming	8
1.2.4 Quasiconvex functions and their properties	11
1.2.5 Lagrangian relaxation in integer programming	12
1.2.6 Integer L-shaped method for two-stage stochastic integer programs	13
1.2.7 Bilevel programming	15
2 Linearization-Based Algorithms for Mixed-Integer Nonlinear Programs with Convex Continuous Relaxation	20
2.1 Introduction	20
2.2 Algorithms for Convex MINLP Problems	25
2.3 Proposed Algorithms	33
2.3.1 Proj-Grad: The Projection-Based LP/NLP-BB Algorithm	34
2.3.2 Hybrid-BB: A Hybrid Between LP/NLP and NLP-based Branch- and-Bound	39
2.4 Computational Results	43
2.4.1 Implementation details	43
2.4.2 Test instances	44

	Page
2.4.3 Results	47
2.5 Conclusion	52
3 A Lagrangian relaxation approach to a class of mixed-integer bilevel programming problems	68
3.1 Introduction	68
3.2 Problem formulation	70
3.3 MINLP reformulation	71
3.4 Lagrangian relaxation for fixed capacity	73
3.5 Capacity size as a decision variable	80
3.6 Computational Results	86
3.6.1 Test instances and settings	86
3.6.2 Results for fixed capacity size	88
3.6.3 Results for capacity size as a decision variable	94
3.7 Conclusion	96
4 A class of optimization problems with monotone nonconvex cost function	98
4.1 Introduction	98
4.2 Related stochastic integer programming literature	100
4.3 Base methods	101
4.4 Method	103
4.5 Numerical Results	108
4.5.1 Applications	109
4.5.2 Results	111
4.6 Conclusion	116
LIST OF REFERENCES	118

DISCARD THIS PAGE

LIST OF TABLES

Table	Page
2.1 Instance classes	46
2.2 Results of instances failed by all algorithms	50
2.3 Summary results by instance class	54
2.4 Summary results for instances not solved by all algorithms	55
2.5 Detailed numerical results of all the instances	56
3.1 Summary results for sensitivity of model to κ	87
3.2 Summary results of instances with fixed capacity and equal weights case . .	90
3.3 Summary results to show the difference between centralized and decentral- ized solutions	92
3.4 Summary results of instances with fixed capacity and unequal weights case .	93
3.5 Summary results of instances with variable capacity size and equal weights case	95
4.1 Summary results on processing root node of SGAPs	113
4.2 Summary results of the methods in the root node	114
4.3 Summary results of branch-and-bound for SGA	115
4.4 Summary results of branch-and-bound for SCPP	116

DISCARD THIS PAGE

LIST OF FIGURES

Figure	Page
2.1 Convexity of the continuous relaxation in a quasiconvex MINLP.	22
2.2 Scatter plot of CPU time in seconds for Proj-Obj vs. Proj-Grad.	48
2.3 Scatter plot of CPU time in seconds of Proj-Grad and LP/NLP-BB on convex instances.	49
2.4 Performance profile of CPU time in seconds of Proj-Grad, Hybrid-BB and NLP-BB.	51

ABSTRACT

Mixed-Integer Nonlinear Programs (MINLPs) are used to model many applied problems. However, finding a global optimal solution to such problems is difficult due to the combined presence of integer variables and nonlinear functions. If a MINLP has special structure, one can exploit the structure and develop a specialized approach that has potential to solve the problem more efficiently than a general-purpose solver. In this thesis, we study MINLPs with three different structures.

We first consider a class of MINLPs in which the set defined by the nonlinear constraint functions is convex, but the nonlinear functions may not be convex. Existing solution approaches for convex MINLPs are not guaranteed to find the global optimal solution of such problems. We propose two linearization-based methods to solve this class of problems. We prove the method finds an optimal solution using a finite number of nodes and cuts. Computational results show our methods are as efficient as the existing methods while solving this more general problem class.

Second, we study a network design problem in which an operator wants to open facilities and determine their size to satisfy demand of customers. Customers face congestion at the facilities they send their demand to, which depends on total usage of facility. We model this problem as a Mixed-Integer Bilevel Program (MIBP) which can be reformulated as a nonconvex MINLP, but is difficult to solve by general purpose solvers. Hence, we propose a Lagrangian relaxation approach which finds a candidate feasible solution along with a

lower bound that can be used to validate the solution quality. We find that the method can efficiently find provably high quality solutions even for large instances.

Finally, we study optimization problems having a nonconvex monotone objective function that is computationally expensive to evaluate. Such problems can be solved using cuts known as “integer L-shaped cuts.” We exploit the monotonicity structure to derive a stronger version of these cuts, and explore the use of mixing inequalities to further tighten the relaxation. Computational studies show that the strengthened cuts yield improved computational performance, but the impact of mixing inequalities is marginal.

Chapter 1

Introduction

In this dissertation, we study several optimization problems that have *integer* or *discrete* variables with nonlinear constraints, a problem class known as Mixed-Integer Nonlinear Programming (MINLP). We formulate such problems as follows:

$$\begin{aligned} z^* = \min \quad & c^T x \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{MINLP}$$

where $c \in \mathbb{R}^n$ and $X = \{x \in \mathbb{R}^{n-p} \times \mathbb{Z}^p \mid g_j(x) \leq 0, j \in J\}$ for some continuous functions $g_j : \mathbb{R}^n \rightarrow \mathbb{R}$. We often write x_I as the vector of variables constrained to be integer. The problem (MINLP) may also have a nonlinear convex objective function which can be equivalently reformulated to a problem with linear objective function. The problem defined in (MINLP) can be used to model a broad class of applications. If all functions $g_j, j \in J$ are linear, the problem (MINLP) reduces to a Mixed Integer Linear Program (MILP) and if $p = 0$, the problem becomes a Nonlinear Program (NLP). For an extensive overview of applications of MINLPs, see [15].

1.1 Thesis overview

MINLPs can be used to model a broad class of applied problems. They can model both discrete decisions and also nonlinear relationships between decision variables. However, this class of problems can be computationally very difficult to solve. There are some existing general-purpose solvers which can be used to find solutions to the MINLPs, but

the ability to solve instances of the size required in practice is often limited. In many problem classes, one could study and exploit special structures appearing in that problem, and derive more efficient approaches than the general-purpose solvers. In this thesis, we follow this approach and study three different special class of MINLPs and investigate new methods that exploit their structure to solve these problems more efficiently. A common theme in these studies is the use of mixed-integer *linear* programming techniques.

In Chapter 2, we consider a class of problems that are assumed to have convex continuous relaxation. Even though the relaxation is convex, the nonlinear functions defining the problem may not be convex. There are several approaches to solve MINLPs defined by convex functions among which linearization-based methods appear to be more efficient than others. These methods approximate the feasible region of the problem by linear constraints and dynamically add linearizations to refine the approximation. However, because they rely on convexity of nonlinear functions, the linearizations they use for the approximation is no longer valid in our setting. Moreover, they solve some separation problems that are guaranteed to work only under the convexity assumption of nonlinear functions. We propose two linearization-based algorithms, one of which is an extension of LP/NLP branch-and-bound of Quesada-Grossmann [100]. The second algorithm is a hybrid of LP/NLP branch-and-bound and NLP branch-and-bound. Assuming bounded integer variables, we show that each integer solution can be visited no more than once, and that solving each node can be done with a finite number of cuts. This result implies that both of the algorithms converge to the optimal solution. We present computational results which indicate the algorithms are competitive with the standard LP/NLP branch-and-bound in convex MINLP instances. Thus, with these methods, a broader class of problems can be solved without sacrificing computational efficiency. On instances where only continuous relaxations are convex, our methods obtain global optimal values while the standard methods fail to do so in some instances.

In Chapter 3, we study a service network design problem in which, given a set of potential facilities, a network operator is seeking to find a subset of facilities to open and their associated capacity, such that the customer service is balanced with cost of opening facilities. Given the subset of open facilities and their sizes, the customers select facilities to which their demands should be sent. The cost of using a facility for a customer depends on both the distance to the facility and the congestion of the facility. The congestion is a function of demand load on a facility which depends on the decisions of all customers. Therefore, an equilibrium problem determines the demand allocation of the customers. There are some existing heuristic methods proposed to solve this class. However, these methods provide with no guarantee on solution quality. We model this problem as a mixed-integer bilevel program (MIBP.) This structure is motivated by several applications such as designing distribution networks in supply chain, preventive healthcare facility design, and evacuation shelter location allocation. The first approach one might take is to substitute the lower-level problem, which is a convex optimization problem, by its sufficient optimality condition (see (1.2.3.2).) The resulting problem is a MINLP for which a general-purpose global optimization solver can only solve small instances. To obtain lower bounds efficiently and to guide the search for a feasible solution, we derive a Lagrangian relaxation which decomposes the problem into a separate subproblem for each facility. Solving the Lagrangian relaxation yields a lower bound and a candidate set of facilities to open. This set is used to derive an NLP that finds a feasible solution to the original bilevel program. In computational experiments on instances much larger than can be solved by alternative approaches, the values obtained from the relaxation compared with upper bounds obtained by solving the NLP indicate that the relaxation yields strong bounds, and feasible solutions found are close to the optimal solution.

The problem that we study in Chapter 4 is an optimization problem which has a non-convex monotone function of binary variables in its objective function. This function may be interpreted as the solution of a second-stage problem, e.g., a mixed-integer recourse

problem in stochastic programming. The current standard technique for solving this class of problems is the Integer L-shaped (ILS) method in which an inequality is added to eliminate infeasible solutions. However, these cuts are weak in the sense that it only eliminates one infeasible solution at a time. We exploit the monotonicity structure of the function and enhance the ILS cuts to derive stronger cuts, which we call Enhanced Integer L-Shaped inequalities (EILS). We, furthermore, investigate the use of the so-called mixing inequalities to combine the EILS inequalities, which has the potential to further tighten the linear programming relaxation. To obtain the most strengthening from the mixing inequalities, we use a method to generate additional points which result in additional EILS cuts and more points in the mixing inequalities. We test the approach on two applications. As we expected, EILS cuts are modestly more effective than ILS cuts terms of computation time and gap closed. However, even in the most extreme setting in which all possible additional points are generated, mixing inequalities have only marginal impact on the root gap closed. Therefore, these inequalities do not appear promising for helping to solve this class of problems. Ultimately, our effort to exploit monotonicity to improve the solution methods for this class of problems had mixed results, and consequently this class of problems remains very challenging.

1.2 Background

There exist numerous different methods to solve problem (MINLP). However, they are mostly made of some distinct modules. In the following subsection we will review one of these modules, *relaxation*. We also review branch-and-bound technique which is one of the standard approaches to solve (MINLP). We then follow by presenting selected introductory reviews of various topics, such as nonlinear programming, quasiconvex functions, Lagrangian relaxation in integer programming, integer L-shaped method for two-stage stochastic integer programs and bilevel programming.

1.2.1 Relaxation and valid inequality

A relaxation of problem (MINLP) is an optimization problem that can be defined as $\min\{c^T z : x \in X_R\}$ where $X \subset X_R$ ([15].) Typically, a relaxation is interesting when it is significantly easier to solve than (MINLP). If the solution of a relaxation is feasible to (MINLP), it is an optimal solution. The reason is that this feasible solution which yields an upper bound for z^* coincides with a solution that is a relaxation solution.

There are two main strategies used to obtain relaxations of (MINLP).

1. *Continuous relaxation* is obtained by relaxing $x_I \in \mathbb{Z}^p$. The problem (MINLP) then reduces to a nonlinear program:

$$\begin{aligned} z_{\text{NLP}} = \min \quad & c^T x \\ \text{s.t.} \quad & g_j(x) \leq 0, \quad j \in J \\ & x \in \mathbb{R}^n. \end{aligned} \tag{1.1}$$

2. *Polyhedral relaxation* relaxes all constraints where $g_j, j \in J$ are nonlinear or replaces nonlinear constraints by their corresponding supporting hyperplanes. For example, if X is defined by convex functions, then each constraint $g_j(x) \leq 0$ can be relaxed with

$$g_j(\bar{x}) + \nabla g_j(\bar{x})^T (x - \bar{x}) \leq 0, \tag{1.2}$$

for some given $\bar{x} \in \mathbb{R}^n$. This hyperplane is obtained from the first-order Taylor series approximation of g_j .

A method could combine both strategies together, and the resulting problem would be still a relaxation of (MINLP).

An inequality $\pi^T x \geq \pi_0$, or in short (π, π_0) is called a *valid inequality* for X if all $x \in X$ satisfy it. By definition, a relaxation of (MINLP) remains a relaxation by adding any valid inequality. However, adding more constraints to a relaxation may strengthen it, i.e. it may yield a better lower bound. The idea of valid inequalities suggests a simple

algorithm to solve MINLPs. Let x^* denote the solution of relaxation (1.1). If it violates the integrality requirement, one can attempt to find a valid inequality which is violated by x^* . Hence adding this valid inequality would eliminate x^* . In the subsequent iteration, the previous solution would be infeasible and the lower bound obtained might be improved. Under some conditions, this approach can solve (MINLP) as explained in Chapter 2.

1.2.2 Branch-and-bound

Inspired by the *divide-and-conquer* idea by Dakin [45] in 1965, a branch-and-bound scheme decomposes the feasible region X into smaller subregions X_k successively such that $X = X_1 \cup \dots \cup X_K$. In searching, some subregions can be *fathomed*. The term fathoming is used when a subregion has no advantage for further exploration; i.e the subregion either might be infeasible, or a feasible solution is found, or the relaxation value is worse than an already found feasible solution. Therefore it can be ignored in future steps. This observation helps the branch-and-bound approach avoid complete enumeration.

In Algorithm 1, we summarize a generic branch-and-bound algorithm. The algorithm considers solving a problem

$$z^* = \min\{c^T x : x \in X\}. \quad (1.3)$$

```

Set  $T = \{X\}$ ,  $UB = +\infty$  and  $LB = -\infty$ .
while Nodes  $\neq \emptyset$  do
    Let  $S = \text{NodeSelection}(T)$  and  $T = T \setminus \{S\}$ .
     $\bar{z} = \text{FindUpperBound}(S)$ .
    if  $\bar{z} < UB$  then  $UB \leftarrow \bar{z}$ .
     $\underline{z} = \text{FindLowerBound}(S)$ .
    if  $\underline{z} < UB$  then
        Choose  $S_1$  and  $S_2$  such that  $S = S_1 \cup S_2$ .
         $T \leftarrow T \cup \{S_1, S_2\}$ .
end

```

Algorithm 1: Generic branch-and-bound algorithm

Having chosen a node associated with a particular subregion, the algorithm first tries to find an upper bound by the function `FindUpperBound`, which can be obtained by finding a feasible solution to problem (1.3). The value of the obtained feasible solution is returned as $\bar{z}$, which is $+\infty$ if the function fails to find any feasible solution. If $\bar{z}$ is better than the best upper bound UB , the algorithm updates UB . Next in function `FindLowerBound( $S$ )`, the algorithm finds a lower bound for $\min\{c^T x : x \in S\}$. If $\underline{z} > UB$, it is dominated by the best upper bound, and so this node can be fathomed. Otherwise, S is divided into two subregions S_1 and S_2 for further exploration. Among the most crucial decisions in a branch-and-bound algorithm are choosing function `NodeSelection`, which determines which remaining subregion to explore, and *branching*, which partitions the current region S into two subregions S_1 and S_2 for future exploration. There are several methods for each of these two functions whose trade-off of their associated computational complexity and effectiveness falls into strategic decisions of branch-and-bound algorithm. Further details about node selection and branching rules can be found in Achterberg et al. [2] for MILP and in Bonami et al. [25] for MINLP.

In an example, we explain the abstract functions that discussed in the generic branch-and-bound algorithm.

Example 1.1. *To solve a MILP, the problem being solved in each node is the continuous relaxation of MILP which is a Linear Program (LP), obtained by relaxing the integrality requirement on variables. The optimal value of LP in a subregion S is the value of $\text{FindLowerBound}(S)$. If the solution of LP satisfies the integrality requirement, it is a feasible solution to the original solution; hence, its value is an upper bound for z^* and can be used as the value for $\text{FindUpperBound}(S)$.*

Now suppose the solution of the continuous relaxation, $\bar{x}$, is not integer feasible, i.e. there exists $I' \subset I$ such that $\bar{x}_i \notin \mathbb{Z}, i \in I'$. For branching, one could choose $i \in I'$ and define two sets, $S_1 = \{x \in \mathbb{R}^n \mid x \in S, x_i \leq \lfloor \bar{x}_i \rfloor\}$ and $S_2 = \{x \in \mathbb{R}^n \mid x \in S, x_i \geq \lceil \bar{x}_i \rceil\}$.

1.2.3 Concepts in theory of nonlinear programming

When $p = 0$ in problem (MINLP), the problem becomes an NLP. A generic NLP can be stated as

$$\begin{aligned} z_{\text{NLP}} = \min \quad & f(x) \\ \text{s.t.} \quad & g_j(x) \leq 0, \quad j \in I \\ & g_j(x) = 0, \quad j \in E, \end{aligned} \tag{NLP}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g_j : \mathbb{R}^n \rightarrow \mathbb{R}$ are assumed to be continuously differentiable, and I and E are finite index sets of inequality and equality constraints, respectively.

We denote $R = \{x \in \mathbb{R}^n \mid g_j(x) \leq 0, j \in I, g_j(x) = 0, j \in E\}$ as the feasible region of (NLP). If f is a convex function, and the set R is a convex set, (NLP) is called a convex programming problem (CPP). One sufficient condition for convexity of R is that $g_j, j \in I$ are convex and $g_j, j \in E$ are linear.

A point x' is called a *feasible point* if it satisfies all the constraints and a feasible point x^* is called a *local minimizer* if and only if

$$f(x^*) \leq f(x) \quad \forall x \in R \cap N(x^*),$$

where $N(x^*)$ is some neighborhood of x^* . The point x^* is a *global minimizer* if and only if

$$f(x^*) \leq f(x) \quad \forall x \in R.$$

Rather than using the definition to verify optimality of a point, optimality of a given solution $\bar{x}$ can be checked by some necessary optimality conditions. The condition is (Proposition 4.7.2 of [17]): if $\bar{x}$ is the optimal solution to (NLP), then there exists $\bar{s} \in \partial f(\bar{x})$ such that

$$\bar{s}^T(x - \bar{x}) \geq 0, \quad \forall x \in R \quad (1.4)$$

where $\partial f(\bar{x})$ denotes the set of subgradients of f at $\bar{x}$, i.e. $\{s \in \mathbb{R}^n : f(\bar{x}) + s^T(x - \bar{x}) \leq f(x), \forall x \in R\}$.

The problem of finding a global minimizer is difficult and often intractable unless (NLP) is a CPP. In the following subsections, we explain the sufficient conditions of a global minimizer of a CPP.

1.2.3.1 Tangent cone and constraint qualifications

We first define the tangent cone at a feasible point x .

Definition 1.2 ([97]). *A tangent (or tangent vector) to R at a point x is a vector for which there exists a feasible sequence $\{x_k\}$ approaching x and a sequence of positive scalars $\{r_k\}$ approaching 0 such that*

$$\lim_{k \rightarrow \infty} \frac{x_k - x}{r_k} = d.$$

The set of all tangents to R at x^* , denoted by $T_R(x^*)$, is called the tangent cone.

Now, we define the set of linearized directions.

Definition 1.3 ([97]). *Given a feasible solution x and the active set constraint $\mathcal{A}(x) = E \cup \{j \in I | g_j(x) = 0\}$, the set of linearized feasible directions $\mathcal{F}(x)$ is*

$$\mathcal{F}(x) = \left\{ d : \begin{array}{l} d^T \nabla g_i(x) = 0, \quad \forall i \in E, \\ d^T \nabla g_i(x) \leq 0, \quad \forall i \in \mathcal{A}(x) \cap I \end{array} \right\}.$$

A condition under which $\mathcal{F}(x) = T_R(x)$ is called a *constraint qualification*. A simple constraint qualification is the *linear independence constraint qualification (LICQ)*, which holds if $\{\nabla g_j(x^*), j \in E \cup I \text{ s.t. } g_j(x^*) = 0\}$ is linearly independent.

1.2.3.2 Lagrangian function and optimality conditions

In an unconstrained optimization problem, i.e. $|E| = |I| = 0$, a stationary point, which is potentially a local optimizer, is characterized by $\nabla f = 0$. This idea is generalized to constrained optimization by introducing the notion of *Lagrangian function* and *Lagrange multipliers*. The Lagrangian function is a weighted linear combination of the objective function and the constraint functions,

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{j \in I} \lambda_j g_j(x) - \sum_{j \in E} \lambda_j g_j(x),$$

where λ_j is the Lagrange multiplier associated with constraint $j \in E \cup I$. Now the notion Karush-Kuhn-Tucker (KKT) first-order optimality conditions for NLP is stated.

Theorem 1.4 ([97]). *If x^* is a local minimizer of (NLP), then under constraint qualifications at x^* there exist Lagrange multipliers λ^* such that*

$$\nabla f(x^*) - \sum_{j \in I} \lambda_j^* \nabla g_j(x^*) - \sum_{j \in E} \lambda_j^* \nabla g_j(x^*) = 0, \quad (1.5a)$$

$$g_j(x^*) \leq 0, \quad \forall j \in I, \quad (1.5b)$$

$$g_j(x^*) = 0, \quad \forall j \in E, \quad (1.5c)$$

$$\lambda_j^* \geq 0, \quad \forall j \in I, \quad (1.5d)$$

$$\lambda_j^* g_j(x^*) = 0, \quad \forall j \in I. \quad (1.5e)$$

Condition (1.5e) is often referred to as the *complementarity slackness* condition.

Theorem 1.4, which is a necessary condition of a local minimizer of (NLP), becomes a sufficient condition for a CPP. Furthermore, since a local minimizer of a CPP is also a global minimizer, a solution to (1.5) is a global minimizer.

Using the Lagrangian function defined previously, *duality*, which is one of the most interesting properties of CPP, is introduced. Let's define the *Lagrangian dual* as

$$\begin{aligned} w^* = \max q(\lambda) \\ \text{s.t. } \lambda_j \geq 0, \quad j \in I, \end{aligned} \tag{1.6}$$

where $q(\lambda) = \inf_x \mathcal{L}(x, \lambda)$ is called the *dual function*. The notion of *weak duality* states that any feasible solution to the dual problem (1.6) provides a lower bound to the primal problem (NLP). In addition to weak duality, for a CPP under constraint qualification, if x^* is a global minimizer of (NLP) and λ^* solves the dual problem (1.6), then $z_{\text{NLP}} = w^*$. Also, the pair (x^*, λ^*) is feasible to (1.5). In the general case (MINLP), or a nonconvex (NLP), weak duality still holds, but often a duality gap exists.

For extensive details and discussions of theory and algorithms of NLP, one could see [14].

1.2.4 Quasiconvex functions and their properties

A function $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be *quasiconvex* if $\text{dom} g$ is convex and the sublevel set

$$S_\alpha = \{x \in \text{dom} g : g(x) \leq \alpha\}, \quad \forall \alpha \in \mathbb{R}$$

is convex [14, 93]. A function f is *quasiconcave* if $-f$ is quasiconvex. This definition implies that all convex functions are quasiconvex.

An equivalent definition is that if a function g satisfies

$$g(\lambda x_1 + (1 - \lambda)x_2) \leq \max\{g(x_1), g(x_2)\}, \tag{1.7}$$

for all $x_1, x_2 \in \text{dom} g$ and $\lambda \in [0, 1]$, g is quasiconvex.

If g is differentiable and quasiconvex, for $x_1, x_2 \in \text{dom}g$, such that $g(x_1) \leq g(x_2)$, then

$$\nabla g(x_2)^T(x_1 - x_2) \leq 0, \quad (1.8)$$

holds. This inequality is of particular interest since if in an NLP, one has $g(x) \leq 0$ as a constraint, then for a given point $\hat{x} \in \mathbb{R}^n$ with $g(\hat{x}) = 0$, the inequality

$$\nabla g(\hat{x})^T(x - \hat{x}) \leq 0, \quad (1.9)$$

is a tight supporting hyperplane for the feasible region.

Bazaraa et al. [14, Theorem 4.2.16, pp 195] show that for an NLP with convex objective function and constraints defined by quasiconvex functions, a solution to the KKT conditions is a global optimal solution.

1.2.5 Lagrangian relaxation in integer programming

We discussed the Lagrangian function in nonlinear programming in Section 1.2.3.2. This idea is also used in integer programming explained in this section. More details can be found in [119].

Consider the integer program

$$\begin{aligned} z = & \min c^T x \\ \text{s.t. } & Dx \leq d \\ & x \in X, \end{aligned} \quad (1.10)$$

where $X \subset \mathbb{Z}_+^n$ and $D \in \mathbb{R}^{m \times n}, d \in \mathbb{R}^m$.

For problem (1.10), given any vector $u \in \mathbb{R}_+^m$ we define the *Lagrangian relaxation problem* as

$$\begin{aligned} z(u) = & \min c^T x - u^T(d - Dx) \\ \text{s.t. } & x \in X. \end{aligned} \quad (1.11)$$

It is easy to show that for any $u \in \mathbb{R}_+^m$, $z(u) \leq z$. The *Lagrangian Dual Problem* is then stated as

$$w = \max\{z(u) : u \geq 0\}. \quad (\text{LD})$$

One method to solve (LD) is the *subgradient* method. Algorithm 2 summarizes the steps of the subgradient method to solve (LD).

```

Initialize  $u = u^0$  and  $k = 0$ .
Initialize the step length  $\mu^0$ .
while stopping criterion not met do
    Solve the Lagrangian problem (1.11) for  $u = u^k$ . Let  $x^k$  denote the solution.
     $u^{k+1} = \max\{u^k - \mu_k(d - Dx^k), 0\}$ .
    Update step length  $\mu_{k+1}$ .
     $k \rightarrow k + 1$ .
end

```

Algorithm 2: Subgradient algorithm to solve (LD).

Updating the step length is a challenging part of the subgradient method, for which many different approaches have been proposed. For example, one method is geometric updating, i.e. $\mu_{k+1} = \rho \cdot \mu_k$ for some $\rho \in (0, 1)$.

1.2.6 Integer L-shaped method for two-stage stochastic integer programs

A two-stage stochastic integer program can be stated as

$$\begin{aligned}
 \min \quad & c^T x + \mathbb{E}_P[Q(x, \omega)] \\
 \text{s.t.} \quad & Ax = b \\
 & x \in \mathbb{R}_+^{n_1 - p_1} \times \mathbb{Z}_+^{p_1}
 \end{aligned} \quad (1.12)$$

where

$$\begin{aligned}
Q(x, \omega) = \min \quad & q_\omega^T y \\
\text{s.t.} \quad & Wy = h_\omega - T_\omega x \\
& y \in \mathbb{R}_+^{n_2 - p_2} \times \mathbb{Z}_+^{p_2},
\end{aligned} \tag{1.13}$$

for some integers n_1, p_1, n_2 and p_2 where $p_1 \leq n_1$ and $p_2 \leq n_2$. Problems (1.12) and (1.13) are called the *first-stage* and *second-stage* problems, respectively.

There are some difficulties in dealing with (1.12). First, evaluating the second-stage problem (1.13) for a fixed first-stage decision x and a particular realization of the uncertain parameter ω may be computationally expensive. Second, evaluation of the expected value of the second-stage cost for a fixed first-stage decision can be computationally expensive as in case of a continuous distribution for P , one needs to integrate $Q(x, \cdot)$, which, in case of discrete distribution, involves solving potentially many integer programs each of which corresponds to one sample taken from the random parameter ω . Finally, since the value function $Q(x, \omega)$ is nonconvex, optimizing the expected second-stage cost also involves difficulties.

If the first-stage variables x are all binary, then problem (1.12) can be reformulated as

$$\min \quad c^T x + \theta \tag{1.14a}$$

$$\text{s.t.} \quad Ax = b, \tag{1.14b}$$

$$\theta \geq \mathbb{E}_P[Q(x, \omega)], \tag{1.14c}$$

$$x \in \{0, 1\}^{n_1}. \tag{1.14d}$$

This problem is relaxed to the following master problem:

$$\min \quad c^T x + \theta \quad (1.15a)$$

$$\text{s.t.} \quad Ax = b, \quad (1.15b)$$

$$\theta + E_k x \geq e_k, \quad k = 1, \dots, s, \quad (1.15c)$$

$$D_k x \geq d_k \quad k = 1, \dots, s, \quad (1.15d)$$

$$x \in \{0, 1\}^{n_1} \quad (1.15e)$$

where s is the number of iterations, and (1.15c) and (1.15d) are the optimality and feasibility cuts obtained so far. We assume that there exists $L > -\infty$ such that $\mathbb{E}_P[Q(x, \omega)] \geq L$. Let $(\bar{x}^v, \hat{\theta}^v)$ be the solution of problem (1.15) in iteration v . Laporte and Louveaux [88] show that

$$\theta \geq (\hat{\theta}^v - L) \left(\sum_{i \in S_v} x_i - \sum_{i \notin S_v} x_i \right) - (\hat{\theta}^v - L)(|S_v| - 1) + L, \quad (1.16)$$

is an optimality cut where $S_v = \{i \mid \bar{x}^v = 1\}$. This inequality can be used to better refine the search space for the first-stage variable x by evaluations of the second-stage problem. Therefore, in each step of the algorithm, adding (1.16) would strengthen the master problem (1.15).

The integer L-shaped method is a branch-and-bound approach that starts by solving the master problem, and then by evaluating the second-stage problem in a branch-and-bound tree, updates the relaxation problem by adding new (1.15c), (1.15d) and (1.16).

1.2.7 Bilevel programming

In classic optimization problems, one decision maker decides on a set of decisions simultaneously. We call these problems *single level* optimization problems. There are also some hierarchical decision systems in which one decision maker decides on a set of decisions in two or more stages, e.g. multi-stage stochastic programming problems. By contrast, a bilevel programming problem incorporates multiple decision makers, typically

non-cooperative ones, in a hierarchical decision problem in which decisions are divided into groups, each of which is controlled by a particular decision maker. A key property of a bilevel program is the dependence of the lower-level decision makers' feasible region on the upper-level decisions; i.e. decisions made by the leader affect future decisions of the followers. Each of the decision makers, leader or follower, acts based on its own interests. However, decisions made by followers may alter the objectives that the leader can accomplish. Therefore, the leader must incorporate the reactions of the followers in its decision making which implies that the leader's objective may depend on the followers decisions as well as its own. Marcotte and Savard [94] provide more details about the bilevel programming.

A mixed-integer bilevel program with a continuous lower level problem is

$$\min f^1(x, y) \quad (1.17a)$$

$$\text{s.t. } g^1(x, y) \leq 0, \quad (1.17b)$$

$$x \in \mathbb{R}^{n_1-p_1} \times \mathbb{Z}^{p_1}, \quad (1.17c)$$

$$y \in \operatorname{argmin} \left\{ f^2(x, y) : \begin{array}{l} g^2(x, y) \leq 0, \\ y \in \mathbb{R}^{n_2} \end{array} \right\}, \quad (1.17d)$$

where x and y are the upper-level and lower-level decision variables respectively, $f^1 : \mathbb{R}^{n_1+n_2} \rightarrow \mathbb{R}$ and $f^2 : \mathbb{R}^{n_1+n_2} \rightarrow \mathbb{R}$ are, respectively, objective functions of upper-level and lower-level problems, and $g^1 : \mathbb{R}^{n_1+n_2} \rightarrow \mathbb{R}^{m_1}$ and $g^2 : \mathbb{R}^{n_1+n_2} \rightarrow \mathbb{R}^{m_2}$ are functions defining the upper-level and lower-level constraints, respectively. The lower-level problem is embedded as a parametric decision problem into the first-level problem.

There are two approaches to bilevel programming: (a) optimistic bilevel programming, and (b) pessimistic bilevel programming. Let's define the set of reactions of followers to the leader's decision $\hat{x}$ as follows:

$$R(\hat{x}) = \{y \in \mathbb{R}^{n_2-p_2} \times \mathbb{Z}^{p_2} : y \in \operatorname{argmin}\{f^2(\hat{x}, y) : g_j^2(\hat{x}, y) \leq 0, j = 1, \dots, m_2\}\}.$$

This set contains all rational responses of the followers to a decision $\hat{x}$ made by the leader. In the optimistic approach, the leader chooses an element of $R(\hat{x})$ that agrees with its objective best, unless $R(\hat{x})$ is a singleton. The pessimistic approach is followed when cooperation of leader and follower is impossible or not allowed, or the leader is risk-averse and prefers to reduce the risk of any inadmissible decision by the followers.

In problem (1.17), the lower-level problem is continuous. Therefore, constraint (1.17d) can be replaced by the corresponding optimality condition, provided that the problem is convex. The resulting reformulation is a mixed-integer nonlinear program:

$$\begin{aligned}
 \min \quad & f^1(x, y) \\
 \text{s.t.} \quad & g^1(x, y) \leq 0, \\
 & x \in \mathbb{R}^{n_1-p_1} \times \mathbb{Z}^{p_1}, \\
 & \nabla f^2(x, y) - \sum_{j=1}^{m_2} \nabla \lambda_j \nabla g_j(x, y) = 0 \\
 & 0 \leq g_j(x, y) \perp \lambda_j \geq 0, \quad j = 1, \dots, m_2,
 \end{aligned} \tag{1.18}$$

where $\lambda_j, j = 1, \dots, m_2$ are the dual multipliers associated with g_j .

Wide range of applications fit into the structure of bilevel programming. In 1970's, bilevel programming and some of its applications in defense as well as production decision making problems were initially investigated by Bracken and McGill in a series of papers [28], [29] and [30]. They named these problems *mathematical programs with optimization problems in the constraints* reflected in (1.17).

One application of bilevel programming is the toll pricing problem [85], [86], where a system operator wishes to maximize the revenue gained from tolls imposed on a road system, which depends directly on what system users (drivers) decide to do. The toll prices must be set to a level that maximizes revenue while system users remain willing to use that road system. This setting can be applied to more general problems of tariff imposition, e.g. [31].

Another application of bilevel programming is *network interdiction* for homeland security and critical infrastructure protection (see for instance [118], [96], [61], [120], [42],

[76], [73], [78].) Typically, the follower is the network operator and is interdicted by the leader who attempts to restrict the performance of the network by removing subsets of the network.

Other applications of bilevel programming include revenue management [43], network design [41], national agriculture planning [58], and decentralized management of multidivisional firms [11]. For further information about applications of bilevel programming, see [57].

The feasible region of bilevel programming problems is often nonconvex even in the simplest case in which all functions are linear and all variables are continuous. Deng [47] showed that finding an $\epsilon > 0$ feasible solution to a linear bilevel program is $\mathcal{NP}$ -hard.

Methods to solve bilevel programming problems can be divided into two sets: methods to solve continuous bilevel programming problems, i.e. $p_1 = p_2 = 0$ in (1.17), and methods to solve integer problems.

For continuous bilevel programs methods include extreme-point approaches ([35],[19]), branch-and-bound (when the lower-level problem is replaced by KKT conditions) ([52] and [58] for linear bilevel programs, [6] and [50] for the quadratic case), complementary pivoting ([18]), sequential linear complementarity ([79], [81] and [80]), descent methods ([10] for general problems and [115] for convex quadratic bilevel programs), penalty function methods ([4], [5],[75]) and trust region ([39].) For a thorough overview of these methods, see [40].

The bilevel programming problem (1.17) has only continuous variables in the lower-level problem. Here we list some approaches for bilevel programs with integer variables in their lower-level problems. For mixed-integer bilevel programs, even with linear functions (MIBLP), Moore and Bard [10] proposed the first branch-and-bound algorithms for MIBLP under several simplifying assumptions. The authors presented another branch-and-bound algorithm in [12] to solve problems with only binary variables. However, their

algorithm is not guaranteed to converge to the optimal solution. Later in 2009, DeNegre and Ralphs [46] devised an algorithm to solve MIBLP in which y never shows up in upper-level constraints. Saharidis and Ierapetritou [104] presented a Benders' decomposition approach.

Chapter 2

Linearization-Based Algorithms for Mixed-Integer Nonlinear Programs with Convex Continuous Relaxation

This chapter is based on material in the paper [71].

2.1 Introduction

A mixed-integer nonlinear program (MINLP) can be expressed as

$$\begin{aligned}
 z_{\text{MINLP}} = \min_{x,y} \quad & c^T x + d^T y \\
 \text{s.t.} \quad & (x, y) \in \mathcal{C} \\
 & y \in \mathbb{Z}^q
 \end{aligned} \tag{MINLP}$$

where $\mathcal{C} = \{(x, y) \in X : g_j(x, y) \leq 0, \forall j \in J\}$, J is the set of indices of nonlinear constraints, $X \subseteq \mathbb{R}^{n+q}$ is a polyhedron, $c \in \mathbb{R}^n$, and $d \in \mathbb{R}^q$. We assume that $\mathcal{C}$ is convex, the functions $g_j, j \in J$ are continuously differentiable on X , and the set X includes finite bounds, $L \leq y \leq U$, on the integer decision variables, where $L, U \in \mathbb{R}^q$. Most existing linearization-based algorithms for MINLPs further assume that the functions $g_j, j \in J$ are convex. Our goal is to derive algorithms that guarantee global optimality under the weaker assumption that the set $\mathcal{C}$ is convex, even though the functions $g_j, j \in J$ may not necessarily be convex.

MINLPs have been used to model a very broad class of applications, such as those in [37, 55, 87, 77, 63], to name just a few. However, solving MINLPs to global optimality

is difficult because they combine two computational challenges: optimizing over integer variables and handling nonlinear functions. For example, the special case of MINLP in which $J = \emptyset$, mixed-integer linear programming (MILP), is $\mathcal{NP}$ -hard [59]. Nevertheless, if the nonlinear functions $g_j, j \in J$ are convex, algorithms have been developed that can solve (MINLP) to global optimality, because in this case the continuous relaxation becomes a convex nonlinear program (NLP) which can be efficiently solved. We refer to the class of MINLPs in which the constraint functions $g_j, j \in J$ are convex as *convex* MINLPs.

Our assumption that $\mathcal{C}$ is convex is satisfied when the nonlinear functions $g_j, j \in J$ are convex. However, this is not a necessary condition for $\mathcal{C}$ to be convex. For example, consider the following MINLP instance:

$$\begin{aligned} \min \quad & x + 3y \\ \text{s.t.} \quad & -xy + 2 \leq 0 \\ & 5x + 7y \leq 35 \\ & -5x + 8y \leq 24 \\ & x \geq 0, \quad y \in \mathbb{Z}_+. \end{aligned}$$

Figure 2.1 shows the continuous relaxation of this example, which is convex, while the first constraint is defined by a nonconvex function. In general, the set $\mathcal{C}$ is guaranteed to be convex if the functions, $g_j, j \in J$ are all quasiconvex. We refer to MINLPs in which the constraint functions $g_j, j \in J$ are all quasiconvex as quasiconvex MINLPs.

The algorithms we present can solve quasiconvex MINLPs because for such problems $\mathcal{C}$ is convex. However, the algorithms can also solve MINLP instances with functions $g_j, j \in J$ that are not necessarily quasiconvex, as long as the set $\mathcal{C}$ is still convex. For example, the set $\tilde{\mathcal{C}} = \{(x, y) \in \mathbb{R}_+^2 : y - \sin(x) \leq 0, y \geq x^2\}$ is convex, while $g_1(x, y) = y - \sin(x)$ is neither convex nor quasiconvex over $\mathbb{R}_+^2$.

Although the objective function of (MINLP) is assumed to be linear, every MINLP with a convex objective function $f(\cdot)$ can be equivalently cast into this form by introducing

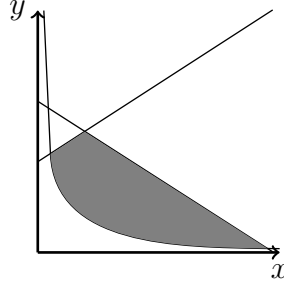


Figure 2.1 Convexity of the continuous relaxation in a quasiconvex MINLP.

an auxiliary variable η for the objective function, adding a new constraint of the form $\eta \geq f(x, y)$ and minimizing η . Note, however, that if the objective $f(\cdot)$ is quasiconvex, the set defined by the constraint $\eta \geq f(x, y)$ is not necessarily convex. Thus, while our proposed methods can solve MINLPs with quasiconvex constraints, they are not guaranteed to solve MINLPs with a quasiconvex objective.

NLP-based branch-and-bound [45], in which a nonlinear programming relaxation of (MINLP) is solved at each node, can also solve MINLPs with convex continuous relaxation, provided the NLP solver can solve the corresponding relaxations to global optimality. Another class of methods for solving MINLP are linearization-based algorithms, and most existing versions of these algorithms assume the constraint functions $g_j, j \in J$ are convex. These algorithms use linearizations of the nonlinear functions defining the constraints to construct a polyhedral outer approximation of the original feasible region, and thus obtain linear and mixed-integer linear approximations of the MINLP. This class includes generalized Benders decomposition [60], outer approximation [49], LP/NLP-based branch-and-bound (LP/NLP-BB for short) [100], and the extended cutting-plane algorithm [117]. We describe outer approximation and LP/NLP-BB in detail in section 2.3. In a recent computational study, Abhishek et al. [1] found that linearization-based algorithms tend to be more efficient than NLP-based branch-and-bound on moderate and hard convex MINLP instances, primarily because the main subproblems in these algorithms are *linear*

programs, and relatively few nonlinear programs need to be solved. Thus, while NLP-based branch-and-bound can solve MINLPs with convex continuous relaxation, there may be computational benefit to deriving more linearization-based algorithms that can also solve this more general class of problems.

There are two reasons why most linearization-based algorithms, such as outer-approximation and LP/NLP-BB, are not guaranteed to find a global optimal solution of (MINLP) if the functions $g_j, j \in J$ are not convex, even when $\mathcal{C}$ is convex. First, these algorithms are based on obtaining an outer approximation of $\mathcal{C}$ constructed using inequalities of the form

$$g_j(\bar{z}) + \nabla g_j(\bar{z})^T(z - \bar{z}) \leq 0 \quad (2.1)$$

where $z = (x, y)$, and $\bar{z} = (\bar{x}, \bar{y})$ is a point in X . Inequality (2.1) is valid for $\mathcal{C}$ if the function g_j is convex, but may incorrectly remove points in $\mathcal{C}$ if g_j is not convex. The second reason outer approximation and LP/NLP-BB may fail is in the handling of vectors $y \in \mathbb{Z}^q$ for which $y \notin \text{proj}_y(\mathcal{C}) := \{y \in \mathbb{R}^q : \exists x \text{ with } (x, y) \in \mathcal{C}\}$. When such a vector is obtained as a solution of an outer approximating linear program (LP) or MILP, these algorithms must find a valid inequality that cuts off this infeasible solution. The standard methods for obtaining such an inequality may fail when the functions $g_j, j \in J$ are not convex. (We provide more details of this second challenge in section 2.3.)

Two existing linearization-based algorithms have been proposed that partially meet our goal of solving MINLPs without assuming the functions $g_j, j \in J$ are convex. First, the α -ECP algorithm has been proposed to solve MINLPs in which the functions $g_j, j \in J$ are pseudoconvex [99, 109]. Pseudoconvex functions generalize convex functions, but are a special case of quasiconvex functions. Since inequality (2.1) is not necessarily valid for pseudoconvex functions, the α -ECP algorithm finds α such that

$$g_j(\bar{z}) + \alpha \nabla g_j(\bar{z})^T(z - \bar{z}) \leq 0$$

is valid for $\mathcal{C}$. To find such an admissible value for α , the algorithm solves a sequence of MILPs. More recently, Bonami et al. [23] derived an extension of the outer approximation

algorithm that only requires $\mathcal{C}$ to be convex. They prove that the inequality

$$\nabla g_j(\bar{z})^T(z - \bar{z}) \leq 0 \quad (2.2)$$

is valid for $\mathcal{C}$ when $g_j(\bar{z}) = 0$, and this serves the basis of their algorithm. This algorithm is similar to the first algorithm we propose, except that their algorithm uses their proposed feasibility pump heuristic to obtain feasible solutions, and we extend these ideas to obtain a variant of the LP/NLP-BB algorithm. In addition, we propose an alternative approach for eliminating infeasible integer solutions $y \notin \text{proj}_y(\mathcal{C})$.

We propose two linearization-based algorithms that can solve MINLPs with convex continuous relaxation $\mathcal{C}$. Although we do not assume $\mathcal{C}$ is defined by convex functions, we do take it as an assumption that the NLP subproblems are solved to global optimality. This assumption is reasonable for quasiconvex MINLPs, but may be a limitation more generally. As in [23], the algorithms that we propose use inequality (2.2) to obtain polyhedral outer approximations of the set $\mathcal{C}$. Our algorithms differ in the method used for eliminating infeasible integer solutions $y \notin \text{proj}_y(\mathcal{C})$. Our first algorithm is an adaptation of the LP/NLP-BB algorithm that uses (2.2) in place of (2.1), and also solves a different feasibility problem for infeasible integer solutions. The second algorithm, referred to as Hybrid-BB, is a hybrid algorithm between NLP branch-and-bound and LP/NLP-BB. Hybrid-BB solves LP relaxations based on the current outer approximation, as in LP/NLP-BB, but when the LP relaxation yields an integer feasible solution, Hybrid-BB solves the NLP relaxation as in the NLP-based branch-and-bound algorithm. An interesting feature of this algorithm is that branching may be required at a node even when the outer approximating LP relaxation at a node yields an integer feasible solution.

The focus of this chapter is on linearization-based algorithms to solve MINLPs with convex continuous relaxation. These problems can also be solved by global optimization methods that do not require the continuous relaxation to be convex, see e.g., [7, 95, 32, 56,

62, 111, 112]. The disadvantage of using these global optimization methods for this problem class is that they do not take advantage of the structure that the continuous relaxation is convex.

The organization of the chapter is as follows. Since the methods proposed in this paper extend outer approximation and LP/NLP-BB to solve MINLPs, the main steps of these algorithms are reviewed in section 2.2. Then, in section 2.3 we explain why these algorithms may fail to solve MINLP if the functions $g_j, j \in J$ are not convex, and then describe two algorithms that overcome this limitation. We present results of computational experiments on our proposed algorithms in section 2.4 and conclude in section 2.5.

2.2 Algorithms for Convex MINLP Problems

Several linearization-based algorithms have been developed for solving convex MINLPs. The basic idea of these algorithms is to find lower and upper bounds on z_{MINLP} , where lower bounds are obtained by solving a problem based on a polyhedral outer approximation of the feasible region and upper bounds are obtained by feasible solutions. These bounds are improved until the optimal solution is achieved. We first describe the subproblems these algorithms require, then explain how they are used to obtain the algorithms.

The first subproblem is called the *NLP relaxation*,

$$\begin{aligned} z_{\text{NLP}[\bar{L}, \bar{U}]} &= \min_{x,y} \quad c^T x + d^T y \\ \text{s.t.} \quad & (x, y) \in \mathcal{C} \\ & y \in [\bar{L}, \bar{U}], \end{aligned} \tag{NLP[\bar{L}, \bar{U}]}$$

where $\bar{L}, \bar{U} \in \mathbb{R}^q$ are given bounds on the integer variables. If $\bar{L}$ and $\bar{U}$ are set to L and U , the initial bounds on y , $z_{\text{NLP}[\bar{L}, \bar{U}]}$ provides a valid lower bound for z_{MINLP} . In this case if $\text{NLP}[\bar{L}, \bar{U}]$ is infeasible, (MINLP) is infeasible. More generally, at a node within a branch-and-bound tree, $\bar{L}$ and $\bar{U}$ may represent restrictions on y defined by the branching

decisions made to obtain this node, in which case $\text{NLP}[\bar{L}, \bar{U}]$ provides a lower bound on the optimal solution at this node.

The second subproblem is *fixed NLP*, which, for a given $\hat{y} \in \mathbb{Z}^q$, is the problem

$$\begin{aligned} z_{\text{NLP}(\hat{y})} &= \min_x c^T x + d^T \hat{y} \\ \text{s.t.} \quad & g_j(x, \hat{y}) \leq 0, \\ & (x, \hat{y}) \in X. \end{aligned} \tag{NLP(\hat{y})}$$

Let $(\bar{x}, \hat{y})$ denote the optimal solution of $\text{NLP}(\hat{y})$, if feasible. In this case, $(\bar{x}, \hat{y})$ is an integer feasible solution to (MINLP), and consequently $z_{\text{NLP}(\hat{y})}$ is an upper bound for z_{MINLP} . The problem $\text{NLP}(\hat{y})$ may be infeasible. Detecting infeasibility requires solving a *feasibility problem* [1]. NLP solvers, such as FilterSQP [54] and IPOPT [116], often solve the feasibility problem by solving an NLP that minimizes some norm of violations of the constraints. In case of ℓ_1 -norm, the feasibility problem

$$\begin{aligned} \min_x \quad & \sum_{j \in J} w_j \max\{g_j(x, \hat{y}), 0\} \\ \text{s.t.} \quad & (x, \hat{y}) \in X \end{aligned} \tag{FNLP(\hat{y})}$$

is solved for some $w \in \mathbb{R}_+^{|J|}$.

In convex MINLPs, which assume the functions $g_j, j \in J$ are convex on X , given a point $(\hat{x}, \hat{y}) \in X$, the linearization (or gradient) inequality

$$g_j(\hat{x}, \hat{y}) + \nabla g_j(\hat{x}, \hat{y})^T \begin{pmatrix} x - \hat{x} \\ y - \hat{y} \end{pmatrix} \leq 0, \tag{2.3}$$

is valid for $\mathcal{C}$ for all $j \in J$. In addition, if $(\hat{x}, \hat{y})$ violates constraint $j \in J$, i.e., $g_j(\hat{x}, \hat{y}) > 0$, then $(\hat{x}, \hat{y})$ violates (2.3) and so adding inequality (2.3) to the outer approximation eliminates $(\hat{x}, \hat{y})$.

When the functions $g_j, j \in J$ are convex, it has been shown that (MINLP) has an equivalent MILP reformulation of finite size, provided the functions $g_j, j \in J$ are continuously differentiable and a certain constraint qualification assumption holds (see [49, 53]).

In other words, the feasible solutions of (MINLP) coincide with those of this MILP. This MILP can be constructed by taking linearizations (2.3) over the points in a set $K = \{(\hat{x}, \hat{y}) : \hat{y} \in \mathbb{Z}^q, L \leq \hat{y} \leq U\}$, where for each $\hat{y} \in \mathbb{Z}^q$ with $L \leq \hat{y} \leq U$, $\hat{x}$ is an optimal solution to $\text{NLP}(\hat{y})$ if it is feasible, and otherwise $\hat{x}$ is an optimal solution to $\text{FNLP}(\hat{y})$. Hence, the outer approximating MILP can be written as

$$\begin{aligned} z_{\text{OA}} = \min_{x,y} \quad & c^T x + d^T y \\ \text{s.t.} \quad & (x, y) \in S_C(K) \\ & (x, y) \in X, y \in \mathbb{Z}^q, \end{aligned} \tag{OA(K)}$$

where $S_C(K) = \{(x, y) : (2.3) \forall j \in J, (\hat{x}, \hat{y}) \in K\}$. This MILP solves the corresponding MINLP problem, provided the *constraint qualification assumption* holds, which states that for every $\hat{y} \in \mathbb{Z}^q$ with $L \leq \hat{y} \leq U$, if $\text{NLP}(\hat{y})$ is feasible, then a constraint qualification holds at the selected optimal solution $(\hat{x}, \hat{y})$, and otherwise a constraint qualification holds in $\text{FNLP}(\hat{y})$ for the selected optimal solution $(\hat{x}, \hat{y})$. Explicitly using the MILP formulation $\text{OA}(K)$ is not feasible because $|K|$ can be impractically large. To overcome this difficulty, Duran and Grossmann [49] proposed the outer approximation algorithm.

The outer approximation algorithm begins with a small (tractable size) subset of linearization points $\mathcal{K} \subset K$ and then solves the MILP $\text{OA}(\mathcal{K})$. Typical implementations of the outer approximation algorithm initialize the set $\mathcal{K}$ with the solution obtained by solving $\text{NLP}[L, U]$. Since $\text{OA}(\mathcal{K})$ is a relaxation of (MINLP), its optimal value, $z_{\text{OA}}(\mathcal{K})$, provides a lower bound for z_{MINLP} . If $\text{OA}(\mathcal{K})$ is infeasible, then (MINLP) is infeasible. Otherwise, the solution $(\hat{x}, \hat{y})$ of $\text{OA}(\mathcal{K})$ is a candidate to construct and solve $\text{NLP}(\hat{y})$. If feasible, the optimal value of $\text{NLP}(\hat{y})$ is an upper bound to MINLP. Moreover, if $(\hat{x}, \hat{y})$ is not feasible to MINLP, then the linearization inequalities (2.3), taken at the solution of $\text{NLP}(\hat{y})$, remove the solution $(\hat{x}, \hat{y})$ from $\text{OA}(\mathcal{K})$. If $\text{NLP}(\hat{y})$ is infeasible, $\text{FNLP}(\hat{y})$ is solved. In either case, the set $\mathcal{K}$ is updated with the optimal solution of the subproblem, and accordingly new linearizations are added to $S_C(\mathcal{K})$. The algorithm proceeds until the

lower and upper bounds are within a pre-specified tolerance. Assuming the integer variables are bounded, and making the constraint qualification assumption, this algorithm is proved to converge to an optimal solution of (MINLP) in a finite number of iterations, since the set K is finite. Algorithm 3 presents the pseudocode of the outer approximation

algorithm. The outer approximation algorithm has been implemented in DICOPT [64] and Bonmin [22].

```

Let  $z_u = +\infty$ ,  $z_l = -\infty$  and set  $\epsilon \geq 0$  as the termination tolerance.
Solve  $\text{NLP}[L, U]$  and denote  $(\bar{x}, \bar{y})$  as its optimal solution.
Let  $\mathcal{K} = \{(\bar{x}, \bar{y})\}$ .
while  $z_u - z_l > \epsilon$  and  $\text{OA}(\mathcal{K})$  is feasible do
    Solve  $\text{OA}(\mathcal{K})$ .
    if  $\text{OA}(\mathcal{K})$  is infeasible then
        Break. (MINLP) is infeasible.
    Let  $z_l = z_{\text{OA}}(\mathcal{K})$  and denote  $(\hat{x}, \hat{y})$  as its optimal solution.
    if  $(\hat{x}, \hat{y}) \in \mathcal{C}$  then
        Break.  $(\hat{x}, \hat{y})$  is an optimal solution of (MINLP).
    Solve  $\text{NLP}(\hat{y})$ .
    if  $\text{NLP}(\hat{y})$  is feasible then
        Let  $(\bar{x}, \hat{y})$  be an optimal solution of  $\text{NLP}(\hat{y})$ .
        Update  $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\bar{x}, \hat{y})\}$ , and accordingly  $S_C(\mathcal{K})$  by new linearizations
        (2.3).
        Update  $z_u = \min\{z_u, z_{\text{NLP}}(\hat{y})\}$ .
    else
        Solve  $\text{FNLP}(\hat{y})$  and let  $(\bar{x}, \hat{y})$  be an optimal solution.
        Update  $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\bar{x}, \hat{y})\}$ , and accordingly  $S_C(\mathcal{K})$  by new linearizations
        (2.3).
 $(\hat{x}, \hat{y})$  is an  $\epsilon$ -optimal solution of (MINLP).

```

Algorithm 3: The Outer Approximation Algorithm.

The downside of the outer approximation algorithm is that it needs to solve a sequence of MILPs in which the number of constraints is increasing. Quesada and Grossmann

[100] proposed an extension to this method, known as the LP/NLP-based branch-and-bound algorithm (LP/NLP-BB), which uses a single branch-and-bound search tree. In this algorithm, the set $\mathcal{K}$ is again initialized by the solution of $\text{NLP}[L, U]$. The algorithm then proceeds as a standard branch-and-bound algorithm, except that the processing of a node in the branch-and-bound tree is slightly different. Algorithms 4 and 5 provide the pseudocode for the steps that LP/NLP-BB takes in a node of the branch-and-bound search tree when solving a convex MINLP. In this algorithm, $\bar{L}$ and $\bar{U}$ represent the bounds on y defined by the current node in the search tree, and $\mathcal{K}$ is the set of linearization points collected throughout the algorithm. At each node, given the current outer approximation $S_C(\mathcal{K})$, the LP relaxation of $\text{OA}(\mathcal{K})$,

$$\begin{aligned} z_{\text{LP}}(\mathcal{K}) = \min_{x,y} \quad & c^T x + d^T y \\ \text{s.t.} \quad & (x, y) \in S_C(\mathcal{K}) \\ & (x, y) \in X, \ y \in [\bar{L}, \bar{U}], \end{aligned} \tag{LP}_C(\mathcal{K}, \bar{L}, \bar{U})$$

is solved. Whenever the LP relaxation at a node yields a solution $(\hat{x}, \hat{y})$ with $\hat{y} \in \mathbb{Z}^q$, the algorithm solves $\text{NLP}(\hat{y})$ and, if needed, $\text{FNLP}(\hat{y})$. The set of linearizations, $S_C(\mathcal{K})$, is then updated by updating $\mathcal{K}$ with the solution of the corresponding subproblem, and the linear program $\text{LP}_C(\mathcal{K}, \bar{L}, \bar{U})$ is re-solved. Finite convergence of LP/NLP-BB follows from finiteness of the outer approximation method (so only finitely many cuts can be added) and boundedness of the integer variables (so only finitely many nodes will be explored).

Solvers such as Bonmin [22], FilMINT [1] and MINOTAUR [92] have an implementation of the LP/NLP-based branch-and-bound algorithm. For a review on algorithms for convex MINLPs, readers are also referred to Bonami et al. [24].

Algorithm: ProcessNode($\bar{L}, \bar{U}$)

NodeDone = FALSE.

while NodeDone = FALSE **do**

 Solve $LP_C(\mathcal{K}, \bar{L}, \bar{U})$.

if $LP_C(\mathcal{K}, \bar{L}, \bar{U})$ is infeasible or $z_{LP}(\mathcal{K}) \geq z_u$ **then**

 Prune the node.

 NodeDone = TRUE.

else

 Let $(\hat{x}, \hat{y})$ be the solution of $LP_C(\mathcal{K}, \bar{L}, \bar{U})$.

if $\hat{y} \in \mathbb{Z}^q$ and $(\hat{x}, \hat{y}) \in \mathcal{C}$ **then**

 Update $z_u = z_{LP}(\mathcal{K})$.

 Prune the node.

 NodeDone = TRUE.

else if $\hat{y} \in \mathbb{Z}^q$ **then**

 ProcessIntegerConvex($\mathcal{K}, z_u, \hat{y}$).

else

 Branch on a variable y_i with $\hat{y}_i \notin \mathbb{Z}$ to create two new subproblems.

 NodeDone = TRUE.

Algorithm 4: Processing a node in LP/NLP-based branch-and-bound for convex MINLPs; z_u represents the best upper bounds obtained so far in the algorithm.

Algorithm: ProcessIntegerConvex($\mathcal{K}, z_u, \hat{y}$)

Solve NLP($\hat{y}$).

if NLP($\hat{y}$) *is feasible* **then**

 Let $(\bar{x}, \hat{y})$ be an optimal solution of NLP($\hat{y}$).

 Update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\bar{x}, \hat{y})\}$ and accordingly $S_C(\mathcal{K})$ by new linearizations (2.3).

 Set $z_u = \min\{z_u, z_{\text{NLP}(\hat{y})}\}$.

else

 Solve FNLP($\hat{y}$) and let $(\bar{x}, \hat{y})$ be an optimal solution.

 Update $\mathcal{K} = \mathcal{K} \cup \{(\bar{x}, \hat{y})\}$ and accordingly $S_C(\mathcal{K})$ by new linearizations (2.3).

Algorithm 5: Processing integer solutions in a node in LP/NLP-based branch-and-bound for convex MINLPs.

2.3 Proposed Algorithms

In this section, we introduce two linearization-based algorithms to solve MINLPs with convex continuous relaxation $\mathcal{C}$. First, we review the challenges with existing linearization-based algorithms to solve these MINLPs. Then we present our proposed algorithms.

The first challenge is that the linearization inequality (2.1) is not necessarily valid for $\mathcal{C}$ if some of the functions $g_j, j \in J$ defining the feasible region are not convex. This challenge can be easily resolved by restricting the use of (2.1) to cases when it is guaranteed to be a valid inequality for $\mathcal{C}$ even when $g_j, j \in J$ are not all convex. In a quasiconvex MINLP, since all $(x, y) \in \mathcal{C}$ must satisfy $g_j(x, y) \leq 0$, by (1.9) the inequality

$$\nabla g_j(\bar{x}, \bar{y})^T \begin{pmatrix} x - \bar{x} \\ y - \bar{y} \end{pmatrix} \leq 0, \quad (2.4)$$

is valid for $\mathcal{C}$ for $(\bar{x}, \bar{y})$ with $g_j(\bar{x}, \bar{y}) \geq 0$.

More generally, Bonami et al. [23] showed the following.

Lemma 2.1 (Lemma 1, [23]). *If $\mathcal{C}$ is convex and $g_j(\bar{x}, \bar{y}) = 0$, then inequality (2.4) is valid for $\mathcal{C}$.*

Note that when $g_j(\bar{x}, \bar{y}) = 0$, (2.4) coincides with (2.1). Hence, in our linearization-based algorithms, to obtain an outer approximation of $\mathcal{C}$ we apply (2.4) at $(\bar{x}, \bar{y})$ for $j \in \hat{J}(\bar{x}, \bar{y}) := \{j \in J : g_j(\bar{x}, \bar{y}) = 0\}$. Thus, we define

$$S(\mathcal{K}) := \left\{ (x, y) \in X : \nabla g_j(\bar{x}, \bar{y})^T \begin{pmatrix} x - \bar{x} \\ y - \bar{y} \end{pmatrix} \leq 0, \forall j \in \hat{J}(\bar{x}, \bar{y}), (\bar{x}, \bar{y}) \in \mathcal{K} \right\}.$$

The second challenge arises when an integer solution $\hat{y}$ is obtained for which the problem $\text{NLP}(\hat{y})$ is not feasible (and so $\hat{y} \notin \text{proj}_y(\mathcal{C}) = \{y \in \mathbb{R}^q : \exists x \text{ with } (x, y) \in \mathcal{C}\}$), in which case the problem $\text{FNLP}(\hat{y})$ is typically solved for the case of convex MINLP. However, if some of the functions $g_j, j \in J$ are not convex, the objective function in $\text{FNLP}(\hat{y})$

might not be convex, and so an NLP solver may fail to find a global solution to $\text{FNLP}(\hat{y})$. Furthermore, the linearizations derived at the solution of $\text{FNLP}(\hat{y})$ may not be sufficient to obtain convergence of the algorithm, because they are not guaranteed to remove $\hat{y}$ from the current outer approximation when $\hat{y} \notin \text{proj}_y(\mathcal{C})$. Our two algorithms differ in the way that the second challenge is addressed, as described in the following two subsections.

2.3.1 Proj-Grad: The Projection-Based LP/NLP-BB Algorithm

The first algorithm, which we refer to as Proj-Grad, is a direct modification of the LP/NLP-BB algorithm, and hence follows a branch-and-bound procedure in which Algorithm 4 is called at every node in the branch-and-bound tree, with the exception that an alternative function is called in the case that $\hat{y} \in \mathbb{Z}^q$. In particular, when an integral solution $\hat{y}$ is obtained for which $\text{NLP}(\hat{y})$ is infeasible, this algorithm solves a feasibility problem different from $\text{FNLP}(\hat{y})$. This *projection* problem was also used in Bonami et al. [23], although we allow different choices in the norm to use in the objective, and more importantly, after solving the projection problem, we derive inequalities that cut off $\hat{y}$ in a different way than in [23].

Suppose that $(\hat{x}, \hat{y})$ is a solution of $\text{OA}(\mathcal{K})$ and $\text{NLP}(\hat{y})$ is infeasible. In this case, we solve

$$\begin{aligned} z_{\text{proj}}(\hat{y}) = \min_{x,y} \quad & \theta_p(y; \hat{y}) \\ \text{s.t.} \quad & (x, y) \in \mathcal{C}, \end{aligned} \tag{ProjNLP}(\hat{y})$$

where $\theta_p(y; \hat{y}) = \|y - \hat{y}\|_p^p$, for $p \in \{1, 2\}$. Bonami et al. [23] proposed solving this problem with $p = 2$. $\text{ProjNLP}(\hat{y})$ finds the closest point in the set $\text{proj}_y(\mathcal{C})$ to the point $\hat{y}$. The optimal value of this problem, $z_{\text{proj}}(\hat{y})$, must be strictly positive because $\hat{y} \notin \text{proj}_y(\mathcal{C})$.

$\text{ProjNLP}(\hat{y})$ is the problem of projecting the point $\hat{y}$ onto the set $\text{proj}_y(\mathcal{C})$, and hence, we can use the optimality condition (1.4). In the case $p = 2$, $\theta_2(y; \hat{y})$ is differentiable, and hence $\hat{s} = (\bar{y} - \hat{y})$ is the unique gradient of $\theta_2(y; \hat{y})$ at $\bar{y}$, and so the inequality

$$(\bar{y} - \hat{y})^T (y - \bar{y}) \geq 0, \tag{2.5}$$

is a valid inequality for $\text{proj}_y(\mathcal{C})$. This inequality can eliminate any point $\hat{y} \notin \text{proj}_y(\mathcal{C})$ since by plugging in $\hat{y}$, the left-hand side would be $-\theta_2(\bar{y}; \hat{y})^2$ which is strictly negative. Bonami et al. [23] use (2.5) to eliminate solutions $\hat{y} \notin \text{proj}_y(\mathcal{C})$. We refer to that algorithm as Proj-Obj (it is a **projection**-based algorithm that uses the the **objective** cut (2.5) to eliminate infeasible integer solutions.) Our proposed algorithm, Proj-Grad, is based instead on the following lemma.

Lemma 2.2. *Suppose $NLP(\hat{y})$ is infeasible for a given $\hat{y}$, i.e., $\hat{y} \notin \text{proj}_y(\mathcal{C})$. Let $(\bar{x}, \bar{y})$ be an optimal solution of $\text{ProjNLP}(\hat{y})$ and assume a constraint qualification holds at $(\bar{x}, \bar{y})$. Then, for all $(x, \hat{y}) \in X$, there exists $j \in \hat{J}(\bar{x}, \bar{y})$ such that*

$$\nabla g_j(\bar{x}, \bar{y})^T \begin{pmatrix} x - \bar{x} \\ \hat{y} - \bar{y} \end{pmatrix} > 0.$$

Proof. By optimality of $(\bar{x}, \bar{y})$ to $\text{ProjNLP}(\hat{y})$ and because a constraint qualification holds at $(\bar{x}, \bar{y})$, there exists $\tilde{s} \in \partial\theta(\bar{y}; \hat{y})$ such that

$$-\tilde{s} = \sum_{j \in \hat{J}(\bar{x}, \bar{y})} \lambda_j \nabla g_j(\bar{x}, \bar{y}),$$

where $\lambda_j \geq 0, j \in \hat{J}(\bar{x}, \bar{y})$ are the corresponding optimal Lagrange multipliers.

Suppose for contradiction that there exists $(x, \hat{y}) \in X$ that satisfies (2.4) for all $j \in \hat{J}(\bar{x}, \bar{y})$. If we then multiply (2.4) by λ_j for $j \in \hat{J}(\bar{x}, \bar{y})$ and add them we obtain

$$0 \geq \sum_{j \in \hat{J}} \lambda_j \nabla g_j(\bar{x}, \bar{y})^T \begin{pmatrix} x - \bar{x} \\ \hat{y} - \bar{y} \end{pmatrix} = -\tilde{s}^T \begin{pmatrix} x - \bar{x} \\ \hat{y} - \bar{y} \end{pmatrix} = \theta_p(\bar{y}, \hat{y}). \quad (2.6)$$

Here, the last equation follows for $p = 2$, because in that case $\tilde{s} = (0, \bar{y} - \hat{y})$. The last equation follows for $p = 1$, because in that case

$$-\tilde{s}^T \begin{pmatrix} x - \bar{x} \\ \hat{y} - \bar{y} \end{pmatrix} = \sum_{i=1}^n 0 \cdot (x_i - \bar{x}_i) + \sum_{i: \bar{y}_i < \hat{y}_i} (\hat{y}_i - \bar{y}_i) - \sum_{i: \bar{y}_i > \hat{y}_i} (\hat{y}_i - \bar{y}_i) = \|\hat{y} - \bar{y}\|_1.$$

But, $\theta_p(\bar{y}, \hat{y}) > 0$ because $\hat{y} \notin \text{proj}_y(\mathcal{C})$, and hence (2.6) is a contradiction. $\square$

Lemma 2.2 implies that if $\hat{y} \notin \text{proj}_y(\mathcal{C})$ and $\text{ProjNLP}(\hat{y})$ is solved with either the 1-norm or squared 2-norm as the objective, the solution $\hat{y}$ can be eliminated by adding inequalities (2.4) derived from an optimal solution $(\bar{x}, \bar{y})$ of $\text{ProjNLP}(\hat{y})$. Note that, if we use the $p = 1$ in the objective of the projection problem $\text{ProjNLP}(\hat{y})$, there might be multiple subgradients of $\theta(y; \hat{y})$ at $\bar{y}$, and it is only guaranteed that there *exist* a subgradient for which inequality (1.4) is valid; (1.4) may not be valid for an arbitrary subgradient at $\bar{y}$. Thus, Lemma 2.2 provides flexibility in the choice of the norm used for the objective in $\text{ProjNLP}(\hat{y})$. In Section 2.4 we compare the computational performance of Proj-Grad, which uses (2.4), to Proj-Obj, which uses (2.5), in the case when the squared 2-norm is used in $\text{ProjNLP}(\hat{y})$. We find that the performance is very similar in a little more than half of the test instances, but that in the remainder Proj-Grad is preferable.

Based on Lemma 2.2, we use inequalities (2.4) to eliminate an integer solution $\hat{y} \notin \text{proj}_y(\mathcal{C})$. Therefore, given a set of points $\mathcal{K}$ the outer approximating MILP is

$$\begin{aligned} z_{\text{OA}} = \min_{x,y} \quad & c^T x + d^T y \\ \text{s.t.} \quad & (x, y) \in S(\mathcal{K}) \\ & y \in \mathbb{Z}^q. \end{aligned} \quad (\text{ProjOA}_p(\mathcal{K}))$$

Once $\text{ProjNLP}(\hat{y})$ is solved and its optimal solution $(\bar{x}, \bar{y})$ is obtained, we only need to update the sets $\mathcal{K}$ and hence $S(\mathcal{K})$.

In the Proj-Grad algorithm, the steps followed in a node of the branch-and-bound tree are the same as presented in Algorithm 4, with two exceptions. First, at a node with bounds $\bar{L}$ and $\bar{U}$ on y and the current outer-approximation set $\mathcal{K}$, the LP relaxation of $\text{ProjOA}_p(\mathcal{K})$,

$$\begin{aligned} z_{\text{LP}}(\mathcal{K}) = \min_{x,y} \quad & c^T x + d^T y \\ \text{s.t.} \quad & (x, y) \in S(\mathcal{K}) \\ & (x, y) \in X, y \in [\bar{L}, \bar{U}], \end{aligned} \quad (\text{LP}(\mathcal{K}, \bar{L}, \bar{U}))$$

is solved in place of $\text{LP}_C(\mathcal{K}, \bar{L}, \bar{U})$, reflecting the use of linearization inequalities (2.4) in place of (2.3). Second, when a solution $\hat{y} \in \mathbb{Z}^q$ is obtained and the node is not pruned

by bound, the algorithm calls the function $\text{ProcessIntegerGeneral}(\mathcal{K}, z_u, \hat{y})$, which is described in Algorithm 6, in place of the function $\text{ProcessIntegerConvex}(\mathcal{K}, z_u, \hat{y})$. Thus, when the optimal solution of $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$, $(\hat{x}, \hat{y})$, has $\hat{y} \in \mathbb{Z}^q$, $\text{NLP}(\hat{y})$ is solved and, if feasible, $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is updated with linearization inequalities (2.4) from the solution of $\text{NLP}(\hat{y})$. If $\text{NLP}(\hat{y})$ is infeasible, then $\text{ProjNLP}(\hat{y})$ is solved and $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is updated with linearization inequalities (2.4) from the solution of $\text{ProjNLP}(\hat{y})$. Then $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is re-solved and this process continues until either the node is pruned or $\hat{y} \notin \mathbb{Z}^q$, after which branching is done to create two new child nodes.

Algorithm: $\text{ProcessIntegerGeneral}(\mathcal{K}, z_u, \hat{y})$

Solve $\text{NLP}(\hat{y})$.

if $\text{NLP}(\hat{y})$ is feasible **then**

 Denote $(\bar{x}, \hat{y})$ as the optimal solution.

 Update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\bar{x}, \hat{y})\}$ and accordingly $S(\mathcal{K})$ by new linearizations (2.4).

 Set $z_u = \min\{z_u, z_{\text{NLP}(\hat{y})}\}$.

else

 Solve $\text{ProjNLP}(\hat{y})$ and let $(\bar{x}, \bar{y})$ be an optimal solution.

 Update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\bar{x}, \bar{y})\}$ and accordingly $S(\mathcal{K})$ by new linearizations (2.4).

Algorithm 6: Processing integer solutions obtained in the Proj-Grad branch-and-bound algorithm for MINLP with convex continuous relaxation.

Theorem 2.3. *Suppose that $\mathcal{C}$ is convex, the integer variables y are bounded, and a constraint qualification holds at the solution of all NLP subproblems including $\text{NLP}(\hat{y})$ and $\text{ProjNLP}(\hat{y})$. Then, the Proj-Grad algorithm terminates finitely with the optimal solution of (MINLP).*

Proof. We first observe that boundedness of the integer variables y ensures that only finitely many nodes will be explored in the branch-and-bound tree. We next argue that

Algorithm 4 is finite at every node. The main loop in this algorithm only repeats if the node is not pruned and $\hat{y} \in \mathbb{Z}^q$. We argue this can happen only finitely many times. First, if $\text{NLP}(\hat{y})$ is infeasible, i.e., $\hat{y} \notin \text{proj}_y(\mathcal{C})$, Lemma 2.2 showed that the inequalities (2.4) prevent $\hat{y}$ from being encountered again as a solution. Therefore, no solution $y \notin \text{proj}_y(\mathcal{C})$ will be encountered by the algorithm more than once. On the other hand, if $\text{NLP}(\hat{y})$ is feasible, we next show that, having added inequalities (2.4) based on the solution of $\text{NLP}(\hat{y})$, if we obtain the same solution again at a node, the node will be pruned by bound. Since there are only finitely many integer solutions, this demonstrates that the processing of each node terminates finitely.

Therefore, suppose $(\hat{x}, \hat{y})$ is the solution of $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$, $\hat{y} \in \mathbb{Z}^q$, and $\text{NLP}(\hat{y})$ is feasible with an optimal solution $(\bar{x}, \hat{y})$. The algorithm updates $\mathcal{K}$ with $(\bar{x}, \hat{y})$ and $S(\mathcal{K})$ accordingly, and then re-solves $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$. Now, suppose, that at some point while processing this same node, $(x', \hat{y})$ is obtained as the optimal solution of $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$. This solution must satisfy the following constraints

$$\nabla g_j(\bar{x}, \hat{y})^T \begin{pmatrix} x - \bar{x} \\ y - \hat{y} \end{pmatrix} \leq 0 \quad \forall j \in \hat{J}(\bar{x}, \hat{y}) \quad (2.7)$$

which have been added to $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ at this node. Since a constraint qualification holds at the optimal solution $(\bar{x}, \hat{y})$ of $\text{NLP}(\hat{y})$, there exist Lagrange multipliers $\lambda_j \geq 0$, $j \in J$ that satisfy

$$- \begin{pmatrix} c \\ d \end{pmatrix} = \sum_{j \in \hat{J}(\bar{x}, \hat{y})} \lambda_j \nabla g_j(\bar{x}, \hat{y}). \quad (2.8)$$

Now, plugging in $(x', \hat{y})$ into (2.7) together with (2.8), one has

$$\begin{pmatrix} c \\ d \end{pmatrix} \begin{pmatrix} x' - \bar{x} \\ 0 \end{pmatrix} \geq 0,$$

which by adding $d^T \hat{y}$ to both sides yields $z_{\text{LP}} = c^T x' + d^T \hat{y} \geq c^T \bar{x} + d^T \hat{y} \geq z_u$, where the last inequality follows because the solution $(\bar{x}, \hat{y})$ was previously found as a feasible

solution to (MINLP). Thus, this node will be pruned by bound, completing the argument that the algorithm terminates finitely.

Correctness of the algorithm then follows by standard arguments for a branch-and-bound algorithm. First, every inequality added to the relaxation problem $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is valid by Lemma 2.1, and because the branch-and-bound tree only prunes nodes that are either infeasible or have a bound higher than a known feasible solution, there must exist an optimal solution that the algorithm does not exclude. In addition, the upper bound is only updated after a feasible solution has been found, and so at termination, this upper bound is the optimal value. $\square$

A key assumption of this algorithm is that the nonlinear programs $\text{NLP}(\hat{y})$ and $\text{ProjNLP}(\hat{y})$ are solved to global optimality. Bazaraa et al. [14, Theorem 4.2.16, pp 195] show that for an NLP with convex objective function and constraints defined by quasiconvex functions, a solution that satisfies the Karush-Kuhn-Tucker (KKT) conditions is a global optimal solution. Therefore, in quasiconvex MINLPs, a global optimal solution of $\text{NLP}(\hat{y})$ and $\text{ProjNLP}(\hat{y})$ can be obtained by a nonlinear programming solver that finds a solution that satisfies the KKT conditions. More generally, our algorithm makes an explicit assumption that both $\text{NLP}(\hat{y})$ and $\text{ProjNLP}(\hat{y})$ for MINLPs with convex continuous relaxation are solved to global optimality.

2.3.2 Hybrid-BB: A Hybrid Between LP/NLP and NLP-based Branch-and-Bound

Our next algorithm, Hybrid-BB, is a hybrid between LP/NLP-BB and NLP-based Branch-and-Bound (NLP-BB for short). NLP-BB is a branch-and-bound algorithm in which the nonlinear programming relaxation, $\text{NLP}[\bar{L}, \bar{U}]$, is solved at every node in the branch-and-bound tree. The algorithm has been implemented in MINLP-BB [90], SBB [33], Bonmin [22] and Minotaur [92].

Algorithm: ProcessNodeHybrid($\bar{L}, \bar{U}$)

Solve LP($\mathcal{K}, \bar{L}, \bar{U}$).

if LP($\mathcal{K}, \bar{L}, \bar{U}$) is infeasible or $z_{\text{LP}}(\mathcal{K}) \geq z_u$ **then**

 Prune the node.

else

 Let $(\hat{x}, \hat{y})$ be an optimal solution of LP($\mathcal{K}, \bar{L}, \bar{U}$).

if $\hat{y} \in \mathbb{Z}^q$ and $(\hat{x}, \hat{y}) \in \mathcal{C}$ **then**

 Update $z_u = z_{\text{LP}}(\mathcal{K})$.

 Prune the node.

else if $\hat{y} \in \mathbb{Z}^q$ **then**

 Solve NLP[$\bar{L}, \bar{U}$].

 Let $(\bar{x}, \bar{y})$ denote the solution.

if NLP[$\bar{L}, \bar{U}$] is infeasible or $z_{\text{NLP}}[\bar{L}, \bar{U}] \geq z_u$ **then**

 Prune the node.

else

 Update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\bar{x}, \bar{y})\}$ and accordingly $S(\mathcal{K})$ by new linearizations (2.4).

if $\bar{y} \in \mathbb{Z}^q$ **then**

 Update $z_u = \min\{z_u, z_{\text{NLP}}[\bar{L}, \bar{U}]\}$.

 Prune the node.

 Solve LP($\mathcal{K}, \bar{L}, \bar{U}$) and let (x', y') be the optimal solution.

if $y' \in \mathbb{Z}^q$ and $(x', y') \in \mathcal{C}$ **then**

 Update $z_u = z_{\text{LP}}(\mathcal{K})$.

 Prune the node.

else if $y' \notin \mathbb{Z}$ **then**

 Branch on a variable y_i with $y'_i \notin \mathbb{Z}$ to create two new subproblems.

else

 Branch on a variable y_i with $\bar{y}_i \notin \mathbb{Z}$ to create two new subproblems.

else

 Branch on a variable y_i with $\hat{y}_i \notin \mathbb{Z}$ to create two new subproblems.

Algorithm 7: Processing a node in the Hybrid-BB branch-and-bound algorithm.

Algorithm 7 describes the steps taken by Hybrid-BB to process a node in the branch-and-bound tree. The main idea of Hybrid-BB is to avoid the need to solve a separate feasibility problem by solving the NLP relaxation, $\text{NLP}[\bar{L}, \bar{U}]$, when $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ yields an integer feasible solution, rather than solving an NLP with the integer variables fixed and then possibly a feasibility problem. (We refer to this algorithm as a hybrid algorithm because this step, solving $\text{NLP}[\bar{L}, \bar{U}]$, is what NLP-BB does at every node.) If $\text{NLP}[\bar{L}, \bar{U}]$ is infeasible, the node is pruned. Otherwise, linearizations (2.4) are added to the LP relaxation $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$. The node can then be pruned if the bound from $\text{NLP}[\bar{L}, \bar{U}]$ is greater than or equal to the current upper bound z_u . Otherwise, let $(\bar{x}, \bar{y})$ be an optimal solution of $\text{NLP}[\bar{L}, \bar{U}]$. If $\bar{y}$ is integer, the algorithm updates the best known solution and prunes the node. If the node is not pruned, then $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is solved again to obtain a (possibly) new LP optimal solution (x', y') . If y' is integral and $(x', y') \in \mathcal{C}$ then the current best solution is updated and the node is pruned. Otherwise, if y' is fractional, branching is done on a variable in which y'_i is fractional. Finally, if y' is integral but $(x', y') \notin \mathcal{C}$, then branching is done on a variable in which $\bar{y}_i$ is fractional (recall that $(\bar{x}, \bar{y})$ is the solution of $\text{NLP}[\bar{L}, \bar{U}]$ at this node). Note that, in this case, the algorithm branches even though y' , the solution to the current LP relaxation, is integral. The following example shows that this step may be necessary.

Example 2.4. Consider the MINLP:

$$\min 5x - 3y \tag{2.9a}$$

$$\text{s.t. } 4x^2 - 35x + 9y \leq -31 \tag{2.9b}$$

$$x \geq 1, y \geq 1, y \in \mathbb{Z}. \tag{2.9c}$$

At the root node, the algorithm solves the continuous relaxation of (2.9) and obtains $(\bar{x}, \bar{y}) = (5/2, 7/2)$ as the optimal solution. Since $(\bar{x}, \bar{y})$ is not integral, the algorithm adds linearization of (2.9b) taken at $(\bar{x}, \bar{y})$,

$$-5x + 3y \leq -2,$$

to the LP relaxation of (2.9), and then solves

$$\begin{aligned} \min \quad & 5x - 3y \\ \text{s.t.} \quad & -5x + 3y \leq -2, \\ & x \geq 1, y \geq 1, \end{aligned}$$

which yields $(x', y') = (1, 1)$. This solution is integral, but violates the nonlinear constraint (2.9b), and so branching must be done on the NLP relaxation solution $\bar{y}$. The first node is created with a bound $y \leq 3$. In this node, the LP relaxation solution $(1, 1)$ remains feasible, but the NLP relaxation solution changes to $(\bar{x}, \bar{y}) \approx (2.221, 3)$ which is integer feasible, and hence the best known objective value is updated to 2.104 and this node is pruned. The other node is created with a new bound $y \geq 4$. The LP relaxation solution of this node is $(\hat{x}, \hat{y}) = (2.8, 4)$ with objective value 2. Because $\hat{y} \in \mathbb{Z}$, the NLP relaxation at this node is again solved yielding solution $(\bar{x}, \bar{y}) \approx (2.829, 4)$ with objective value 2.144 and so this node is pruned by bound and the algorithm terminates.

Correctness and finiteness of the Hybrid-BB branch-and-bound algorithm follows directly from correctness and finiteness of NLP-BB, and the fact that $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is a valid relaxation of the feasible region at each node. Thus, it is valid to prune a node when $\text{LP}(\mathcal{K}, \bar{L}, \bar{U})$ is infeasible or provides a bound larger than the best-known objective value z_u . Other than that change, the only other difference between Hybrid-BB and NLP-BB is in the way branching decisions are made, which does not affect correctness of the algorithm. The only assumptions required for Hybrid-BB are that the integer variables are bounded, the feasible region $\mathcal{C}$ is convex, and the NLP relaxations are solved to global optimality.

Mahajan et al. [91] proposed an algorithm for convex MINLPs, QP-Diving, which is very similar to Hybrid-BB, in that the algorithm first solves an inexpensive approximation of the NLP subproblem at a node, and then only solves the NLP relaxation when an integer

feasible solution is obtained. The main difference is that the inexpensive approximation solved in QP-Diving is a quadratic program, whereas we solve only the linear program relaxation defined by the current outer approximation. The other difference is that we add linearization inequalities (2.4), so the algorithm applies to MINLPs with convex continuous relaxation, whereas QP-Diving adds (2.3), so that, as stated, QP-Diving requires the functions $g_j, j \in J$ to be convex. Of course, if QP-Diving is slightly modified to use (2.4) instead of (2.3), then this assumption could be relaxed to match our assumption that $\mathcal{C}$ is convex.

Bonami et al. [22] also propose a method that is a hybrid between LP/NLP-BB and NLP-BB. The main difference is the choice of when an NLP relaxation will be solved. In [22], the choice is allowed to be arbitrary. For the computational experiments, they solve the NLP relaxation every ten nodes. Our proposed hybrid algorithm suggests only to solve the NLP relaxation when the LP relaxation is integer feasible, and in that case it is *mandatory*. As a result of this choice, and the choice of linearizations used in the algorithm, our algorithm is able to solve any MINLP with convex continuous relaxation, whereas this is not guaranteed by the algorithm proposed in [22].

2.4 Computational Results

2.4.1 Implementation details

We implemented our algorithms in MINOTAUR[92], an open-source toolkit for solving MINLPs. MINOTAUR includes implementations of NLP-BB and a simple version of LP/NLP-BB for convex MINLPs. We used this implementation as a starting point for our linearization-based algorithms.

We performed our computational experiments on a six-core 2.66 GHz Intel Xeon with 128 GB RAM running RedHat Linux Enterprise Server 6.1. In our experiments, we used CPLEX 12.2 [44] as the LP solver, and, unless otherwise specified, filterSQP [54] as

the NLP solver within MINOTAUR. MINOTAUR is equipped with a presolve procedure which accelerates convergence of NLP-BB, but is not implemented for LP/NLP-BB. To make a consistent comparison between these algorithms, we turned the presolve off for NLP-BB by applying the option `---presolve off`. We use a time limit of one hour for every instance and algorithm.

2.4.2 Test instances

We used convex instances from the MacMINLP collection [89], the GAMS MINLP World [34] and the collection on the website of the IBM-CMU research group [38]. The test set consists of 271 total instances, which we categorize into 20 instance classes including multi-product batch plant design problems BatchS-H and BatchS-M [101, 114], constrained layout problems CLay-H and CLay-M [105], farmland layout problems FLay-H and FLay-M [105], block layout design problems fo-m-no-o [37], network design problems with congestion constraints nd [26, 16, 27], GAMS Development Client Models netmod [34], retrofit planning RSyn-H and RSyn-M [105], safety layout problems SLay-H and SLay-M [105], stochastic service system design problems sssd [51], synthesis design problem Syn-H and Syn-M [49, 113], cutting stock problems tls-trimloss [72], separable quadratic uncapacitated facility location problems uflquad [66], the perspective reformulation of the uflquad instances uflquad-pers [67], and source inversion of drinking water networks Water [87]. We create one additional category called other for the remaining instances.

The uflquad-pers test instances from Günlük and Linderoth [67] are derived by a reformulation of a problem containing constraints of the form:

$$x_{ij}^2 - y_{ij} \leq 0, \quad 0 \leq y_{ij} \leq z_i$$

where y_{ij} and x_{ij} are continuous variables and z_i is a binary variable. They strengthen the first of the above constraints with the following constraint,

$$x_{ij}^2 - y_{ij}z_i \leq 0 \quad (2.10)$$

which is defined by a nonconvex function, but defines a convex feasible region when intersected with the other constraints. They find that this reformulation yields a dramatic improvement in solution times when solved using NLP-BB. The constraint (2.10) can be reformulated as a second-order cone constraint. However, Günlük and Linderoth [68] point out that using (2.10) improves the performance of NLP solvers on average nearly seven times. For these instances, we use IPOPT as the NLP solver within all the algorithms because we found that filterSQP occasionally produced incorrect results when solving the NLP relaxation.

We also created another collection of quasiconvex MINLP instances in the following manner. Some of the instances `fo-m-no-o` [37] have constraints of the form $f(x, y) = \frac{a}{x} - y \leq 0$ where x and y are decision variables and $a \in \mathbb{R}_+$. For these instances, we reformulated the set $U = \{(x, y) \in \mathbb{R}_+^2 : \frac{a}{x} - y \leq 0\}$ as $\{(x, y) \in \mathbb{R}_+^2 : -xy + a \leq 0\}$, which uses the quasiconvex function $\tilde{f}(x, y) = -xy + a$. Of course, this reformulation is likely the opposite of what one would typically want to do in practice (one would prefer to reformulate quasiconvex constraints as convex constraints). Our only purpose in doing this reformulation is to obtain quasiconvex MINLP test instances. One may consider these instances as simulating formulations that may be written by an inexperienced modeler. We denote the quasiconvex formulation of these instances by `fo-m-no-o-qc`.

Table 2.1 displays statistics about the instance classes including the total number of instances, whether the objective function is nonlinear, whether the instances contain quasiconvex constraints, and the range of total number of variables, number of integer variables, total number of constraints and number of nonlinear constraints for the instances in each class.

Instance Class	Total inst	NL Obj	QC inst	Vars	Int vars	Cons	NL cons
BatchS	10	✓		[132,558]	[53,251]	[371,2327]	[1,1]
CLay-H	6			[90,275]	[18,55]	[132,395]	[24,60]
CLay-M	6			[30,85]	[18,55]	[54,155]	[24,60]
FLay-H	5			[46,566]	[4,60]	[51,693]	[2,6]
FLay-M	5			[14,86]	[4,60]	[11,93]	[2,6]
fo-m-no-o	39			[26,182]	[6,72]	[43,435]	[6,18]
fo-m-no-o-qc	29		✓	[112,180]	[42,72]	[204,426]	[7,9]
nd	5			[338,816]	[26,48]	[225,417]	[26,48]
netmod	4	✓		[456,1998]	[136,462]	[666,3137]	[0,0]
RSyn-H	24			[700,2720]	[148,576]	[1045,4980]	[6,112]
RSyn-M	24			[170,1440]	[69,576]	[286,3728]	[3,112]
SLay-H	7	✓		[140,1010]	[24,180]	[174,1305]	[0,0]
SLay-M	7	✓		[44,290]	[24,180]	[54,405]	[0,0]
sssd	14			[60,243]	[44,207]	[40,92]	[12,27]
Syn-H	24			[42,1528]	[5,320]	[58,2904]	[3,112]
Syn-M	24			[20,840]	[5,320]	[28,2104]	[3,112]
tls-trimloss	12			[37,812]	[31,668]	[24,384]	[2,12]
uflquad	10	✓		[650,3240]	[15,40]	[650,3280]	[0,0]
uflquad-pers	10		✓	[1275,6440]	[15,40]	[1275,6480]	[625,3200]
Water	4	✓		[195,107222]	[7,14]	[283,108217]	[0,0]
other	13	✓		[6,480]	[3,480]	[6, 462]	[0,40]

Table 2.1 Instance classes; total instances in each class, nonlinear objective or not, quasiconvex instance or not, the range [min,max] of number of variables, integer variables, constraints and nonlinear constraints in each class.

2.4.3 Results

We conduct three sets of experiments. In the first experiment, we examine the performance of two variants of projection-based LP/NLP-BB algorithm presented in Section 2.3.1. The first, Proj-Obj, uses the inequality (2.5) to eliminate integral solutions $\hat{y} \notin \text{proj}_y(\mathcal{C})$, and the second, Proj-Grad, uses (2.4). We then compare the LP/NLP-BB algorithm for convex instances with Proj-Grad. Finally, we compare three algorithms that can solve MINLPs with convex continuous relaxations: NLP-BB, Hybrid-BB, and Proj-Grad.

We first discuss the comparison of Proj-Obj and Proj-Grad. Proj-Obj was able to solve 215 of the instances within the time limit, whereas Proj-Grad was able to solve 222 instances. For the 215 instances solved by both methods, the geometric mean of solution time using Proj-Obj was 20.9 seconds, compared to 18.3 seconds for Proj-Grad. Thus, when averaged over all instances, we observe that Proj-Grad is able to solve slightly more instances in the time limit, and to solve them about 12% faster. However, we also observed there are a significant number of instances in which the choice of Proj-Obj vs. Proj-Grad has little impact. We therefore look more closely at the instances with more than 5% difference in computation time, which yields 102 of the 215 instances solved by both. Figure 2.2 displays a scatter plot of the CPU time for the two methods on all instances in this set. From this figure, we can see that in some cases Proj-Grad performs significantly better than Proj-Obj (note that Figure 2.2 uses a log scale), whereas the reverse is never true. The geometric mean of the solution time of these 102 instances was 48.6 seconds for Proj-Obj, compared to 32.5 for Proj-Grad. Thus, Proj-Grad reduced solution time on these instances by about 33% on average.

In the next experiment, we compare the performance of LP/NLP-BB and Proj-Grad. In this experiment, we consider only convex MINLP instances, since LP/NLP-BB only guarantees optimality for this case. Figure 2.3 shows the scatter plot of CPU time for

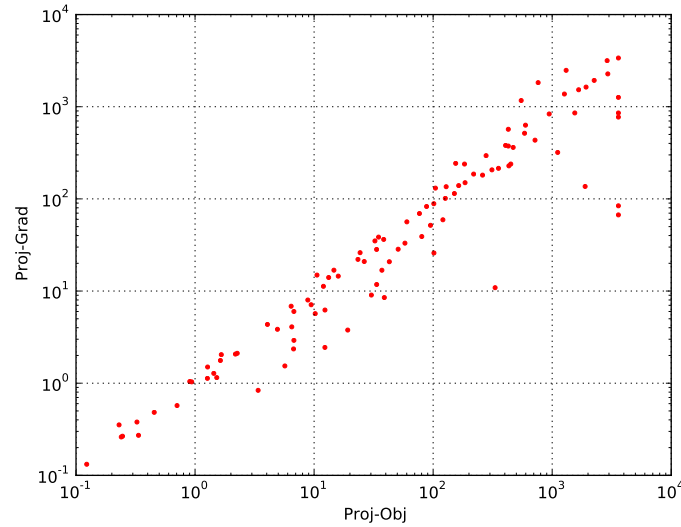


Figure 2.2 Scatter plot of CPU time in seconds for Proj-Obj vs. Proj-Grad.

Proj-Grad and LP/NLP-BB over all 240 instances. The plotted points are almost all very close to the 45-degree line, which shows that the two algorithms have comparable computational performance. Quantitatively, LP/NLP-BB solved 190 instances within the time limit, whereas Proj-Grad solved 191. Among the 189 instances that both algorithms solved within time limit, the geometric mean of solution time was 13.8 seconds for LP/NLP-BB and 14.4 seconds for Proj-Grad. There are 240 instances plotted in Figure 2.3 for which the geometric mean solution times for LP/NLP-BB and Proj-Grad are 46.5 and 48.2 seconds, respectively. (For these means, 3600 seconds is used as the solution time for an instance that hit the time limit.) Therefore, even though Proj-Grad has the additional ability to solve MINLPs with convex continuous relaxation, the performance of Proj-Grad is comparable to LP/NLP-BB on convex MINLP instances.

We also applied LP/NLP-BB to the instances `fo-m-no-o-qc`. These instances have constraints that are defined by quasiconvex functions, and hence LP/NLP-BB is not guaranteed to find the global optimal solution. Indeed, we observed three instances for which LP/NLP-BB converged to a solution with a significant gap from the optimal solution:

fo7_ar2_1_qc with 0.05% gap, fo7_qc with 1.08% gap and fo9_ar4_1_qc with 3.9% gap.

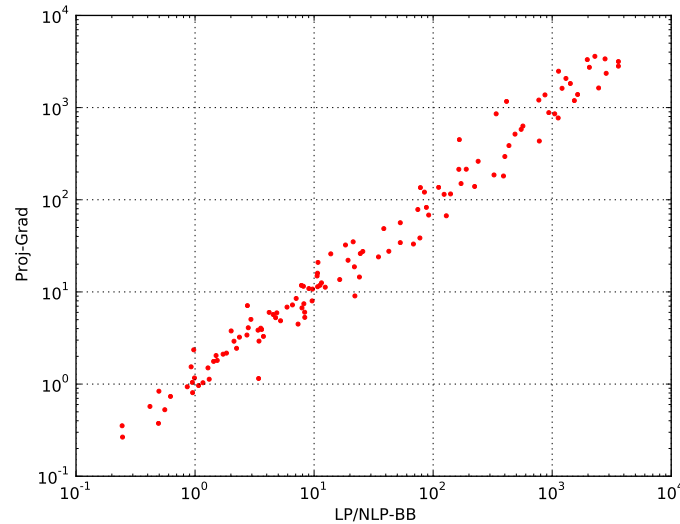


Figure 2.3 Scatter plot of CPU time in seconds of Proj-Grad and LP/NLP-BB on convex instances.

In our final experiment, we compare the performance of three algorithms, NLP-BB, Hybrid-BB and Proj-Grad which are able to solve MINLPs with convex continuous relaxation. First, of the 271 instances, there are three instances in which at least one of the three algorithms failed due to an NLP solver error. None of these instances were solved to optimality within the time limit by any of the algorithms. Table 2.2 presents the lower and upper bounds obtained by the algorithms within the time limit when applicable. These instances are excluded from all remaining results.

We use a performance profile, proposed by Dolan and Moré [48], to graphically present the results on the remaining instances. Let $\mathcal{P}$ be the set of problem instances and $\mathcal{S}$ be the set of algorithms we tested on these instances. For each $p \in \mathcal{P}$ and $s \in \mathcal{S}$ we evaluate the

	Lower bound			Upper bound		
	NLP-BB	Hybrid-BB	Proj-Grad	NLP-BB	Hybrid-BB	Proj-Grad
tls7	2.75	–	2.70	*	–	*
Water0202	61.16	–	–	125.19	–	–
Water0303	75.52	–	–	2632.33	–	–

Table 2.2 Results of instances on which any one of the three algorithms failed. A ‘–’ indicates an instance that failed due to an NLP error. A ‘*’ represents an instance for which no feasible solution was found within the time limit.

solution time $t_{p,s}$. The *performance ratio* is defined as

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,i} : i \in \mathcal{S}\}}.$$

Then for $s \in \mathcal{S}$, a distribution function $\rho_s(\tau)$ is defined as

$$\rho_s(\tau) = \frac{\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}}{|\mathcal{S}|}.$$

ρ_s is an increasing function which maps each $\tau \in \mathbb{R}$ onto $[0, 1]$.

The performance profile comparing NLP-BB, Hybrid-BB and Proj-Grad is shown in Figure 2.4. The figure shows that Proj-Grad is the dominant algorithm. NLP-BB has the slowest performance, which is likely a result of solving a nonlinear program at each node of the tree. The performance of Hybrid-BB is slightly worse than Proj-Grad.

Table 2.3 presents summary results of these three algorithms on each instance class. The table lists the number of instances in each instance class, the number of instances solved by each algorithm within the time limit, the number of instances solved by all three algorithms within the time limit, and the geometric mean of CPU time for each algorithm over the set of instances that were solved by all algorithms. From this table we observe that Proj-Grad and Hybrid-BB have similar performance on many classes of instances, with the

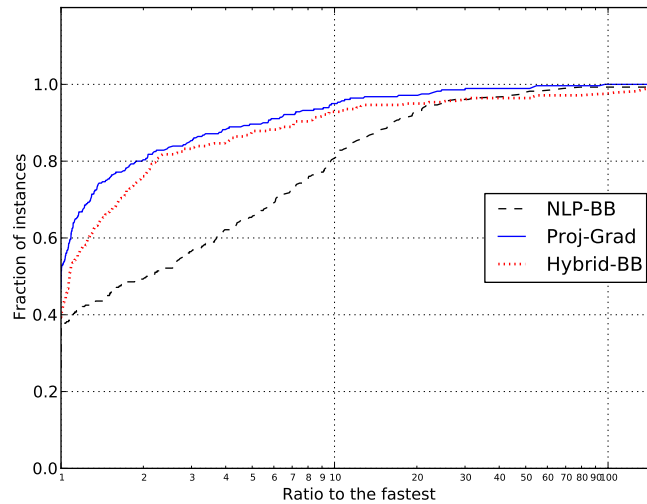


Figure 2.4 Performance profile of CPU time in seconds of Proj-Grad, Hybrid-BB and NLP-BB.

exceptions of CLay-H, CLay-M, netmod, SLay-H, SLay-M, uflquad, and uflquad-pers. Proj-Grad outperformed Hybrid-BB on all of these except uflquad and uflquad-pers. We also observed that the linearization-based algorithm Proj-Grad usually performed better than NLP-BB, with the exception of the FLayer-H, FLayer-M, netmod, SLay-H, SLay-M, uflquad, and Water instances. The only instance classes for which Hybrid-BB seemed best were nd, and perhaps fo-o-m-no-qc, for which Hybrid-BB solved one more instance than Proj-Grad.

Table 2.4 presents summary statistics for the instances for which at least one of the algorithms hit the time limit. For some of these instances, none of the algorithms found a feasible solution within the time limit, and so these are excluded from the averages. The number of such instances is indicated in the ‘# excluded’ column. The number of remaining instances is listed in the ‘# of instances’ column, and the geometric mean CPU time and arithmetic average remaining gap over these instances is reported for each

algorithm and instance class. For an instance not solved within the time limit, 3600 seconds is used as the time in the calculation of mean CPU time, and the remaining gap is calculated as $(UB - LB)/UB$ where UB and LB are the best upper and lower bounds, respectively, obtained by the algorithm in the one hour time limit. If an algorithm did not find a feasible solution (upper bound) within the time limit, then the gap is calculated using the worst upper bound obtained by any other algorithm. Cases where the average includes such gap calculations are marked with a ‘*’, in which case the number in parentheses represents the number of instances for which the algorithm did find an upper bound. A ‘–’ means that all the instances in that instance class were solved within time limit. The results on these more difficult instances reinforce the conclusions drawn from Table 2.3. The two linearization-based algorithms, Hybrid-BB and Proj-Grad, have similar performance, with Proj-Grad tending to perform slightly better. The linearization-based algorithms usually left smaller optimality gap than NLP-BB, with the exception of the SLay, uflquad, and other instances.

2.5 Conclusion

We studied linearization-based algorithms for MINLPs with convex continuous relaxation. This class of problems contains problems with constraints defined by convex functions, quasiconvex functions, or even sometimes nonconvex functions. We proposed two algorithms: the projection-based LP/NLP branch-and-bound (Proj-Grad) and Hybrid LP/NLP branch-and-bound (Hybrid-BB) algorithms. The Proj-Grad algorithm is a variant of the algorithm recently proposed by Bonami et al. [23] (which we called Proj-Obj), in which we use gradient linearizations at the solution of the projection problem to cut off an infeasible integer solution, rather than using an inequality derived from the objective function. We found the performance of Proj-Grad to be generally similar to Proj-Obj, but

significantly better on a portion of the instances. The Hybrid-BB algorithm is a hybrid between NLP-based branch-and-bound and the LP/NLP algorithm.

On convex MINLP instances, Proj-Grad had similar computational performance to the LP/NLP branch-and-bound algorithm that assumes the functions defining the constraints of the MINLP are convex. Thus, we found that even though Proj-Grad can solve a larger class of instances, there was little loss of computational performance when comparing these two algorithms.

The implementations of all the linearization-based algorithms we considered were deliberately kept basic for the purposes of comparisons in this paper. In particular, in these algorithms we add linearization inequalities only after solving a nonlinear program, which is the minimum required to obtain convergence of the algorithm. However, for convex MINLPs, it has been observed that adding additional linearization inequalities can lead to better performance [1]. It will be interesting to explore whether the performance of our proposed algorithms for solving MINLPs with convex continuous relaxation can be improved with similar strategies.

Instance class	# of instances	# solved by			# solved by all	GeoMean solved by all		
		NLP-BB	Hybrid-BB	Proj-Grad		NLP-BB	Hybrid-BB	Proj-Grad
Batch	10	10	10	10	10	31.8	14.8	14.2
CLay-H	6	6	6	6	6	53.7	25.8	10.0
CLay-M	6	6	6	6	6	14.2	28.7	7.4
FLay-H	5	4	4	4	4	7.2	10.3	12.6
FLay-M	5	4	4	4	4	2.0	13.9	12.6
fo-o-m-no	39	18	31	33	18	288.4	53.3	45.8
fo-o-m-no-qc	29	14	28	27	14	457.2	66.9	64.9
nd	5	4	5	5	4	239.4	16.6	20.2
netmod	4	3	3	3	3	55.9	184.8	119.9
RSyn-H	24	24	24	24	24	144.6	8.7	6.6
RSyn-M	24	6	9	9	6	315.3	35.0	38.4
SLay-H	7	7	6	6	6	1.4	18.1	10.5
SLay-M	7	7	6	7	6	0.3	32.7	4.6
sssd	14	4	7	7	4	15.1	16.0	18.0
tls-trimloss	12	2	4	4	2	0.3	2.6	1.9
Syn-H	24	24	24	24	24	1.5	0.9	0.9
Syn-M	24	17	19	19	17	5.6	2.7	2.9
uflquad	10	9	6	6	6	22.0	138.9	196.2
uflquad-pers	10	9	10	10	9	375.2	46.6	66.2
Water	4	2	2	2	2	2.0	4.5	4.1
other	14	14	11	12	11	0.7	1.9	1.1

Table 2.3 Summary results by instance class including: number of instances, number of instances solved by each algorithm within the time limit, number of instances solved by all algorithms and geometric mean of CPU time in seconds on the instances solved by all algorithms.

Instance class	# of instances	GeoMean of CPU time			% Gap			# excluded
		NLP-BB	Hybrid	Proj-Grad	NLP-BB	Hybrid	Proj-Grad	
FLay-H	1	3600.0	3600.0	3600.0	10.4	3.7	7.2	1
FLay-M	1	3600.0	3600.0	3600.0	3.6	8.0	4.9	
fo-o-m-no	21	3600.0	1424.7	1493.1	*(11) 22.6	*(20) 6.6	5.8	
fo-o-m-no-qc	14	2085.5	553.9	534.0	*(10) 16.0	–	0.1	
nd	1	3600.0	1718.5	1387.5	13.4	–	–	
netmod	1	3600.0	3600.0	3600.0	39.4	31.2	44.2	
RSyn-M	18	3600.0	2901.5	2971.4	279.8	92.1	92.7	
SLay-H	1	67.6	3600.0	3600.0	–	1.0	1.1	
SLay-M	1	27.2	3600.0	2643.0	–	3.1	–	
sssd	10	3600.0	3207.4	3401.5	0.2	0.1	0.1	
tls-trimloss	5	94.6	66.3	67.9	*(2) 50.2	39.7	35.6	4
Syn-M	7	3600.0	2961.0	2904.9	249.7	99.5	98.8	
uflquad	4	1771.8	3600.0	3600.0	7.3	55.4	56.4	
uflquad-pers	1	3600.0	228.2	451.3	23.4	–	–	
other	3	49.1	3600.0	1209.1	–	31.8	15.5	

Table 2.4 Summary results for instances not solved by all algorithms: number of instances not solved by all algorithms, geometric mean of CPU time in seconds and % gap on these instances for each algorithm. A ‘–’ indicates all the instances were solved within time limit. ‘*(n) g ’ indicates an upper bound was found for only n of the instances.

In this case, in the calculation of the average gap g , if the algorithm did not obtain an upper bound for an instance the upper bound obtained by a different algorithm is used. The column ‘# excluded’ indicates the number of instances for which none of the solvers found an upper bound. These instances are excluded from the all averages in this table.

Table 2.5: Detailed numerical results of all the instances

	CPU time in seconds					Number of nodes				
	NLP-BB	LP/NLP-C	Hybrid	Proj-Obj	Proj-Grad	NLP-BB	LP/NLP-C	Hybrid	Proj-Obj	Proj-Grad
Batch_Storage10_BM_10_10_6_4	37.0	53.0	62.8	60.1	56.3	1789	11655	14253	12551	11995
Batch_Storage_BM_10_10_6_4	43.7	69.3	63.4	67.3	66.9	2271	15339	13381	16013	15953
Batch_Storage_BM_8_6_6_4	1.7	3.5	3.5	3.5	3.5	131	423	397	423	423
Batch_Storage_BM_9_9_6_4	9.9	7.0	7.2	6.7	6.7	521	1979	2083	1889	1889
Batch_Storage_CH_8_6_6_4	7.9	5.4	5.2	5.6	5.5	183	497	519	497	497
Batch_Storage_CH_9_9_6_4	26.6	18.0	18.3	18.1	18.4	199	1919	2475	1915	1915
BatchS101006M	23.0	5.2	5.2	5.0	5.1	1605	1615	1651	1655	1655
BatchS121208M	142.0	10.7	11.7	11.6	11.4	9055	4513	4433	4043	4043
BatchS151208M	183.1	22.6	29.4	23.5	22.7	8267	7117	9775	7117	7117
BatchS201210M	313.7	25.7	29.5	27.3	27.5	7757	5381	5929	5389	5389
CLay0203H	3.4	2.8	7.2	6.5	4.1	353	361	457	475	315
CLay0203M	0.8	2.1	9.8	6.8	2.9	377	281	477	391	325
CLay0204H	22.1	3.4	4.8	1.5	1.1	1873	2029	2489	1123	1123
CLay0204M	6.1	3.4	16.2	4.9	3.8	3137	1489	2351	2233	1797
CLay0205H	304.7	21.3	32.2	32.4	35.0	13329	13579	22019	19865	23233
CLay0205M	86.5	7.1	15.7	38.9	8.5	22347	14837	20071	13603	14861
CLay0303H	6.7	4.2	18.7	6.8	6.0	837	903	1053	1127	679
CLay0303M	4.1	2.8	29.1	9.4	7.1	1463	1031	1369	1181	891

CLay0304H	416.1	13.8	270.8	101.5	25.9	28903	8305	22179	45277	15691
CLay0304M	72.9	7.8	215.6	33.5	11.8	17463	7439	26271	15427	4319
CLay0305H	374.7	77.4	52.4	34.8	38.5	14445	37355	25425	25445	29231
CLay0305M	68.4	10.8	35.7	26.4	20.9	13523	13909	32841	20481	48133
FLay02H	0.2	0.5	0.6	0.5	0.5	7	7	7	7	7
FLay02M	0.1	0.6	0.6	0.7	0.7	7	7	7	7	7
FLay03H	0.5	1.7	2.0	2.3	2.1	105	119	113	113	113
FLay03M	0.2	1.5	2.3	1.7	1.8	103	103	105	105	105
FLay04H	18.8	8.1	17.7	12.1	11.5	2321	2539	2727	2655	2655
FLay04M	3.9	10.7	32.7	16.1	16.0	2555	2551	2801	2557	2557
FLay05H	1738.3	1305.1	531.1	2005.8	2072.4	96513	103687	91317	113997	113997
FLay05M	125.6	772.5	790.4	1159.3	1207.1	80867	96795	98683	102139	102139
FLay06H	3600	3600	3600	3600	3600	166941	1000653	693081	883165	882389
FLay06M	3600	3600	3600	3600	3600	2504321	1104567	752077	1358785	1361469
fo7	2967.3	779.8	750.9	717.6	433.6	491965	912179	812235	608835	483847
fo7_2	1548.0	171.6	276.7	185.4	149.8	258295	170737	308543	183385	169791
fo7_ar25_1	679.4	78.2	241.9	128.7	135.5	105915	83955	224289	128627	127537
fo7_ar25_1_QC	1076.7	76.0	77.5	101.3	88.7	209377	82759	99587	110993	118349
fo7_ar2_1	954.1	127.1	102.8	104.6	130.9	159403	125297	99463	94811	113723
fo7_ar2_1_QC	1050.3	99.5	215.3	154.6	243.0	198493	101115	219627	149691	236655
fo7_ar3_1	1767.2	84.5	104.3	125.6	121.0	240197	91541	108489	130081	135447
fo7_ar3_1_QC	835.6	77.1	97.9	95.0	91.1	169547	85687	115007	100617	93273
fo7_ar4_1	1411.9	129.0	84.5	3600	67.0	185863	167361	96367	114577	76939
fo7_ar4_1_QC	986.5	73.3	68.9	76.7	69.4	202573	91485	74537	92359	75593

fo7_ar5_1	640.8	68.3	43.6	58.0	33.1	86099	73071	52153	73649	43143
fo7_ar5_1_QC	229.5	35.3	53.4	3600	84.1	46287	44307	71629	159643	129801
fo7_QC	3600	300.3	554.1	1112.6	319.2	746635	416885	657913	606739	419573
fo8	3600	1535.3	1699.7	-1.0	1191.0	466273	1512859	2192235	-1	1179715
fo8_ar25_1	3600	1420.9	1780.3	762.3	1825.5	452401	1184923	1870545	824797	1764603
fo8_ar25_1_QC	3600	3600	1192.0	958.4	935.2	593059	3347039	1296589	996069	1064129
fo8_ar2_1	3600	1131.1	2964.6	1309.9	2482.1	487029	761609	2019675	901681	1606999
fo8_ar3_1	3600	230.3	221.5	183.5	238.7	436535	313029	293325	246721	276239
fo8_ar3_1_QC	3600	187.2	112.4	111.0	112.4	609641	244351	177071	175353	175353
fo8_ar4_1	3600	223.2	140.9	163.9	139.4	373915	316779	182803	179351	149749
fo8_ar4_1_QC	3222.5	272.9	245.3	311.8	206.5	514073	364177	285093	404941	241847
fo8_ar5_1	3600	166.0	553.5	466.7	449.0	366395	202639	686557	581017	565967
fo8_ar5_1_QC	3600	506.8	273.5	428.0	569.2	485525	716393	251617	406005	611079
fo9	3600	3600	3600	-1.0	3600	284519	3992427	7556249	-1	6343703
fo9_ar25_1	3600	3600	3600	3600	3600	317833	3499175	3655927	3839813	3506531
fo9_ar2_1	3600	3600	3600	3600	3600	328649	3330469	3161189	3016545	3001477
fo9_ar2_1_QC	3600	3600	3600	3600	3600	334695	4330647	4040159	3031631	4454159
fo9_ar3_1	3600	413.6	523.0	549.2	1167.6	283891	524493	655091	629739	1526677
fo9_ar3_1_QC	3600	374.8	378.9	405.9	380.3	429637	543603	479043	495137	489407
fo9_ar4_1	3600	869.9	1181.5	1266.5	1373.4	252149	990713	1382481	1448289	1585651
fo9_ar4_1_QC	3600	1119.4	2092.7	3600	1263.3	412475	1198671	2485147	3214095	1409381
fo9_ar5_1	3600	3600	3600	2895.2	3161.3	234329	4015569	3331839	2230093	2978277
fo9_ar5_1_QC	3600	3600	2691.6	3600	3480.8	389681	3920515	3068589	2373039	3377239
geartrain	0.2	1.8	0.6	2.2	2.2	15	59	11	73	73

m3	0.3	1.0	0.6	0.8	0.8	49	45	47	47	45
m6	8.0	6.0	8.7	12.3	6.2	3363	9895	15539	11841	11967
m7	468.5	390.0	208.4	260.1	181.1	105713	475943	283649	321023	235547
m7_ar25_1	22.1	2.9	5.5	5.0	5.0	5807	2853	5187	5375	5375
m7_ar25_1_QC	17.7	3.5	3.4	3.4	3.4	5049	3685	3567	3667	3667
m7_ar2_1	100.1	19.3	23.4	23.4	22.1	25695	20345	26807	27363	25687
m7_ar2_1_QC	100.8	28.5	26.7	25.7	25.7	31571	29919	28685	26971	26971
m7_ar3_1	133.5	18.4	30.3	31.2	32.3	25751	16745	31039	28429	31275
m7_ar3_1_QC	123.6	23.7	25.6	33.5	28.3	24193	22695	26083	26637	28299
m7_ar4_1	51.1	22.0	16.3	30.3	9.0	10513	26053	15343	34259	10717
m7_ar4_1_QC	56.0	20.0	15.2	14.7	16.8	13531	27567	16547	15881	20819
m7_ar5_1	222.0	21.8	17.4	18.8	18.7	47401	22787	20145	22179	21833
m7_ar5_1_QC	308.4	34.5	19.3	14.3	14.3	68019	39401	21127	18641	18641
nd-10	13.8	2.0	2.5	19.2	3.8	417	539	583	687	645
nd-11	48.0	4.5	4.2	10.2	5.7	851	905	1125	1247	1123
nd-12	2712.7	123.7	110.6	-1.0	114.5	26977	22835	37361	-1	25225
nd-13	1825.9	91.7	65.8	-1.0	68.2	11253	18051	14455	-1	13333
nd-14	3600	1636.1	1718.5	-1.0	1387.5	21417	171505	196691	-1	176377
netmod.dol1	3600	3600	3600	3600	3600	547	406263	177639	390821	390839
netmod.dol2	872.2	2586.2	1346.7	2611.3	2603.8	185	71221	33793	74505	74505
netmod.kar1	14.5	26.2	68.3	25.8	25.7	309	2779	6265	2779	2779
netmod.kar2	13.9	25.9	68.7	25.7	25.8	309	2779	6265	2779	2779
no7_ar25_1	2885.1	324.6	231.9	218.9	185.9	478203	273781	239695	182931	174257
no7_ar25_1_QC	3600	222.9	422.2	355.9	352.4	717347	220489	404025	355565	355565

no7_ar2_1	1806.5	140.1	109.0	113.8	115.6	336591	122391	104529	112003	112003
no7_ar2_1_QC	2235.8	101.8	133.7	150.8	114.3	432669	130257	128815	142443	118273
no7_ar3_1	3600	487.4	511.9	586.2	515.4	619935	405963	444975	471217	417999
no7_ar3_1_QC	3600	438.0	399.2	470.3	361.8	739459	409073	368755	397789	332165
no7_ar4_1	3600	338.4	412.1	1548.5	855.8	506291	345601	421659	1305171	711461
no7_ar4_1_QC	2278.6	257.0	474.1	428.3	374.3	436065	278583	417999	376695	360671
no7_ar5_1	2407.8	189.7	204.0	354.0	214.5	326763	163311	181013	269981	190039
no7_ar5_1_QC	1302.6	182.6	565.5	383.3	390.7	229213	179287	499325	345289	352769
o7_2	3600	1972.4	3600	-1.0	3314.6	723581	1770279	3729469	-1	3168155
o7_ar25_1	3600	2457.4	1790.7	1925.9	1633.1	777553	1986039	1538253	1629407	1382299
o7_ar25_1_QC	3600	1617.5	1625.0	1669.2	1527.5	880789	1471967	1445807	1453889	1329501
o7_ar2_1	3600	567.6	664.9	596.9	630.7	737433	458595	568077	556175	512413
o7_ar2_1_QC	3600	669.4	870.0	943.9	834.6	816711	597289	803371	803809	731299
o7_ar3_1	3600	2780.7	2492.8	3600	3377.8	715267	2393603	2102525	2967595	2518613
o7_ar3_1_QC	3600	2245.5	2314.0	2933.2	2270.9	875349	2001069	2262661	2500073	1998115
o7_ar4_1	3600	3600	3600	-1.0	-1.0	674621	3232071	3227173	-1	-1
o7_ar4_1_QC	3600	3441.6	3373.9	3600	3600	896431	3223547	3094933	3483567	3444435
o7_ar5_1_QC	3600	2041.7	1758.6	2253.8	1930.7	864729	1771383	1724989	2027241	1652115
o8_ar4_1	3600	3600	3600	3600	3600	526603	3668819	4519407	3677135	3834709
o9_ar4_1	3600	3600	3600	3600	3600	389065	3701489	4651095	3248159	4111761
optprloc	0.3	0.5	0.6	3.4	0.8	25	149	185	273	171
Process8_convex	0.1	0.3	0.4	0.4	0.3	3	5	5	5	5
Process_20_bigM	1.0	1.0	1.0	1.2	1.2	421	459	441	447	447
RSyn0805H	3600	425.0	400.0	391.7	344.2	45315	86759	76447	75017	75017

RSyn0805M	22.6	3.6	3.5	3.9	3.9	3367	3497	3749	3659	3659
RSyn0805M02H	79.7	5.9	5.9	6.4	6.8	299	297	335	339	339
RSyn0805M02M	2442.0	164.4	165.6	212.0	214.0	76133	62239	65969	84249	84249
RSyn0805M03H	185.4	6.6	4.2	7.0	7.2	143	199	175	185	185
RSyn0805M03M	3600	1210.5	957.6	1626.9	1619.2	68923	318863	261247	465325	465325
RSyn0805M04H	122.1	7.7	6.1	7.6	7.6	123	139	119	119	119
RSyn0805M04M	3600	433.4	416.3	379.0	386.9	44883	86759	76447	75017	75017
RSyn0810H	3600	3600	3600	3600	3600	38905	1428249	1529029	1647879	1630369
RSyn0810M	70.3	10.6	11.6	10.6	14.9	13489	14075	14055	14055	14055
RSyn0810M02H	332.8	5.2	6.4	4.9	4.9	931	247	235	229	229
RSyn0810M02M	3600	2842.1	2410.8	2298.6	2354.7	158331	1122985	948609	918061	918061
RSyn0810M03H	850.1	9.7	13.3	10.7	10.7	1279	493	599	565	565
RSyn0810M03M	3600	3600	3600	3600	3600	77759	1982829	2089537	1983643	1976199
RSyn0810M04H	173.9	16.5	15.6	16.6	16.1	103	431	355	375	375
RSyn0810M04M	3600	3600	3600	3600	3600	38667	1225379	1373673	1376269	1448357
RSyn0815H	3600	3600	3600	3600	3600	35821	1672799	1651489	1677039	1647267
RSyn0815M	83.4	11.2	12.1	11.8	11.8	9609	10953	11793	10953	10953
RSyn0815M02H	45.5	8.2	11.8	7.6	7.4	73	363	463	355	355
RSyn0815M02M	3600	3600	3600	3600	3600	130831	2004259	1831343	1802181	1755079
RSyn0815M03H	2305.7	16.4	11.2	13.7	13.6	3267	431	401	547	547
RSyn0815M03M	3600	3600	3600	3600	3600	63325	1779179	1702421	1686829	1589575
RSyn0815M04H	1072.2	25.1	28.2	25.3	26.0	625	385	619	479	479
RSyn0815M04M	3600	3600	3600	3600	3600	32335	1446627	1389427	1355579	1357647
RSyn0820H	3600	3600	3600	3600	3600	30473	1555473	1446227	1502689	1530927

RSyn0820M	1453.3	87.7	90.1	88.1	82.7	131971	91155	93579	91155	91155
RSyn0820M02H	220.1	7.3	5.0	4.5	4.5	313	195	185	183	183
RSyn0820M02M	3600	3600	3600	3600	3600	114099	2938805	2868109	2888343	2890415
RSyn0820M03H	2809.1	11.5	26.7	12.5	12.5	2741	465	857	563	563
RSyn0820M03M	3600	3600	3600	3600	3600	59041	2193629	2286199	2176679	2186445
RSyn0820M04H	500.0	34.7	27.7	23.7	24.0	269	975	363	367	367
RSyn0820M04M	3600	3600	3600	3600	3600	29835	1445657	1444227	1315149	1322863
RSyn0830H	3600	3600	3600	3600	3600	14569	1224339	1236967	1235069	1229819
RSyn0830M	2088.2	238.8	245.3	264.6	261.5	152385	188577	196329	210649	210649
RSyn0830M02H	171.7	7.9	11.6	6.7	6.7	253	99	99	99	99
RSyn0830M02M	3600	3600	3600	3600	3600	69765	2261777	2223069	2396987	2174299
RSyn0830M03H	278.8	24.0	31.3	16.0	14.5	117	233	245	221	221
RSyn0830M03M	3600	3600	3600	3600	3600	27391	1336023	1410519	1458749	1416111
RSyn0830M04H	1677.8	53.0	78.7	34.4	34.2	793	841	1035	559	559
RSyn0830M04M	3600	3600	3600	3600	3600	13543	933451	1013123	969649	1030781
RSyn0840H	3600	3600	3600	3600	3600	10791	833731	844383	791127	762933
RSyn0840M	3600	3600	3600	3600	3600	421697	3482893	3465401	3464527	3466053
RSyn0840M02H	114.8	8.4	7.9	5.4	5.3	95	79	69	69	69
RSyn0840M02M	3600	3600	3600	3600	3600	57769	2046109	1864599	1748205	1819929
RSyn0840M03H	937.0	14.4	24.3	14.3	15.0	403	191	223	203	203
RSyn0840M03M	3600	3600	3600	3600	3600	24429	1251319	1410199	1438529	1453523
RSyn0840M04H	1427.6	42.4	44.6	27.2	27.6	571	511	529	487	487
RSyn0840M04M	3600	3600	3600	3600	3600	11337	1003949	978595	974823	997929
Safety3	65.8	3600	3600	3600	3600	36077	2720149	2294749	2766099	2782059

Safety_Layout_BM_4_2	0.9	0.5	1.3	0.5	0.5	579	265	501	327	327
Safety_Layout_BM_5_2	5.7	1.1	8.8	1.0	1.0	2511	795	2353	1767	1767
Safety_no_rotation_BM	4.5	9.0	469.1	331.6	10.9	1477	4287	56101	69139	6823
safety_no_rotation_CH	15.0	111.3	3600	1891.6	136.4	613	45331	155339	488313	57921
SafetyLayout4_Convex	18.8	2.2	5.0	12.3	2.4	1055	1015	1525	3333	723
SLay04H	0.3	0.8	2.2	0.8	0.8	183	121	201	121	121
SLay04M	0.1	1.2	3.4	1.2	1.2	163	141	203	141	141
SLay05H	0.3	2.7	4.0	2.7	2.7	193	459	403	459	459
SLay05M	0.1	2.9	11.7	2.9	2.9	173	383	505	383	383
SLay06H	0.7	4.4	5.3	4.3	4.3	269	737	983	737	737
SLay06M	0.2	2.1	43.7	2.1	2.1	239	553	1847	553	553
SLay07H	1.2	12.5	26.0	12.7	12.8	283	3397	3103	3397	3397
SLay07M	0.2	5.8	34.3	5.9	5.8	229	1253	1975	1253	1253
SLay08H	5.6	33.1	142.3	33.0	32.7	1383	8127	22635	8127	8127
SLay08M	0.5	6.3	109.6	6.3	6.2	345	1509	6847	1509	1509
SLay09H	17.4	366.9	209.1	353.6	361.4	2547	81381	36587	81381	81381
SLay09M	1.4	35.8	184.5	36.1	35.7	973	14713	19595	14713	14713
SLay10H	67.6	3600	3600	3600	3600	6267	630205	610289	646571	664289
SLay10M	27.2	2663.6	3600	2662.0	2643.0	12751	569787	325579	569787	569787
sssd-10-4-3	6.5	4.3	6.5	4.1	4.3	4727	6055	6753	8373	8373
sssd-12-5-3	18.0	9.6	15.7	8.9	8.0	9501	20259	30999	14115	14115
sssd-15-6-3	166.0	936.9	160.7	898.5	881.3	64895	1886463	342961	1886463	1886463
sssd-16-7-3	3600	2053.4	3600	2677.6	2736.1	1594765	3387959	9592043	5480379	5480379
sssd-16-8-3	3600	3600	3600	3600	3600	1252061	8356545	9505289	8391307	8404611

sssd-17-7-3	3600	3600	3278.9	3600	3600	1445261	8570209	7326611	9266529	9254581
sssd-18-6-3	3600	3600	1380.0	2813.8	2817.3	1443437	10883179	3559589	7768851	7768851
sssd-18-7-3	3600	3553.4	3600	3583.7	3432.3	1407615	9380323	10207261	9511295	9511295
sssd-18-8-3	3600	3600	3600	3600	3600	1133977	11659205	8989449	9492803	9345517
sssd-20-7-3	3600	2284.2	3249.0	3600	3600	1343065	6088901	6163959	6746337	6748647
sssd-20-8-3	3600	3600	3600	3600	3600	1072727	9440523	11802627	9263157	9228627
sssd-20-9-3	3600	3600	3600	3600	3600	1007629	13032697	12428647	12720825	11890729
sssd-22-8-3	3600	3600	3600	3600	3600	1032709	13419985	13592163	13957805	13798237
sssd-8-4-3	2.6	2.7	4.0	3.4	3.4	2497	3727	3773	3065	3065
stockcycle	119.9	3600	3600	3600	3600	22263	667029	685131	652829	658249
Syn05H	0.1	0.3	0.3	0.2	0.3	1	3	3	3	3
Syn05M	0.1	0.2	0.2	0.2	0.4	11	11	11	11	11
Syn05M02H	0.2	0.2	0.4	0.2	0.2	3	1	3	1	1
Syn05M02M	0.2	0.4	0.4	0.4	0.4	19	21	21	21	21
Syn05M03H	0.3	0.2	0.5	0.2	0.3	7	1	5	1	1
Syn05M03M	0.4	0.5	0.4	0.4	0.4	29	31	37	37	37
Syn05M04H	0.2	0.3	0.5	0.3	0.3	5	1	5	1	1
Syn05M04M	0.6	0.6	0.5	0.5	0.5	43	53	53	53	53
Syn10H	0.2	0.1	0.1	0.1	0.1	3	3	3	3	3
Syn10M	0.2	0.4	0.4	0.3	0.4	35	35	35	35	35
Syn10M02H	0.7	0.4	0.3	0.4	0.4	5	5	5	5	5
Syn10M02M	1.6	1.7	1.7	1.7	1.7	237	247	247	247	247
Syn10M03H	0.6	0.4	0.3	0.4	0.4	9	3	3	3	3
Syn10M03M	7.8	2.4	2.6	3.2	3.2	691	811	803	775	775

Syn10M04H	0.9	0.5	0.4	0.5	0.5	9	7	7	7	7
Syn10M04M	27.8	4.8	5.2	5.2	5.3	1609	1859	1857	1859	1859
Syn15H	0.2	0.2	0.2	0.2	0.2	3	3	3	3	3
Syn15M	0.3	1.2	0.9	0.9	1.0	73	77	77	77	77
Syn15M02H	0.3	0.3	0.3	0.3	0.3	3	3	3	3	3
Syn15M02M	3.8	1.8	1.8	1.7	1.8	419	467	467	467	467
Syn15M03H	0.7	0.4	0.4	0.4	0.4	5	3	3	3	3
Syn15M03M	25.8	3.8	3.8	3.3	3.3	1539	1529	1775	1775	1775
Syn15M04H	1.1	0.4	0.5	0.4	0.4	7	3	3	3	3
Syn15M04M	126.2	10.7	10.5	10.6	10.9	4719	4853	4447	4057	4057
Syn20H	2.0	0.9	0.7	0.9	0.9	7	7	5	7	7
Syn20M	1.7	1.4	1.4	1.6	1.8	659	765	793	795	795
Syn20M02H	2.1	0.9	0.8	0.9	0.9	7	7	7	7	7
Syn20M02M	168.0	24.5	24.0	24.3	26.1	16081	18657	18871	19345	19345
Syn20M03H	10.5	1.5	1.2	1.7	2.0	17	21	17	21	21
Syn20M03M	2817.7	400.3	270.6	278.5	294.5	137083	173389	141471	154301	154301
Syn20M04H	12.7	2.0	1.5	2.2	2.1	21	21	21	21	21
Syn20M04M	3600	2269.8	2414.1	2275.7	2259.1	174249	818011	827905	827061	827061
Syn30H	0.3	1.3	0.7	1.3	1.1	5	13	7	13	13
Syn30M	17.8	4.9	5.3	5.8	5.9	4071	5831	5693	5617	5617
Syn30M02H	1.7	1.7	1.8	1.8	1.8	5	19	35	19	19
Syn30M02M	3600	1255.5	1366.9	1277.2	1277.9	255317	687647	750039	733667	733667
Syn30M03H	6.9	2.9	2.7	2.9	2.9	13	77	67	77	77
Syn30M03M	3600	3600	3600	3600	3600	127215	2393725	2548713	2258061	2298769

Syn30M04H	23.9	8.3	9.6	6.2	6.0	15	103	147	103	103
Syn30M04M	3600	3600	3600	3600	3600	71845	1791227	1751169	1900119	1942575
Syn40H	2.0	1.7	2.0	1.7	1.7	13	49	39	49	49
Syn40M	283.6	38.5	44.5	48.7	48.6	46493	53531	58105	59771	59771
Syn40M02H	10.4	3.6	2.8	4.0	4.0	11	57	73	73	73
Syn40M02M	3600	3600	3600	3600	3600	174061	2873029	2813591	2814371	2853199
Syn40M03H	30.6	12.4	11.8	12.0	11.3	37	169	209	241	241
Syn40M03M	3600	3600	3600	3600	3600	91539	1979081	1839729	1816529	1990499
Syn40M04H	197.5	13.8	15.0	13.3	14.0	135	249	287	219	219
Syn40M04M	3600	3600	3600	3600	3600	42869	1373425	1564479	1569145	1566349
synthes1	0.2	0.8	0.5	0.8	0.8	5	5	5	5	5
synthes2	0.2	0.5	0.5	0.5	0.5	11	17	17	17	17
synthes3	0.1	0.9	1.0	1.0	1.0	11	35	27	35	35
tls12	3600	3600	3600	3600	3600	54337	2966297	5624349	5293769	3741339
tls2	0.3	1.0	3.0	6.7	2.4	483	341	357	577	499
tls4	3600	1123.3	802.9	3600	771.9	50007	1065199	948027	722529	832563
tls5	3600	3600	3600	3600	3600	49665	3453963	3970909	688339	3519667
tls6	3600	3600	3600	3600	3600	400805	3315809	4374279	1330129	2955717
tls7	3600	3600	-1.0	3600	3600	77145	5231471	-1	2217949	5238785
trimloss12	3600	3600	3600	3600	3600	104177	1830249	3660979	4763789	3363859
trimloss2	0.3	0.9	2.2	5.7	1.5	401	301	365	463	381
trimloss4	3600	1050.5	660.1	3600	855.5	2927	1019861	822127	832717	981797
trimloss5	3600	3600	3600	3600	3600	85033	3649629	4640787	542095	4876829
trimloss6	3600	3600	3600	3600	3600	108745	5192509	7262715	1388469	5429209

trimloss7	3600	3600	3600	3600	3600	108745	5067913	7219489	1405769	5555875
uflquad-15-60	27.8	175.8	121.2	176.4	175.8	809	3395	1245	3395	3395
uflquad-15-80	66.2	433.1	331.9	431.9	429.8	821	4141	1655	4141	4141
uflquad-20-150	1188.0	3600	3600	3600	3600	2069	11035	2673	11453	11307
uflquad-20-40	22.7	225.7	190.8	227.4	233.6	905	4769	1913	4769	4769
uflquad-20-50	873.7	3600	3600	3600	3600	11543	34095	17723	35235	35387
uflquad-25-25	8.0	74.4	51.0	74.5	78.4	817	2219	829	2219	2219
uflquad-25-30	9.8	71.9	56.9	71.3	71.0	751	2015	809	2015	2015
uflquad-25-40	33.9	549.0	322.8	582.0	580.7	1397	8985	2583	8985	8985
uflquad-30-100	2637.2	3600	3600	3600	3600	5529	27265	1919	27383	27383
uflquad-40-80	3600	3600	3600	3600	3600	22133	37899	2969	38101	38225
Water0202	3600	-1.0	-1.0	-1.0	-1.0	17	-1	-1	-1	-1
Water0202R	0.6	1.3	1.7	1.3	1.5	23	31	23	31	31
Water0303	3600	-1.0	-1.0	-1.0	-1.0	15	-1	-1	-1	-1
Water0303R	6.3	11.5	12.0	11.3	11.4	137	451	407	451	451

Chapter 3

A Lagrangian relaxation approach to a class of mixed-integer bilevel programming problems

3.1 Introduction

In this chapter, we study the problem of service network design in a setting where customers choose which facilities to obtain their service from. In classic approaches, a network operator makes all decisions about location of facilities and determines the assignment of customer demands to facilities with the goal to minimize both the fixed cost associated with facilities and variable cost of meeting customer demands. The variable cost of a customer in a facility is assumed to be linear in the amount of demand of the customers served in the facility ([83].) However, in many applications, customers choose their serving facilities once the decision about location of facilities has been made. This suggests that a bilevel model better suits such a problem in which a system operator, called the *leader*, wishes to choose a subset of facilities to open as well as the size of the open facilities to balance customer service with cost of the facilities. Then, given a subset of open facilities, customers, also referred to as *followers*, choose which facilities to send their demand to ([70],[74].)

We study a bilevel service network design problem that captures the effect of facility congestion by considering a congestion function as the variable cost of customers. In this model, we assume that customers independently choose which facilities to have their demand sent to in order to minimize travel and waiting cost. Therefore, the customer demand

allocation decisions and resulting facility loads have to be determined as the solution of an equilibrium problem ([110].)

This setting appears in many applications. For example, in a preventive healthcare network design problem [121], a government wishes to maximize participation of potential clients from some regions in preventive healthcare services by improving the accessibility of facilities. On the other hand, clients choose open facilities which minimize time of both traveling and waiting to get served. Likewise, in a shelter location allocation problem ([107], [84]), local government decides to offer sheltering service in some places. Given a set of open shelters, once a flood or hurricane happens, people individually choose which shelter and which route they prefer.

We first assume that the capacity of each facility is given. By this assumption, we present the formulation of the problem in section 3.2. The lower-level of this problem is a continuous nonlinear program which we show is convex. Therefore, optimal solution of this problem can be characterized by KKT conditions. This allows the bilevel program to be formulated as a Mixed-Integer Nonlinear Programming (MINLP) problem which we present in section 3.3. However, this problem is very hard to solve so that even very small instances may not be solved to optimality in reasonable time by general purpose global optimization techniques. In section 3.4, we first derive a MINLP relaxation (referred to as MINLP-R) by relaxing the optimality requirement of the customers. However, because this MINLP is still hard to solve, we derive a Lagrangian dual problem which provides a lower bound to the optimal value of MINLP-R. We apply the subgradient algorithm to optimize the value of Lagrangian dual problem. In each iteration of the subgradient algorithm, the problem decomposes into $|M|$ nonconvex NLP subproblems. We propose a bisection algorithm to solve the subproblems which finds the global optimal solution efficiently. The Lagrangian dual problem also provides a set of candidate facilities to open, which we use to obtain a feasible solution to the original bilevel program. This feasible solution gives a valid upper bound to the value of bilevel problem. In section 3.5, we relax

the assumption of having fixed capacity and introduce a decision variable to represent the capacity of each facility. Then, we show the MINLP formulation and how the Lagrangian relaxation approach can be extended to this case. Numerical experiments are presented in section 3.6 which indicate that our approach can find solutions with small gap to the true optimal solution in reasonable time.

3.2 Problem formulation

Let the set of possible facilities be denoted by M with given capacities $K_j, j \in M$, and the set of customers be denoted by N with given demands $D_i, i \in N$ to be satisfied. We introduce a binary variable $z_j, j \in M$ denoting if the facility is open or not. Opening a facility j incurs a fixed cost f_j to the leader objective function. We define variable y_{ij} as the amount of demand of customer i supplied by facility j . The unit cost of sending demands from customer i to facility j is represented by c_{ij} . The congestion in a facility j , which we denote with $W_j(Y_j)$, is a function of the used capacity, i.e. $Y_j = \sum_{i \in N} y_{ij}$. We assume that the congestion function is increasing differentiable convex in Y_j and the cost of using all capacity, $W_j(K_j)$, is very high.

The problem can be formulated as follows:

$$Z_{\text{BILP}} = \min \sum_{j \in M} f_j z_j + \kappa \sum_{j \in M} \sum_{i \in N} \omega_i y_{ij} W_j(Y_j) \quad (3.1a)$$

$$\text{s.t.} \quad \sum_{i \in N} y_{ij} = Y_j, \quad j \in M, \quad (3.1b)$$

$$0 \leq Y_j \leq K_j, \quad j \in M \quad (3.1c)$$

$$y_{i.} \in \operatorname{argmin} \left\{ \sum_{j \in M} y_{ij} (c_{ij} + W_j(Y_j)) : y_{i.} \in R_i(z) \right\}, \quad i \in N, \quad (3.1d)$$

$$z_j \in \{0, 1\}, \quad j \in M. \quad (3.1e)$$

The objective function of the upper-level problem (3.1a), consists of two terms: one term for the cost of opening facilities and another term for the waiting time of customers which

can also be interpreted as the service level. In the service level, we assume that the leader might consider positive weights ω_i for customers, reflecting relative importance of customers.. The parameter $\kappa > 0$, which is chosen by the system operator, is a given weight of relative importance of service level to cost. Constraint (3.1b) prevents exceeding capacity usage of each facility. Constraint (3.1d) enforces optimality of flow variables, y_i , that associate with each customer $i \in N$. In the optimization problem embedded in this constraint, the objective function is the total cost imposed to customers, i.e. cost of travel and cost of congestion. In this constraint, set $R_i(z)$ denotes the feasible region of customer $i \in N$ which can be expressed as the following:

$$R_i(z) = \left\{ y_{i\cdot} \in \mathbb{R}_+^{|M|} : \sum_{j \in M} y_{ij} = D_i, y_{ij} \leq D_i z_j, j \in M \right\}.$$

This set defines decisions which satisfy all demand of a customer, no matter if the demand is sent to multiple facilities. Furthermore, demand can be sent to only open facilities.

In this bilevel formulation, we assumed that if customers have multiple optimal solutions for their problem, the leader can choose one that best contributes to its objective function. This is known as the *optimistic* approach in bilevel programming literature (see section 1.2.7.)

3.3 MINLP reformulation

The standard approach to deal with problem (3.1) is to substitute constraint (3.1d) by the optimality condition (1.5) of the embedded optimization problem. However, we need to confirm both the convexity of the problem and constraint qualifications which guarantee that KKT conditions characterize the global optimal solution.

The feasible region is defined by linear constraints, therefore linear independence constraint qualification (LICQ) (see section (1.2.3.1)) holds. We show that the objective function of (3.1d) is convex. Convexity of the objective function together with linearity of constraints affirm the convexity of the problem.

Constraint (3.1d) is written for each customer independently, so in term $y_{ij}W_j(Y_j)$, $y_{i'j}, i' \in N \setminus \{i\}$ is fixed. Write h_i for the objective function of (3.1d):

$$h_i(y_{i\cdot}) := \sum_{j \in M} \tilde{h}_{ij}(y_{ij})$$

where

$$\tilde{h}_{ij}(y_{ij}) := y_{ij}(c_{ij} + W_j(\sum_{i' \in N} y_{i'j})).$$

The first and second derivative of $\tilde{h}_{ij}$ are respectively

$$\tilde{h}'_{ij}(y_{ij}) = y_{ij} + W_j(\sum_{i' \in N} y_{i'j}) + y_{ij}W'_j(\sum_{i' \in N} y_{i'j}),$$

and

$$\tilde{h}''_{ij}(y_{ij}) = 2W'_j(\sum_{i' \in N} y_{i'j}) + y_{ij}W''_j(\sum_{i' \in N} y_{i'j}).$$

Since W_j is increasing convex, this implies $\tilde{h}''_{ij}(y_{ij}) \geq 0$, and therefore $\tilde{h}_{ij}$ is convex over the domain $y_{ij} \in \mathbb{R}_+$. It follows that h_i is convex on $\mathbb{R}_+^{[M]}$.

The KKT conditions for constraint (3.1d) of customer $i \in N$ are

$$\begin{aligned} \sum_{j \in M} y_{ij} &= D_i, \\ c_{ij} + W_j(Y_j) + y_{ij}W'_j(Y_j) + \lambda_{ij} - \mu_i - \alpha_{ij} &= 0, \quad j \in M, \\ 0 \leq D_i z_j - y_{ij} \perp \lambda_{ij} &\geq 0, \quad j \in M, \\ 0 \leq \alpha_{ij} \perp y_{ij} &\geq 0, \quad j \in M, \end{aligned} \tag{3.2}$$

where α_{ij} , λ_{ij} and μ_i are the associated dual multipliers.

Given the optimality condition (3.2), the MINLP reformulation of (3.1) is summarized as follows:

$$\begin{aligned}
Z_{\text{MINLP}} = \min \quad & \sum_{j \in M} f_j z_j + \kappa \sum_{j \in M} \sum_{i \in N} \omega_i y_{ij} W_j(Y_j) \\
\text{s.t.} \quad & \sum_{i \in N} y_{ij} = Y_j, & j \in M, \\
& 0 \leq Y_j \leq K_j z_j, & j \in M, \\
& \sum_{j \in M} y_{ij} = D_i, & i \in N, \\
& c_{ij} + W_j(Y_j) + y_{ij} W'_j(Y_j) + \lambda_{ij} & \\
& \quad -\mu_i - \alpha_{ij} = 0, & j \in M, \\
& 0 \leq D_i z_j - y_{ij} \perp \lambda_{ij} \geq 0, & i \in N, j \in M, \\
& 0 \leq y_{ij} \perp \alpha_{ij} \geq 0, & i \in N, j \in M, \\
& z_j \in \{0, 1\}, & j \in M.
\end{aligned} \tag{3.3}$$

We refer to this problem as *MINLP* reformulation.

3.4 Lagrangian relaxation for fixed capacity

In this section, we show how the Lagrangian relaxation approach is employed to obtain a solution and a lower bound to the bilevel program (3.1).

We first relax the bilevel programming formulation presented in (3.1) as follows:

$$Z_{\text{MINLPR}} = \min \quad \sum_{j \in M} f_j z_j + \kappa \sum_{j \in M} \sum_{i \in N} \omega_i y_{ij} W_j(Y_j) \tag{3.4a}$$

$$\text{s.t.} \quad \sum_{i \in N} y_{ij} = Y_j, \quad j \in M, \tag{3.4b}$$

$$0 \leq Y_j \leq K_j z_j, \quad j \in M, \tag{3.4c}$$

$$0 \leq y_{ij} \leq D_i z_j, \quad i \in N, j \in M, \tag{3.4d}$$

$$\sum_{j \in M} y_{ij} = D_i, \quad i \in N, \tag{3.4e}$$

$$z_j \in \{0, 1\}, \quad j \in M. \tag{3.4f}$$

We refer to this problem as *MINLP-R*. This relaxation is obtained by including constraints of the lower-level problem in the upper-level problem and relaxing optimality requirement of customers on y variables. Because of this, the problem is a centralized planning model, i.e. the leader chooses the flow between customers and facilities centrally regardless of preferences of the customers. The optimal value of this problem is a lower bound to Z_{BILP} , the optimal value of the original bilevel program (3.1).

In general, the Z_{MINLPR} may be much smaller than Z_{BILP} . However, in our experiments with instances of this problem, we have found that these values to be close. This suggests that for this problem, the centralized planning model appears to be a good approximation for the actual bilevel programming formulation.

We observe that in problem (3.4), constraint (3.4e) links variables associated with different facilities. By introducing Lagrange multiplier δ_i for this constraint, the Lagrangian function is

$$\begin{aligned}\mathcal{L}(z, Y, y, \delta) &= \sum_{j \in M} f_j z_j + \kappa \sum_{j \in M} \sum_{i \in N} \omega_i y_{ij} W_j(Y_j) + \sum_{i \in N} \delta_i (D_i - \sum_{j \in M} y_{ij}) \\ &= \sum_{i \in N} \delta_i D_i + \sum_{j \in M} \left(f_j z_j + \sum_{i \in N} y_{ij} (\kappa \omega_i W_j(Y_j) - \delta_i) \right).\end{aligned}$$

Given multiplier $\delta \in \mathbb{R}^{|N|}$, the Lagrangian relaxation problem is

$$\begin{aligned}Z_{\text{LR}}(\delta) &= C(\delta) + \min_{z, Y, y} \sum_{j \in M} (f_j z_j + \sum_{i \in N} y_{ij} (\kappa \omega_i W_j(Y_j) - \delta_i)) \\ \text{s.t. } &\sum_{i \in N} y_{ij} = Y_j, & j \in M, \\ &0 \leq Y_j \leq K_j z_j, & j \in M, \\ &0 \leq y_{ij} \leq D_i z_j, & i \in N, j \in M, \\ &z_j \in \{0, 1\}, & j \in M,\end{aligned}\tag{3.5}$$

where $C(\delta) = \sum_{i \in N} \delta_i D_i$. The value $Z_{\text{LR}}(\delta)$ is a lower bound to the value of relaxation problem stated in (3.4), hence a lower bound to Z_{BILP} . The Lagrangian dual problem is

$$Z_{\text{DUAL}} = \max_{\delta} Z_{\text{LR}}(\delta).\tag{3.6}$$

Problem (3.5) can be decomposed into $|M|$ subproblems, one for each $j \in M$. The j th subproblem is

$$\begin{aligned}
 Z_{\text{LR}}^j(\delta) = \min_{z, Y, y} \quad & f_j z_j + \sum_{i \in N} y_{ij} (\kappa \omega_i W_j(Y_j) - \delta_i) \\
 \text{s.t.} \quad & \sum_{i \in N} y_{ij} = Y_j, \\
 & 0 \leq Y_j \leq U_j, \\
 & 0 \leq y_{ij} \leq D_i z_j, \quad i \in N, \\
 & z_j \in \{0, 1\}.
 \end{aligned} \tag{3.7}$$

Then $Z_{\text{LR}}(\delta) = C(\delta) + \sum_{j \in M} Z_{\text{LR}}^j(\delta)$. There is only a single binary variable in subproblem (3.7). If $z_j = 0$, then $Z_{\text{LR}}^j(\delta) = 0$. If $z_j = 1$, the problem becomes a form of nonlinear knapsack problem. In this case, we can write the problem (3.7) as

$$Z_{\text{LR}}^j(\delta) = f_j + \min_{s_j \in [0, U_j]} g_j(Y_j), \tag{3.8}$$

where

$$\begin{aligned}
 g_j(Y_j) = \min_y \quad & \sum_{i \in N} y_{ij} (\kappa \omega_i W_j(Y_j) - \delta_i) \\
 \text{s.t.} \quad & \sum_{i \in N} y_{ij} = Y_j, \\
 & 0 \leq y_{ij} \leq D_i, \quad i \in N.
 \end{aligned} \tag{3.9}$$

Problem (3.9) is a linear knapsack problem which can be solved as follows. For $j \in M$, and given value for Y_j , to find the optimal solution, we sort the coefficients of $y_{ij}, i \in N$. Suppose σ is a permutation of $\{1, \dots, |N|\}$ such that for given $Y \in [0, K_j]$ one has $\kappa \omega_{\sigma(|N|)} W_j(Y) - \delta_{\sigma(|N|)} \geq \dots \geq \kappa \omega_{\sigma(2)} W_j(Y) - \delta_{\sigma(2)} \geq \kappa \omega_{\sigma(1)} W_j(Y) - \delta_{\sigma(1)}$. We denote

$$i^*(Y) = \max \left\{ i : \sum_{k=1}^i D_{\sigma(k)} \leq Y \right\}.$$

Then, the solution of the knapsack problem for Y is: $y_{\sigma(i)j} = D_{\sigma(i)}, i \leq i^*(Y), y_{\sigma(i^*(Y)+1)j} = Y - \sum_{k=1}^{i^*(Y)} D_{\sigma(k)}$ and $y_{\sigma(i)j} = 0, i > i^*(Y) + 1$.

We analyze the behavior of function $g_j(Y)$ in Theorem 3.1. Without loss of generality, we assume that $\kappa \omega_i W_j(Y) - \delta_i, i = 1, \dots, |N|$ values are ordered increasingly. Let's write

$\tilde{K}_j(i) := \sum_{l \leq i} D_l$ and $i^* := \max\{i : \tilde{K}_j(i) \leq K_j\}$. The domain of g_j is divided into $i^* + 1$ subintervals $\tilde{K}(i^* + 1) > \tilde{K}(i^*) > \tilde{K}(i^* - 1) > \dots > \tilde{K}(1) > \tilde{K}(0)$ where $\tilde{K}_j(0) = 0$ and $\tilde{K}_j(i^* + 1) = K_j$.

Theorem 3.1. Assume $\omega_i = 1, i \in N$ and $H(Y) = YW(Y)$ is convex on $Y \in [0, K_j]$. Then, g_j is convex over its domain $[0, K]$.

Proof. We first rewrite the objective function of (3.9):

$$\begin{aligned} \sum_{i \in N} y_i (\kappa W(Y) - \delta_i) &= \kappa W(Y) \sum_{i \in N} y_i - \sum_{i \in N} \delta_i y_i \\ &= \kappa YW(Y) - \sum_{i \in N} \delta_i y_i, \end{aligned}$$

where we used $\sum_{i \in N} y_i = Y$ constraint in the second equation. Now, the optimization problem (3.9) is simplified as

$$\begin{aligned} g_j(Y) &= \kappa YW(Y) + \min_y \quad - \sum_{i \in N} \delta_i y_i \\ &\quad \text{s.t.} \quad \sum_{i \in N} y_i = Y, \\ &\quad \quad \quad 0 \leq y_i \leq D_i, \quad i \in N, \\ &= \kappa YW(Y) + F(Y), \end{aligned}$$

where $F(Y)$ is the value function of the linear program with Y being in the right-hand side. Therefore, $F(Y)$ is convex and the results follows. $\square$

Conjecture 3.2. g_j is convex over its domain $[0, K]$ for general values for ω_i .

The result of Theorem 3.1 and Conjecture (3.2) is used to solve the subproblem (3.7) when $z_j = 1$. In this case, we only need to minimize $g_j(Y)$. Since g_j is convex, we can find its minimizer Y^* by the bisection method presented in Algorithm 8 ([13].)

Input: (δ, j) and tolerance $\epsilon > 0$

Output: (g_j^*, y^*)

Algorithm: subproblem(δ, j)

Let $Y_l = 0, Y_u = K_j, Y_m = \frac{Y_l + Y_u}{2}, Y_{m_l} = \frac{Y_l + Y_m}{2}$ and $s_{m_u} = \frac{s_m + s_u}{2}$.

Compute $g_j(Y_l), g_j(Y_u), g_j(Y_m), g_j(Y_{m_l})$ and $g_j(Y_{m_u})$.

$Y^* = \operatorname{argmin}\{g_j(Y) : Y \in \{Y_l, Y_{m_l}, Y_m, Y_{m_u}, Y_u\}, \text{ and } g_j^* = g_j(Y^*)\}.$

while $|Y_l - Y_u| > \epsilon$ **do**

if $g_j^* = g_j(Y_l)$ **or** $g_j^* = g_j(Y_{m_l})$ **then** Let $Y_u := Y_m$.

else if $g_j^* = g_j(Y_u)$ **or** $g_j^* = g_j(Y_{m_u})$ **then** Let $Y_l := Y_m$.

else Let $Y_l := Y_{m_l}$ and $Y_u := Y_{m_u}$.

 Update $Y_{m_l} = \frac{Y_l + Y_m}{2}$ and $Y_{m_u} = \frac{Y_m + Y_u}{2}$.

 Compute $g_j(Y_{m_l})$ and $g_j(Y_{m_u})$.

$Y^* = \operatorname{argmin}\{g_j(Y) : Y \in \{Y_l, Y_{m_l}, Y_m, Y_{m_u}, Y_u\}, \text{ and } g_j^* = g_j(Y^*)\}.$

Let y^* be the solution of (3.9) for $Y = Y^*$.

Algorithm 8: Binary search to $\min_s g_j(s)$ in (3.8)

We can then apply a subgradient-based algorithm to solve the Lagrangian dual problem (3.6). In each iteration, we solve the subproblem (3.7) for all $j \in M$ by applying Algorithm 8 for given Lagrange Multipliers δ . If the cost of opening facility j , denoted by $g_j^* + f_j$, is positive, facility j is closed, i.e. $\hat{z}_j = 0$. Otherwise, facility j is open, i.e. $\hat{z}_j = 1$. Algorithm 9 summarizes these steps in more details. This algorithm is proved to be convergent equipped with geometric updating rule starting with sufficiently large ρ and μ (see [119].) As a heuristic stopping criterion, we compute the relative difference of two steps, say $Z_{LR}(\delta^{k-b})$ and $Z_{LR}(\delta^k)$, as

$$\frac{|Z_{LR}(\delta^k) - Z_{LR}(\delta^{k-b})|}{Z_{LR}(\delta^k)},$$

for some constant positive integer b . If this value is less than a predetermined value $\epsilon > 0$, we stop the algorithm.

Output: $(\hat{z}, Z_{\text{LR}})$

Initialize $\delta_i := 0, i \in N$, sufficiently large $\mu > 0$ and $\rho \in (0, 1)$.

$k := 0$.

while *stopping criteria not met* **do**

Let $Z_{\text{LR}} := 0$.

Compute the constant term $C(\delta) = \sum_{i \in N} \delta_i D_i$.

for $j = 1 : |M|$ **do**

$(g_j^*, y_{\cdot j}^*) = \text{subproblem}(\delta, j)$.

if $g_j^* + f_j > \epsilon$ **then**

Let $y_{\cdot j}^* := 0$ and $\hat{z}_j := 0$.

else

$\hat{z}_j := 1$ and update $Z_{\text{LR}} \leftarrow Z_{\text{LR}} + g_j^* + f_j$.

Update $\delta_i \leftarrow \delta_i + \mu \cdot (D_i - \sum_{j \in M} y_{ij}^*)$ for all $i \in M$.

$\mu \leftarrow \mu \cdot \rho, k \leftarrow k + 1, Z_{\text{LR}} \leftarrow Z_{\text{LR}} + C(\delta)$.

Algorithm 9: Subgradient algorithm to solve Lagrangian dual (3.6)

Algorithm 9 finds a solution $\hat{z}$ which represents which facilities are open or closed. If we fix z in MINLP (3.3), and solve the resulting NLP and obtain a solution, then this solution is a feasible to bilevel program (3.1). However, in computational experiments, we observed that general purpose NLP solvers cannot solve this NLP in a reasonable time. We next show how we derive an alternative NLP formulation which we found to be computationally more tractable.

As we showed in section 3.3, the optimization problem of constraint (3.1d) is a convex programming problem. Given $\hat{z}$, let $\mathcal{L}_i(y, Y, \lambda, \mu, \alpha; \hat{z})$ denote the Lagrangian function

for customer i optimization problem where $\lambda_{ij} \geq 0$, μ_i and $\alpha_{ij} \geq 0$ are the associated Lagrange multipliers with the constraints. The dual function is defined as,

$$\begin{aligned} q_i(\lambda, \mu, \alpha; \hat{z}) &:= \min_{Y, y} \mathcal{L}_i(y, Y, \lambda, \mu, \alpha; \hat{z}) \\ &= \sum_{j \in M} -D_i \lambda_{ij} \hat{z}_j + \mu_i D_i + \min_{y_{i, Y}} \sum_{j \in M} y_{ij} (c_{ij} + W_j(Y_j) + \lambda_{ij} - \mu_i + \alpha_{ij}). \end{aligned}$$

We write $h_{ij}(y) = y_{ij}(c_{ij} + W_j(\sum_{i' \in N} y_{i'j}) + \lambda_{ij} - \mu_i + \alpha_{ij})$. This is a convex function; hence its global minimizer is attained at the solution of $h'_{ij}(y) = 0$:

$$\begin{aligned} h'_{ij}(y) &= c_{ij} + W_j(\sum_{i' \in N} y_{i'j}) + \lambda_{ij} - \mu_i - \alpha_{ij} + y_{ij} W'_j(\sum_{i' \in N} y_{i'j}) = 0 \quad \Rightarrow \\ c_{ij} + \lambda_{ij} - \mu_i + W_j(\sum_{i' \in N} y_{i'j}) - \alpha_{ij} &= -y_{ij} W'_j(\sum_{i' \in N} y_{i'j}). \end{aligned} \quad (3.10)$$

The equation in (3.10) can be used to simplify the function q_i for customer i :

$$q_i(\lambda, \mu, \alpha; \hat{z}) = - \sum_{j \in M} D_i \lambda_{ij} \hat{z}_j + \mu_i D_i - \sum_{j \in M} y_{ij}^2 W'_j(Y_j).$$

We now define the dual problem by maximizing q_i as:

$$\begin{aligned} \max_{\lambda, \alpha, \mu} \quad & \mu_i D_i - \sum_{j \in M} D_i \lambda_{ij} \hat{z}_j - \sum_{j \in M} y_{ij}^2 W'_j(Y_j) \\ \text{s.t.} \quad & c_{ij} + W_j(Y_j) + \lambda_{ij} - \mu_i - \alpha_{ij} + y_{ij} W'_j(Y_j) = 0, \quad j \in M \\ & \lambda_{ij} \geq 0, \quad \alpha_{ij} \geq 0, \quad j \in M. \end{aligned}$$

By eliminating α , we obtain:

$$\max_{\mu, \lambda} \quad \mu_i D_i - \sum_{j \in M} D_i \lambda_{ij} \hat{z}_j - \sum_{j \in M} y_{ij}^2 W'_j(Y_j)$$

$$\text{s.t.} \quad c_{ij} + W_j(Y_j) + \lambda_{ij} - \mu_i + y_{ij} W'_j(Y_j) \geq 0, \quad j \in M \quad (3.11a)$$

$$\lambda_{ij} \geq 0, \quad j \in M. \quad (3.11b)$$

Since the optimization problem in constraint (3.1d) is a convex program, strong duality holds. Thus for customer i , one could write the optimality condition as

$$\mu_i D_i - \sum_{j \in M} D_i \hat{z}_j \lambda_{ij} \geq \sum_{j \in M} y_{ij} (c_{ij} + W_j(Y_j) + y_{ij} W'_j(Y_j)), \quad j \in M \quad (3.12a)$$

$$y_{i\cdot} \in R_i(\hat{z}), \quad (3.12b)$$

$$(3.11a), (3.11b). \quad (3.12c)$$

We replace constraint (3.1d) by optimality conditions (3.12):

$$\begin{aligned} Z_{\text{NLP}} = \min \quad & \sum_{j \in M} f_j \hat{z}_j + \kappa \sum_{j \in M} \sum_{i \in N} \omega_i y_{ij} W_j(Y_j) \\ \text{s.t.} \quad & \sum_{i \in N} y_{ij} = Y_j, \quad j \in M, \\ & 0 \leq Y_j \leq U_j \hat{z}_j, \quad j \in M, \\ & y_{i\cdot} \in R_i(\hat{z}), \quad i \in N, \\ & (3.11a), (3.11b), (3.12a), \quad i \in N. \end{aligned} \quad (3.13)$$

This problem is referred to as *fixed-NLP*. Problem (3.13) is a nonconvex optimization problem and $Z_{\text{NLP}} \geq Z_{\text{BILP}}$. Any feasible solution to this problem, obtained from general purpose solvers, still yields a valid upper bound to Z_{BILP} .

In summary, the method consists of two steps. In first step, we run the subgradient algorithm summarized in Algorithm 9 to obtain a solution $\hat{z}$ and a value for the dual problem which is a lower bound to Z_{BILP} . We then solve fixed-NLP for $\hat{z}$. If this problem returns a solution, then that solution is a feasible solution to bilevel program (3.1) and the optimal value of fixed-NLP is an upper bound to Z_{BILP} .

3.5 Capacity size as a decision variable

In this section, we extend the problem described in Section 3.2 to the cases where the capacity of the facilities are assumed to be a decision variable. In this section, we assume that all customers have equal importance weights, i.e. $\omega_i = 1, i \in N$. Let U_j denote the

size of facility j . In addition to the fixed cost of opening facilities, we add a variable cost of capacity size $U_j, j \in M$ in the objective function of the leader.

$$Z_{\text{BILP}}^U = \min_{z, U, y, Y} \sum_{j \in M} f_j z_j + \sum_{j \in M} a_j U_j + \kappa \sum_{j \in M} \sum_{i \in N} y_{ij} W_j(U_j, Y_j) \quad (3.14a)$$

$$\text{s.t.} \quad \sum_{i \in N} y_{ij} = Y_j, \quad j \in M, \quad (3.14b)$$

$$0 \leq Y_j \leq U_j, \quad j \in M \quad (3.14c)$$

$$0 \leq U_j \leq K_j z_j, \quad j \in M, \quad (3.14d)$$

$$y_{i \cdot} \in \operatorname{argmin} \left\{ \sum_{j \in M} y_{ij} (c_{ij} + W_j(U_j, Y_j)) : y_{i \cdot} \in R_i(z) \right\}, \quad i \in N, \quad (3.14e)$$

$$z_j \in \{0, 1\}, \quad j \in M.$$

where a_j is the unit cost of capacity size. The MINLP reformulation of (3.14) is obtained by replacing the KKT conditions of optimization problem in constraint (3.1d) and summarized as follows:

$$\begin{aligned} Z_{\text{MINLP}}^U &= \min \quad \sum_{j \in M} (f_j z_j + a_j U_j) + \kappa \sum_{j \in M} \sum_{i \in N} y_{ij} W_j(Y_j) \\ \text{s.t.} \quad &\sum_{j \in M} y_{ij} = D_i, \quad i \in N, \\ &c_{ij} + W_j(U_j, Y_j) + y_{ij} \frac{\partial W_j(U_j, Y_j)}{\partial y_{ij}} + \lambda_{ij} \\ &\quad - \mu_i - \alpha_{ij} = 0, \quad j \in M, \\ &0 \leq D_i z_j - y_{ij} \perp \lambda_{ij} \geq 0, \quad i \in N, j \in M, \\ &0 \leq y_{ij} \perp \alpha_{ij} \geq 0, \quad i \in N, j \in M, \\ &(3.14b), (3.14c), (3.14d), \\ &z_j \in \{0, 1\}, \quad j \in M. \end{aligned} \quad (3.15)$$

The MINLP relaxation then becomes:

$$\min \sum_{j \in M} (f_j z_j + a_j U_j) + \kappa \sum_{j \in M} \sum_{i \in N} y_{ij} W_j(U_j, Y_j) \quad (3.16a)$$

$$\text{s.t. } 0 \leq y_{ij} \leq D_i z_j, \quad i \in N, j \in M, \quad (3.16b)$$

$$\sum_{j \in M} y_{ij} = D_i, \quad i \in N, \quad (3.16c)$$

$$(3.14b), (3.14c), (3.14d),$$

$$z_j \in \{0, 1\}, \quad j \in M.$$

This problem is a nonconvex MINLP because of the $y_{ij} W_j(U_j, Y_j)$ term in the objective function.

We now relax constraint (3.16c) and, for a given dual multiplier $\delta \in \mathbb{R}^{|N|}$, obtain the Lagrangian relaxation problem:

$$\begin{aligned} Z_{\text{LR}}(\delta) = C(\delta) + \min_{z, y, Y, U} \quad & \sum_{j \in M} (f_j z_j + a_j U_j + \sum_{i \in N} y_{ij} (\kappa W_j(U_j, Y_j) - \delta_i)) \\ \text{s.t.} \quad & (3.14b), (3.14c), (3.14d), (3.16c), \\ & z_j \in \{0, 1\}, \quad j \in M, \end{aligned}$$

where $C(\delta) = \sum_{i \in N} \delta_i D_i$. To find the value of δ that yields a strong lower bound, we solve the Lagrangian dual problem:

$$Z_{\text{DUAL}} = \max_{\delta} Z_{\text{LR}}(\delta). \quad (3.17)$$

Problem (3.17) is decomposed into $|M|$ subproblems. The j th subproblem is

$$\begin{aligned} Z_{\text{LR}}^j(\delta) = \min_{z, s, y} \quad & f_j z_j + a_j U_j + \sum_{i \in N} y_{ij} (\kappa W_j(U_j, Y_j) - \delta_i) \\ \text{s.t.} \quad & (3.14b), (3.14c), (3.14d), (3.16c), \\ & z_j \in \{0, 1\}. \end{aligned}$$

If facility j is decided to be closed, then $z_j = 0$, hence $Z_{\text{LR}}^j(\delta) = 0$. Otherwise, the subproblem can be simplified as

$$\begin{aligned} Z_{\text{LR}}^j(\delta) = f_j + \min_{U_j, Y_j} \quad & a_j U_j + g_j(U_j, Y_j), \\ \text{s.t.} \quad & 0 \leq Y_j \leq U_j, \\ & U_j \leq K_j, \end{aligned} \quad (3.18)$$

where

$$\begin{aligned} g_j(U_j, Y_j) = \min_y \quad & \sum_{i \in N} y_{ij} (\kappa W_j(U_j, Y_j) - \delta_i) \\ \text{s.t.} \quad & \sum_{i \in N} y_{ij} = Y_j, \\ & 0 \leq y_{ij} \leq D_i, \quad i \in N. \end{aligned} \quad (3.19)$$

Theorem 3.3. *Suppose that $H(U, Y) = YW(U, Y)$ is convex and W is continuous on $[0, K]$. Then, g is convex and continuous on its domain.*

Proof. Let us rewrite the objective function of (3.19):

$$\begin{aligned} \sum_{i \in N} y_i (\kappa W(U, Y) - \delta_i) &= \kappa W(U, Y) \sum_{i \in N} y_i - \sum_{i \in N} y_i \delta_i \\ &= \kappa Y W(U, Y) - \sum_{i \in N} y_i \delta_i, \end{aligned}$$

where we used the constraint of (3.19) in the equation. Now, the optimization problem is simplified as

$$\begin{aligned} g(U, Y) &= \kappa Y W(U, Y) + \min_y \quad - \sum_{i \in N} y_i \delta_i \\ &\quad \text{s.t.} \quad \sum_{i \in N} y_i = Y, \\ &\quad \quad 0 \leq y_i \leq D_i, \quad i \in N, \\ &= \kappa Y_j W_j(U_j, Y_j) + F(Y), \end{aligned}$$

where $F(Y)$ is the value function of a linear program with Y in the right-hand side. Therefore, $F(Y)$ is convex, and by assumption, the result follows. $\square$

In order to solve (3.18), we reformulate its embedded optimization problem as follows:

$$\begin{aligned}
& \min_{\eta, U_j, Y_j} && a_j U_j + \eta, \\
& \text{s.t.} && 0 \leq Y_j \leq U_j, \\
& && U_j \leq K_j, \\
& && \eta \geq g_j(U_j, Y_j).
\end{aligned} \tag{3.20}$$

Since g_j is convex, the constraint $\eta \geq g_j(U_j, Y_j)$ can be approximated by

$$\eta \geq g_j(\hat{U}_j, \hat{Y}_j) + \hat{s}_U(U_j - \hat{U}_j) + \hat{s}_Y(Y_j - \hat{Y}_j),$$

where $(\hat{s}_U, \hat{s}_Y)$ is a subgradient taken at $(\hat{U}_j, \hat{Y}_j)$. To obtain a subgradient, we take the dual of linear program defining function F :

$$\begin{aligned}
g_j(U_j, Y_j) = & \kappa Y_j W_j(U_j, Y_j) + \max_{\rho, \pi_i} \rho Y_j - \sum_{i \in N} D_i \pi_i \\
& \text{s.t.} \quad \rho - \pi_i \leq -\delta_i, & i \in N, \\
& \pi_i \geq 0, & i \in N,
\end{aligned}$$

Let $(\rho^*, \delta_i^*, \dots, \delta_{|N|}^*)$ denote the optimal solution of the dual. Then,

$$g_j(U_j, Y_j) = \kappa Y_j W_j(U_j, Y_j) + \rho^* Y_j - \sum_{i \in N} D_i \pi_i^*. \tag{3.21}$$

We can then derive a subgradient for it which can be obtained from (3.21):

$$s_Y = \frac{\partial g_j}{\partial Y_j} = \kappa W_j(U_j, Y_j) + \kappa Y_j \frac{\partial W_j}{\partial Y_j} + \rho^*,$$

and

$$s_U = \frac{\partial g_j}{\partial U_j} = \kappa Y_j \frac{\partial W_j}{\partial U_j}.$$

Using a cutting plane approach, we solve problem (3.20).

Now that we have the method to minimize $g_j(U_j, Y_j)$, we can solve the Lagrangian dual problem (3.17) using the subgradient method described in Algorithm 10.

To obtain an upper bound, we can first replace the lower level problem in bilevel program (3.14) with the optimality conditions, using an argument analogous to one we made

Output: $(\hat{z}, Z_{\text{LR}})$

Initialize $\delta_i := 0, i \in N$, sufficiently large $\mu > 0$ and $\rho \in (0, 1)$.

$k := 0$.

while *stopping criteria not met* **do**

 Let $Z_{\text{LR}} := 0$.

 Compute the constant term $C(\delta) = \sum_{i \in N} \delta_i D_i$.

for $j = 1 : |M|$ **do**

$(g_j^*, y_{.j}^*, U_j^*) = \text{cutting-plane}(\delta, j)$.

if $g_j^* + f_j + a_j U_j^* > \epsilon$ **then**

 Let $y_{.j}^* := 0$ and $\hat{z}_j := 0$.

else

$\hat{z}_j := 1$ and update $Z_{\text{LR}} \leftarrow Z_{\text{LR}} + g_j^* + f_j + a_j U_j^*$.

 Update $\delta_i \leftarrow \delta_i + \mu \cdot (D_i - \sum_{j \in M} y_{ij}^*)$ for all $i \in M$.

$\mu \leftarrow \mu \cdot \rho, k \leftarrow k + 1, Z_{\text{LR}} \leftarrow Z_{\text{LR}} + C(\delta)$.

Algorithm 10: Subgradient algorithm to solve Lagrangian dual (3.6)

in Section 3.4. The optimality conditions similar to (3.12) for the case of variable capacity size are:

$$\mu_i D_i - \sum_{j \in M} D_i \hat{z}_j \lambda_{ij} \geq \sum_{j \in M} y_{ij} (c_{ij} + W_j(U_j, Y_j) + y_{ij} \frac{\partial W_j(U_j, Y_j)}{\partial y_{ij}}), \quad (3.22a)$$

$$c_{ij} + W_j(U_j, Y_j) + y_{ij} \frac{\partial W_j(U_j, Y_j)}{\partial y_{ij}} + \lambda_{ij} - \mu_i \geq 0 \quad j \in M \quad (3.22b)$$

$$y_{i \cdot} \in R_i(\hat{z}), \quad (3.22c)$$

$$\lambda_{ij} \geq 0, \quad j \in M. \quad (3.22d)$$

The resulting optimization problem is a MINLP. However, we fix the binary variable z with $\hat{z}$ obtained from Algorithm 10. The resulting problem is the following NLP:

$$\begin{aligned}
Z_{\text{NLP}} = \min \quad & \sum_{j \in M} (f_j \hat{z}_j + a_j U_j) + \kappa \sum_{j \in M} \sum_{i \in N} y_{ij} W_j(U_j, Y_j) \\
\text{s.t.} \quad & \sum_{i \in N} y_{ij} = Y_j, & j \in M, \\
& 0 \leq Y_j \leq U_j, & j \in M, \\
& 0 \leq U_j \leq K_j \hat{z}_j, & j \in M, \\
& y_{i \cdot} \in R_i(\hat{z}), & i \in N, \\
& (3.22a), (3.22b), (3.22d), & i \in N.
\end{aligned} \tag{3.23}$$

Any local solution of this NLP is a true feasible solution to the original bilevel program (3.14), hence an upper bound is obtained for Z_{BILP}^U .

3.6 Computational Results

In this section, we discuss the computational results for the algorithms presented in previous sections. We first present how the test instances are generated. Then, we show the results of the algorithms for fixed capacity with equal and unequal weights, and variable capacity with equal weights.

3.6.1 Test instances and settings

We randomly generated test instances as follows. We first choose problem sizes (number of facilities and customers.) We generated fixed costs $f_j, j \in M$ by $\text{U}(10, 200)$ and demands $D_i, i \in N$ by $\text{U}(300, 500)$ where $\text{U}(a, b)$ represents uniform distribution in an interval $[a, b]$. Given demands, capacity $K_j, j \in M$ is generated by $\text{U}(1.5\bar{D}, 2.5\bar{D})$ where $\bar{D} = \sum_i D_i / |N|$. We also generated $|M| + |N|$ random points in $[0, 150]^2$ representing location of facilities and customers, respectively. The Euclidean norm between these points is used as c_{ij} . The congestion we use is a quadratic function ([103])

$$W_j(Y) = M_j \left(\frac{Y}{K_j} \right)^2,$$

where $M_j = 250$. For each problem size, three sets of data sets are generated. We also consider three values for parameter κ to analyze the sensitivity of the model to this parameter. The values were chosen in a range that yielded solutions with a reasonable balance between facility cost and congestion.

For the case of fixed capacity, to determine proper values for parameter κ , we compute the utilization of facilities in solutions of Lagrangian relaxation for three levels of κ as follows

$$\text{Utilization} = \frac{Y_j}{K_j}.$$

The average of utilization in each problem size and number of open facilities are presented in Table 3.1. A large value for κ results in opening more facilities to give better service. Beyond a certain threshold, all facilities are preferred to be open. Below a certain level, although the Lagrangian relaxation returns a solution, fixed-NLP is infeasible, because the amount of capacity is below the required demand. We chose 0.0005, 0.001 and 0.002 as three values for parameter κ for further comparisons between formulations. These values generated reasonable solutions in which the utilization of facilities lies in (0.55, 0.85) for different sizes. Also, about 50% to 90% of facilities are open.

		κ		0.0005		0.001		0.002	
$ N $	$ M $	# open	Util.	# open	Util.	# open	Util.	# open	Util.
5	5	3.00	0.800	4.00	0.667	4.30	0.577		
10	10	6.32	0.754	7.31	0.658	8.32	0.567		
50	50	28.33	0.833	34.33	0.692	40.66	0.582		
100	100	57.66	0.825	68.99	0.692	81.33	0.585		

Table 3.1 Summary results for sensitivity of model to κ

We applied the same approach for the case of variable capacity and chose 0.0025, 0.004 and 0.007 which balance the service level and cost of opening the facilities. These

instances also require values for the unit cost of capacities, i.e., $a_j, j \in M$, which were randomly generated by $U(0.1, 2)$.

3.6.2 Results for fixed capacity size

In our computational experiments, we compare the performance of our proposed subgradient algorithm for Lagrangian dual problem (3.6) with MINLP formulation (3.3). We solve MINLP formulation (3.3) using the global optimization solver BARON 12.7.3 in GAMS with 3 hours time limit. The solution obtained from Algorithm 9 is used to obtain fixed-NLP (3.13) which is implemented in GAMS and solved using KNITRO 9.0.0.

In the other experiment, we solve MINLP relaxation problem (3.4) using convex MINLP solver bonmin and show the relative gap between its value with an upper bound of Z_{BILP} . The solution of this problem can also be used to construct the fixed-NLP (3.13) which gives an upper bound.

Table 3.2 summarizes the results of solving MINLP formulation (3.3), MINLP relaxation (3.4) and Lagrangian relaxation in plus fixed-NLP described in Algorithm 9. For MINLPs, the table lists geometric mean of CPU time in seconds, geometric mean of the number of nodes of branch-and-bound, geometric mean of relative gaps between lower and upper bound, where lower bound is the lower bound returned by the solver, i.e. BARON for MINLP and bonmin for MINLP relaxation. For the upper bound, we use two values. If we use the upper bound returned by the solver, BARON or bonmin, we denote the gap by "Gap-A". Alternatively, we write "Gap-B" if we use the best known upper bound to Z_{BILP} , the smallest between the value of fixed-NLP with the solution of Lagrangian relaxation, value of fixed-NLP with the solution of MINLP relaxation and upper bound returned by BARON for MINLP formulation if any upper bound was found. For the Lagrangian relaxation, the table displays the geometric mean of CPU time in seconds for both subgradient algorithm and fixed-NLP, the geometric mean of number of iterations of subgradient method, Gap-A which is the relative gap between the lower bound (value of

the Lagrangian relaxation) and fixed-NLP with the solution of the Lagrangian relaxation, and Gap-B, the gap between the lower bound and the best known upper bound. In our method, the average CPU time to obtain a feasible solution consists of time of both the subgradient algorithm and solving fixed-NLP. However, fixed-NLP owns at most 2.5% of the total time displayed in Table 3.2.

$ N $	$ M $	κ	MINLP				MINLP-R				Lagrangian dual + fixed-NLP			
			Time(s)	Node	% Gap-A	% Gap-B	Time(s)	Node	% Gap-A	% Gap-B	Time(s)	Iteration	% Gap-A	% Gap-B
5	5	0.0005	49.7	3065.6	-	-	0.3	17.4	-	-	21.1	32918.8	0.40	0.40
		0.001	278.3	12997.8	-	-	0.3	26.4	-	0.34	17.3	32079.6	2.00	1.64
		0.002	1011.8	34518.6	-	-	0.3	22.3	-	0.47	18.9	34027.8	0.84	0.62
10	10	0.0005	10800	461092.6	53.77	48.36	2.1	221.6	-	0.24	13.8	36975.7	0.26	0.26
		0.001	10800	612090.3	63.65	61.47	2.1	226.1	-	0.47	14.1	37229.8	0.86	0.86
		0.002	10800	283249.2	72.10	70.18	1.6	163.0	-	0.89	14.4	37012.8	1.04	1.04
50	50	0.0005	10800	106.9	99.03	63.92	10800	19772.2	60.27	60.08	484.2	39296.3	0.24	0.24
		0.001	10800	147.9	99.50	74.40	10800	7927.9	58.57	58.29	481.7	39104.7	0.41	0.41
		0.002	10800	104.8	99.75	81.72	10800	3689.2	51.67	51.84	479.2	38492.5	0.75	0.74
100	100	0.0005	10800	*	*	*	10800	171.5	79.14	77.81	2578.7	38492.5	0.18	0.18
		0.001	10800	*	*	*	10800	167.9	77.43	71.41	2616.9	39930.0	0.24	0.24
		0.002	10800	*	*	*	10800	144.9	76.06	61.74	1820.0	40743.6	0.49	0.49

Table 3.2 Summary results of solving the MINLP, the MINLP relaxation and the Lagrangian relaxation with fixed-NLP for fixed capacity with equal weights case: geometric mean of CPU times in seconds, geometric mean of gap and geometric mean of number of nodes and iterations of MINLP solvers and subgradient algorithm, respectively. A '-' indicates gap is zero. A '*' means that no iteration was made and no upper bound was found. "Gap-A" is the relative gap between the lower and the upper bound obtained from the algorithm. "Gap-B" represents the relative gap between the lower bound of the method and the best known upper bound.

While MINLP did not solve instances as small as 10×10 , our method finds solutions with at most 1.64% gap to the optimal solution for instances as large as 100×100 , which was accomplished in less than 50 minutes. We also observe that the gap of MINLP relaxation for the instances solved to optimality is less than 2%, which implies that this relaxation provides strong lower bounds to the original bilevel program. However, upper bounds obtained from solutions of MINLP relaxation by solving fixed-NLP are weaker than upper bounds resulted from solving fixed-NLP with the solution of Lagrangian relaxation. The solutions obtained from the MINLP relaxation are better than those obtained from the Lagrangian relaxation in only four instances out of 36 total instances, with 0.44% geometric mean difference.

In the MINLP reformulation, relaxing the optimality requirement of the lower level problems leads to the MINLP relaxation. Therefore, the solution of the MINLP relaxation is the solution of a *centralized* decision making problem in which one decision maker sets decision for all parties involved in the problem. However, in reality customers make their decisions independently in a *decentralized* decision making problem. We compare the solutions of centralized and decentralized problems by first solving the relaxed MINLP (3.4). Let $(\hat{z}, \hat{y})$ denote the solution of the relaxation which is the solution of the centralized decision problem. Now, given $\hat{z}$ we optimize the lower level problems for each customer $i \in N$:

$$\begin{aligned} \min_{y_{ij}} \quad & \sum_{j \in N} c_{ij} y_{ij} + y_{ij} W(\sum_{j' \neq j} \hat{y}_{ij'} + y_{ij}) \\ \text{s.t.} \quad & \sum_{j \in M} y_{ij} = D_i \\ & 0 \leq y_{ij} \leq D_i \hat{z}_j \quad j \in M. \end{aligned} \tag{3.24}$$

Let $\bar{y}$ denote the solution. Table 3.3 shows the geometric mean and the largest value of $\|\hat{y} - \bar{y}\|_2$ for instances of the same size. However, since only instances of size 5 and 10 were solved within the time limit, we only include the difference of these two sizes. The relative gap between the largest value of differences in Table 3.3 with D_i , which is

a number in $(300, 500)$, is about 1.2%-2.1%. This small gap implies that the solution of MINLP-R approximates the solution of the bilevel program with small gaps.

$ N $	$ M $	κ	Δ	Max
5	5	0.0005	2.149	6.034
		0.001	2.307	6.273
		0.002	1.524	2.078
10	10	0.0005	3.593	3.972
		0.001	3.397	4.505
		0.002	4.277	5.376

Table 3.3 Summary results to show the difference between centralized and decentralized solutions: Δ represents the geometric mean of norm of difference between the $\hat{y}$ of MINLP relaxation with $\tilde{y}$ the solution of each customer optimization problem given $\hat{z}$, solution of MINLP relaxation. Max shows the largest of the norm of differences.

$ N $	$ M $	κ	MINLP				MINLP-R				Lagrangian dual + fixed-NLP			
			Time(s)	Node	% Gap-A	% Gap-B	Time(s)	Node	% Gap-A	% Gap-B	Time(s)	Iteration	% Gap-A	% Gap-B
5	5	0.0005	57.8	12166.0	-	-	0.3	54.7	-	-	54.9	28266.9	0.66	0.66
		0.001	163.7	28056.6	-	-	0.5	23.1	-	-	59.6	29254.5	1.79	1.59
		0.002	938.7	164398.3	-	-	0.4	18.5	-	-	53.4	26866.5	0.96	0.96
10	10	0.0005	10800	149560.5	49.53	48.90	12.3	214.6	-	-	10.6	29612.1	0.76	0.76
		0.001	10800	228155.7	65.73	63.69	5.1	224.9	-	-	11.4	30047.1	1.81	1.37
		0.002	10800	247964.2	76.20	74.80	4.3	161.8	-	-	10.2	28285.4	1.69	1.68
50	50	0.0005	10800	1048.1	67.61	62.45	10800	1953.9	67.24	67.80	363.7	27974.7	2.36	2.36
		0.001	10800	1023.6	76.55	73.16	10800	1046.3	64.37	64.91	390.2	29494.8	3.12	3.12
		0.002	10800	1418.3	84.13	81.24	10800	373.2	57.77	59.36	387.0	29552.7	1.56	1.56
100	100	0.0005	10800	*	*	*	10800	-	81.78	80.97	2853.8	31668.4	2.32	2.30
		0.001	10800	*	*	*	10800	-	78.93	76.01	2794.1	31246.1	1.09	1.09
		0.002	10800	*	*	*	10800	13.3	84.49	78.22	2866.9	32161.4	1.30	1.30

Table 3.4 Summary results of solving the MINLP, the MINLP relaxation and the Lagrangian relaxation with fixed-NLP for fixed capacity with unequal weights case: geometric mean of CPU times in seconds, geometric mean of gap and geometric mean of number of nodes and iterations of MINLP solvers and subgradient algorithm, respectively. A '-' indicates gap is zero. A '*' means that no iteration was made and no upper bound was found. "Gap-A" is the relative gap between the lower and the upper bound obtained from the algorithm. "Gap-B" represents the relative gap between the lower bound of the method and the best known upper bound.

Table 3.4 presents the same results of Table 3.2 for case of having weights on customers. These results show the strength of the solutions of the proposed Lagrangian relaxation, which have gaps between $0.14\% - 2.80\%$ to the upper bound. The MINLP relaxation performs very well for small instances with finding solutions equal to the upper bound, but fails to solve to optimality within 3 hours time limit for 50×50 and 100×100 instances. In contrast, the Lagrangian relaxation method could find a solution in less than 50 minutes for 100×100 instances. MINLP relaxation again appears to give relatively poor upper bounds (when the solutions are passed to fixed-NLP). In only four out of 36 instances the solutions of MINLP relaxation resulted in better upper bounds than that from the Lagrangian approach, and the geometric mean of relative gap was 0.42% .

3.6.3 Results for capacity size as a decision variable

In this section, we present the performance of the method for instances with variable capacity and equal weights. We solve the MINLP reformulation (3.15) with the general purpose global optimization solver BARON in GAMS. Because MINLP relaxation (3.16) is a nonconvex MINLP, we solve this problem with BARON as well. Finally, we report the results of solving instances with the Lagrangian relaxation approach in Algorithm 10 and subsequently solving the fixed-NLP (3.23) which is solved by `knitro`.

We applied the same analysis to find some values for parameter κ such that the average utility and the average number of open facilities fall in reasonable range. We choose values 0.0025, 0.004 and 0.007 which yield average utility in $(0.55, 0.85)$ and average number of open facilities in $(0.48, 0.9)$.

$ M $	$ N $	κ	KKT				MINLP-R				Lagrangian Relaxation			
			Time	Node	(%) Gap-A	(%) Gap-B	Time	Node	(%) Gap-A	(%) Gap-B	Time	Iteration	(%) Gap-A	(%) Gap-B
5	5	0.0005	10800	1037735.9	56.09	17.36	10800	3246841.2	52.17	28.18	85.9	7949.3	6.86	6.86
		0.001	10800	941220.3	65.14	43.15	10800	3369665.3	55.08	26.17	86.9	7834.9	6.40	6.40
		0.002	10800	1286100.0	74.05	48.15	10800	3976057.9	57.90	12.95	95.0	7913.8	6.11	6.11
10	10	0.0005	10800	259497.6	54.41	10.11	10800	208069.3	25.24	25.24	471.4	8201.4	6.30	6.30
		0.001	10800	427598.3	62.91	31.28	10800	826865.4	25.75	44.93	501.9	8183.5	6.89	6.89
		0.002	10800	570136.8	71.57	48.42	10800	687805.7	42.52	78.25	488.4	8142.5	6.76	6.76
50	50	0.0005	10800	61.7	*	*	10800	41563.3	13.21	23.98	2883.9	8523.9	7.82	7.82
		0.001	10800	89.1	*	*	10800	28496.1	36.73	51.62	2917.4	8459.6	7.40	7.40
		0.002	10800	89.2	*	*	10800	31921.2	50.05	61.23	2956.0	8467.6	7.31	7.31
100	100	0.0005	10800	2.6	*	*	10800	6137.6	14.36	27.01	6853.9	8647.3	8.58	8.58
		0.001	10800	3.0	*	*	10800	4923.1	35.51	50.32	6723.0	8627.0	8.85	8.85
		0.002	10800	2.62	*	*	10800	2042.6	74.93	49.76	7135.6	8626.6	9.37	9.37

Table 3.5 Summary results of solving the MINLP, the MINLP relaxation and the Lagrangian relaxation with fixed-NLP for variable capacity with equal weights case: geometric mean of CPU times in seconds, geometric mean of gap and geometric mean of number of nodes and iterations of MINLP solvers and subgradient algorithm, respectively. A '-' indicates gap is zero. A '*' means that no iteration was made and no upper bound was found. "Gap-A" is the relative gap between the lower and the upper bound obtained from the algorithm. "Gap-B" represents the relative gap between the lower bound of the method and the best known upper bound.

Table 3.5 shows the results of solving the MINLP reformulation, the MINLP relaxation and the Lagrangian relaxation approach. We include the geometric mean of CPU times, number of nodes or iteration, Gap-A and Gap-B as explained in the previous section. In these instances, neither of the MINLP formulations could be solved to optimality in the time limit. The Lagrangian relaxation approach yields smaller gaps with the upper bound obtained from fixed-NLP. For the larger instances with 50 or 100 facilities, the method requires more time compared to the time to solve fixed capacity instances. One potential reason could be that this method requires minimizing $g_j(U_j, Y_j)$ by a cutting plane algorithm, whereas when the capacity is fixed the method minimized $g_j(Y_j)$ by the bisection method.

3.7 Conclusion

We studied the problem of designing a distribution network in which customers choose which facilities to send their demand to. The goal is to optimize a hierarchical decision system in which a system operator chooses some facilities to open such that the cost of opening facilities is balanced with the quality of service to the customers. In the lower level of the decision system, customers choose open facilities to send their demand to such that the travel cost as well as the cost of congestion at facilities is minimized. Because the cost of congestion in a facility depends on all customers, the solution is an equilibrium to the system.

We first assumed that the capacity of each facility is given and fixed. In this case, we proposed a Lagrangian relaxation approach to find a lower bound for cases where customers can have equal or unequal weights. The method returns a solution which is used to find an upper bound to the true optimal value of the bilevel program. We conducted a set of experiments with instances of different sizes to evaluate the performance of the proposed method. The solutions resulted from the method turned out to be very close to the true

optimal solution while the standard approach of KKT reformulation could only be solved for instances as small as instances with 5 facilities.

In the proposed approach, the optimality requirement of customers is relaxed. The solution of this approach may be interpreted as the solution of a centralized decision problem in which decisions of supplier and customers are made in a single optimization problem. However, in reality each customer solves his/her own optimization problem independently which we call the decentralized decision problem. The results of comparing the norm of differences between these two solutions in the case of fixed capacity and equal weights show at most only 1%-3% gap. Therefore, one can conclude that the solution resulted from the MINLP relaxation very well satisfies the optimality requirement of customers' optimization problem as well.

We extended the approach to problems with variable capacity and equal weights. The subgradient approach can still solve the Lagrangian dual problem. However, we no longer apply the bisection algorithm to minimize the g_j function. Rather, we minimize it using a cutting plane approach. In this case, none of the MINLP reformulation or MINLP relaxation, even small instances with 5 facilities, was solved to optimality in time limit. However, the Lagrangian relaxation approach converged to solution with at most 9.5% gap to the optimal solution. The solution time in this approach is longer which might be because of the cutting plane approach to minimize g_j function.

We close the conclusion by an open question. The case of having variable capacities with equal weights was treated in this chapter. However, the more interesting case to have unequal weights is still open to be further explored. The reason being was that in order to have an efficient approach to minimize the function $g_j(U_j, Y_j)$, defined in (3.19), for unequal weights, we need to further study this function.

Chapter 4

A class of optimization problems with monotone nonconvex cost function

4.1 Introduction

In this chapter, we study a class of optimization problem that can be stated as follows:

$$\begin{aligned} \min \quad & c^T x + d^T z + Q(x) \\ & x \in X, \\ & (x, z) \in Y, \end{aligned} \tag{4.1}$$

where $Y \subset \mathbb{R}^n$, $X \subset \{0, 1\}^p$ and for some $p \leq n$, $c \in \mathbb{R}^p$, $d \in \mathbb{R}^{n-p}$ and the function $Q : \{0, 1\}^p \rightarrow \mathbb{R}$ is potentially nonconvex. This class, as an example, includes two-stage stochastic programs with integer second-stage variables. When all variables are continuous and all constraints are linear, to which we refer as two-stage stochastic *linear* program, the L-shaped method ([108], [20]) can efficiently solve the problem. This method is proved to converge because of the convexity of the value function of second-stage problem (see [82] and [21] for extensive details of the stochastic linear programs.)

Problem (4.1) can be reformulated to the following:

$$\begin{aligned} \min \quad & c^T x + d^T z + \theta \\ \text{s.t.} \quad & \theta \geq Q(x), \end{aligned} \tag{4.2a}$$

$$x \in X, \tag{4.2b}$$

$$(x, z) \in Y. \tag{4.2c}$$

We assume that $Q \geq L$ for some known value $L \in \mathbb{R}$ and define $Z := \{(x, z, \theta) : (4.2)\}$.

We also make the following assumption on the function Q .

Assumption 4.1. *We assume there exists a partial order $\preceq$ on X such that for all $x, x' \in X$ one has:*

$$x' \preceq x \Rightarrow Q(x) \leq Q(x').$$

Here we make no convexity or differentiability assumption on Q . As an example, in a two-stage stochastic program where $Q(x) = \sum_{k \in K} p_k q_k(x)$ and

$$\begin{aligned} q_k(x) = \min \quad & d^T y \\ \text{s.t.} \quad & W^k y \geq h^k - T^k x, \end{aligned} \tag{4.3}$$

if $T^k \geq 0, k \in K$, Assumption 4.1 holds with partial order $\preceq$ defined as the component-wise vector order.

We are interested in applications where Evaluating Q is computationally expensive. Hence, one goal is to solve (4.2) with a small number of evaluations of Q . An existing approach to solve this problem is the integer L-shaped method of Laporte and Louveaux [88]. The details will be explained in Section 4.3. To solve problem (4.2), their method relaxes constraint (4.2a) and add an inequality to approximate it. In this chapter, using monotonicity property of Q , we derive valid inequalities that give a better approximation of the relaxed inequality which results in tighter relaxation than integer L-shaped method. We apply an inequality which is similar to the inequality that Pan and Morton [98] use in a stochastic network interdiction problem to improve the approximation of the value function of the second stage problem. We extend this approach to any problem with monotone recourse function depending on binary first-stage variables.

The chapter is organized as follows. We review some related work in stochastic integer programming in Section 4.2. In Section 4.4, we first derive valid inequalities which attempt to use the monotonicity property of Q to strengthen the relaxation. We also show

two examples that satisfy Assumption 4.1. Then, an algorithm is presented to solve the problem. In Section 4.5, we conduct some experiments on two applications to compare the proposed methods with the existing methods. We then conclude the chapter in Section 4.6.

4.2 Related stochastic integer programming literature

In contrast to stochastic linear programming, two-stage stochastic mixed-integer programs, which arise in many applications, are more computationally difficult to solve than the two-stage stochastic linear program. The main challenge is that the value function is no longer guaranteed to be convex. In a special case when the first-stage variables are pure binary, Laporte and Louveaux [88] introduced the integer L-shaped method to approximate the value function of the second-stage problem (see details in Section 4.3.) An inequality is applied in a finitely convergent branch-and-bound algorithm to reduce the feasible space of first-stage variables.

For two-stage stochastic mixed-integer programming, some methods have been proposed. Carøe and Schultz [36] proposed scenario decomposition approach, known as *dual decomposition* approach. In a branch-and-bound, using scenario decomposition approach of Rockafellar and Wets [102], they obtain lower bounds by Lagrangian dual by dualizing non-anticipativity constraints. To solve the Lagrangian dual problem, the authors propose a subgradient algorithm which provides tight bounds. The subproblems solved in subgradient algorithm are mixed-integer programs because they include variables of both first and second stage problems, The Lagrangian problem and subgradient algorithm are all implemented in a branch-and-bound framework.

For a pure integer second-stage program with discrete distributions of stochastic parameters, Schultz et al. [106] proposed a finite method, which in a basic form is a complete enumeration. The method identifies a candidate set in the space of first-stage variables which contains the optimal solution. For each member of this set, the corresponding

second-stage integer programming problem has to be solved for all possible realizations of the uncertain parameter. The authors examined various ideas to reduce the number of elements in the candidate set.

Ahmed et al. [3] study a mixed-integer first-stage program with a pure integer second-stage problem. They propose a finite branch-and-bound algorithm which is more general than that of [106] because it avoids explicit enumeration of all possible discontinuous pieces of the value function of the second-stage problem. Using a variable transformation, a special structure of the discontinuities is obtained such that by a branching strategy all the discontinuity pieces are isolated and by a bounding strategy the exact representation of the value function of the second-stage problem is obtained.

Recently, Angulo et al. [8] improved the integer L-shaped method. To do this, they propose a technique to avoid exact evaluations of the second-stage problem, which is typically hard. One of the techniques they apply is to solve the LP relaxation of the second-stage problem and add the Benders' cut to approximate the value function. In addition, the authors discuss a framework that generates optimality cuts through a cut-generating linear program. This procedure considers information from all solutions found during the course of the algorithm.

4.3 Base methods

In this section, we review two basic approaches that we use to enhance our proposed method. The first approach is the Benders' decomposition which is guaranteed to find the optimal solution of a two-stage linear programming problem. The second approach, though, can be applied to two-stage integer programs with first-stage binary variables.

Suppose that a two-stage stochastic program is given. The first-stage problem is the following

$$\begin{aligned} \min \quad & c^T x + \theta \\ \text{s.t.} \quad & x \in X \\ & \theta \geq \mathcal{Q}(x), \end{aligned} \tag{4.4}$$

where $X \subset \mathbb{R}^n$ is a polyhedron, $\mathcal{Q}(x) = \sum_{k \in K} p_k q_k(x)$, K is the set of scenarios and p_k is the probability of scenario k . The second-stage problem for scenario k is as a linear program:

$$\begin{aligned} q_k(x) = \min \quad & d^T y \\ \text{s.t.} \quad & W^k y \geq h^k - T^k x, \end{aligned} \tag{4.5}$$

where $y \in \mathbb{R}^p$. Suppose that for $x \in X \subset \mathbb{R}^n$, problem (4.5) is feasible and π is the dual multiplier associated with the constraint of (4.5). One could show that

$$\mathcal{Q}(x) \geq \sum_{k \in K} p_k (\pi_k^T h - (T^T \pi_k)^T x),$$

is valid. Then,

$$\theta \geq \sum_{k \in K} p_k (\pi_k^T h - (T^T \pi_k)^T x), \tag{4.6}$$

is called the Benders' cut and valid for the feasible region of problem (4.4). The reason is that $\mathcal{Q}(x)$ for $x \in X$ is convex and π is a subgradient of $\mathcal{Q}$.

If, in (4.5), one requires $y \in \mathbb{R}^{p-r} \times \mathbb{Z}^r$. In this case, $\mathcal{Q}$ is no longer convex. Therefore, inequality (4.6) is not guaranteed to be valid. Suppose that $x \in \{0, 1\}^n$. In this case, Laporte and Louveaux [88] proposed a method to deal with case. For $(\bar{\theta}, \bar{x})$, the solution of master problem, the method evaluates $\mathcal{Q}(\bar{x})$. Then, the authors show that the inequality

$$\theta \geq (\mathcal{Q}(\bar{x}) - L) \left(\sum_{i: \bar{x}_i=0} x_i + \sum_{i: \bar{x}_i=1} (1 - x_i) \right) + \mathcal{Q}(\bar{x}) \tag{4.7}$$

is valid, and eliminates $\bar{x}$ if $\bar{\theta} < \mathcal{Q}(\bar{x})$. We refer to this method as the *standard integer L-shaped* (SILS) method.

4.4 Method

Suppose for a given $\bar{x}^l \in X$ we have evaluated $\mathcal{Q}(\bar{x}^l)$. By Assumption 4.1, for all $x \preceq \bar{x}^l$, one has $\mathcal{Q}(x) \geq \mathcal{Q}(\bar{x}^l)$. Now, we introduce a binary variable y_l such that

$$x \preceq \bar{x}^l \Rightarrow y_l = 1, \quad (4.8)$$

and

$$y_l = 1 \Rightarrow \theta \geq \mathcal{Q}(\bar{x}^l). \quad (4.9)$$

Statement (4.9) is formulated as follows

$$\theta \geq (\mathcal{Q}(\bar{x}^l) - L)y_l + L. \quad (4.10)$$

Here we show how to model statement (4.8) in two examples.

Example 4.2 (Component-wise order). *Suppose that the partial order $\preceq$ is defined as the component-wise order, i.e. $x' \preceq x$ means that $x'_i \leq x_i$ for $i = 1, \dots, n$. In this case, (4.8) is equivalent to*

$$y_l = 0 \Rightarrow \exists i : \quad x_i \geq \bar{x}_i^l + 1.$$

Therefore, for some j with $\bar{x}_j^l = 0$, we must have $x_j = 1$. This can be modeled as

$$\sum_{j: \bar{x}_j^l = 0} x_j \geq 1 - y_l. \quad (4.11)$$

If we eliminate y_l from (4.10), one has

$$\theta \geq (\mathcal{Q}(\bar{x}^l) - L)(1 - \sum_{j: \bar{x}_j^l = 0} x_j) + L. \quad (4.12)$$

Comparing to inequality (4.7), which is used in SILS, inequality (4.12) only includes variables with 0 values in $\bar{x}^l$. This implies that any solution that is eliminated by (4.7) is also eliminated by (4.12). Therefore, under the monotonicity assumption, inequality (4.12) yields a tighter approximation.

Example 4.3 (Multiple-choice models). Suppose for each $j = 1, \dots, m$, we want to choose one of the values $u_{jr}, r = 1, \dots, m_j$ where $u_{j1} \leq u_{j2} \leq \dots \leq u_{jm_j}$. Let's define a binary variable x_{jr} which equals to 1 if u_{jr} is chosen. We denote the vector $r(\bar{x}^l)$ as the vector of indexes $(r_1, r_2, \dots, r_{|M|})$ where $\bar{x}_{jr_j}^l = 1$ for $j = 1, \dots, |M|$. This problem can be modeled by

$$\sum_{r=1}^{m_j} x_{jr} = 1, \quad j = 1, \dots, |M|, \quad (4.13)$$

In order to model (4.8), we define the binary variable y_l which equals to 1 if $r(x) \leq r(\bar{x}^l)$. This can be modeled using the inequality

$$\sum_{j \in M} \sum_{r \leq r_j(\bar{x}^l)} x_{jr} \leq y_l + n - 1. \quad (4.14)$$

Now, by eliminating y_l from (4.10) and given (4.13), we have

$$\theta \geq (\mathcal{Q}(\bar{x}^l) - L)(1 - \sum_{j \in M} \sum_{r > r_j(\bar{x}^l)} x_{jr}) + L. \quad (4.15)$$

For a set of solutions $\bar{x}^l, l = 1, \dots, r$, we can now mix inequalities (4.10) to obtain a class of inequalities known as the *mixing inequalities* ([9, 69].) Given a solution $\bar{x}$, identifying a violated inequality in this class can be accomplished in polynomial time.

Theorem 4.4. Suppose we have a set of solutions $\bar{x}^l, l = 1, \dots, r$ for which we have the values $\mathcal{Q}(\bar{x}^l), l = 1, \dots, r$. Then for any subset $T \subset \{1, \dots, r\}$ with $T = \{\sigma(1), \dots, \sigma(|T|)\}$ such that $\mathcal{Q}(\bar{x}^{\sigma(1)}) \geq \mathcal{Q}(\bar{x}^{\sigma(2)}) \geq \dots \geq \mathcal{Q}(\bar{x}^{\sigma(|T|)})$, the inequality

$$\theta \geq \sum_{l=1}^{|T|} (\mathcal{Q}(\bar{x}^{\sigma(l)}) - \mathcal{Q}(\bar{x}^{\sigma(l+1)})) y_{\sigma(l)} + L, \quad (4.16)$$

is valid for X where $\mathcal{Q}(\bar{x}^{\sigma(|T|+1)}) = L$.

Note that we can also eliminate y_l variables in (4.16). For example, for a component-wise order, we apply

$$\theta \geq \sum_{l=1}^{|T|} (\mathcal{Q}(\bar{x}^{\sigma(l)}) - \mathcal{Q}(\bar{x}^{\sigma(l+1)})) (1 - \sum_{j: \bar{x}_j^{\sigma(l)} = 0} x_j) + L. \quad (4.17)$$

Likewise, in the multiple-choice order, one may use

$$\theta \geq \sum_{l=1}^{|T|} (\mathcal{Q}(\bar{x}^{\sigma(l)}) - \mathcal{Q}(\bar{x}^{\sigma(l+1)})) \left(1 - \sum_{j \in M} \sum_{r > r_j(\bar{x}^{\sigma(l)})} x_{jr}\right) + L. \quad (4.18)$$

Write $S := \{1, \dots, r\}$ to denote the set of indexes of integer solutions $\bar{x}$ found so far. These solutions might be the integer solutions found throughout the branch-and-bound algorithm or any other methods. Suppose that we have evaluated $\mathcal{Q}(\bar{x}^l), l \in S$. We define the set E as

$$E := \{x \in \{0, 1\}^p : (4.17) \text{ or } (4.18) \text{ is satisfied}, l \in S\}.$$

We consider the following master problem:

$$\min c^T x + d^T z + \theta \quad (4.19a)$$

$$x \in E, \quad (4.19b)$$

$$(\theta, x) \in F, \quad (4.19c)$$

$$(\theta, x) \in B, \quad (4.19d)$$

$$x \in X, \quad (4.19e)$$

$$(x, z) \in Y, \quad (4.19f)$$

where F is the set of points satisfying mixing inequalities (4.17) or (4.18), depending on what partial order we encounter in the problem, and B is the set of points that satisfy Benders inequality (4.6).

We solve problem (4.19) in a branch-and-bound tree by starting with $r := 0$, $S := \emptyset$ and $E := \{0, 1\}^p$. The advantage of this method is that rather than solving the master problem, which is an integer program and computationally expensive, to optimality iteratively and adding cuts in each iteration, we only solve the initial master problem once in a single tree and dynamically add the cuts as necessary. Algorithm 11 describes the steps in a node of the tree.

Input: (n_B^l, n_M^l, l)

Let $(\hat{\theta}^r, \hat{x}^r)$ denote the solution of LP relaxation in current node.

if $\hat{x} \in \{0, 1\}^p$ **then**

if $\hat{x}^r$ violates the Benders' cut (4.6) and $n_B^l < r_B^l$ **then**

 Update B accordingly.

$n_B^l \leftarrow n_B^l + 1$.

 Evaluate $Q(\hat{x})$.

if $\hat{\theta} < Q(\hat{x})$ **then**

 Update $S \leftarrow S \cap \{r\}$ and update E .

 Resolve LP relaxation in the current node and let $r \leftarrow r + 1$.

else

 Update the upper bound and prune the node.

end

else

if $\hat{x}^r$ violates the Benders' cut (4.6) and $n_B^l < r_B^l$ **then**

 Update B accordingly.

$n_B^l \leftarrow n_B^l + 1$.

if $\hat{x}^r$ violates any mixing inequality and $n_M^l < r_M^l$ **then**

 Find the most violated mixing inequality and update F accordingly.

$n_M^l \leftarrow n_M^l + 1$.

if B or F was changed **then**

 Resolve LP relaxation in the current node and let $r \leftarrow r + 1$.

else

 Do branching.

end

end

Algorithm 11: Steps of the method in a node. r_B^l and r_M^l are the number of Benders and mixing inequalities allowed to be added in node l , respectively. We set $r_B^0 = r_M^0 = \infty$

As an enhancement we generate extra solutions to potentially make the mixing inequalities more violated, and hence more influential. To do so, we first want to mention that the SGAP is a special case of SCPP when $|M_j| = 2, j \in M$. Therefore, the idea could be immediately employed for SGAP. We first observe that from inequality (4.14) and binary restriction of y^l , one has

$$\bar{y}^l = \max\{0, 1 - \sum_{j \in M} \sum_{r > r_j(\bar{x}^l)} x_{jr}\}.$$

Therefore, if we add extra solutions $\bar{x}^l$, such that their associated $\bar{y}^l$ evaluated at the current LP relaxation solution is closest to 1, then the right-hand side of the mixing inequality would be potentially larger, hence the inequality would be more violated, which is advantageous. We define the following index sets for facility j :

$$I_j^F(\bar{x}) = \{r \in M_j : \bar{x}_{jr} \in (0, 1)\},$$

which is the indexes of fractional solutions, and

$$I_j^1(\bar{x}) = \{r \in M_j : \bar{x}_{jr} = 1\}.$$

Note that either $F_j = \emptyset$ or $S_j = \emptyset$. We initialize an index set R as

$$R(\bar{x}) = \Pi_{j \in M: I_j^F(\bar{x}) \neq \emptyset} I_j^F(\bar{x}).$$

Then for any $(r_{f_1}, r_{f_2}, \dots, r_{f_q}) \in R(\bar{x})$, a new solution $\hat{x}$ is generated by letting $x_{jr} = 1$ for all $r \in I_j^1(\bar{x})$ when $I_j^1(\bar{x}) \neq \emptyset$, $x_{f_i, r_{f_i}} = 1, i = 1, \dots, q$, and $x_{jr} = 0$ otherwise. Now, if

$$\sum_{j \in M} \sum_{r > r_j(\hat{x}^l)} \bar{x}_{jr} < 1,$$

then $\hat{x}$ is acceptable to be added into the pool of extra solutions. We then evaluate $\mathcal{Q}(\hat{x})$ and consider it in separating $\bar{x}$ by mixing inequalities.

This enhancement does the best to strengthen mixing inequalities by providing more solutions. This is a computationally expensive approach because it requires evaluating

$\mathcal{Q}$ for all new solutions. However, we start with this approach in order to find the best possible impact the mixing inequalities might have on tightening the LP relaxation. If mixing inequalities are not effective at closing the gap with this setting, we can conclude that their use is not promising. Otherwise, we would seek a more moderate approach for identifying additional solutions. Unfortunately, as we will see in the next section, even this most aggressive setting does not demonstrate significant benefit from using the mixing inequalities.

4.5 Numerical Results

The goal of the computational experiments is to examine the performance of the method proposed in Section 4.4. To do so, we implemented four methods. As the base method, we implemented the Standard Integer L-Shaped (SILS) method of Laporte and Louveaux [88] which was discussed in Section 4.3. The second method is the proposed method in Section 4.4 where we only used inequality (4.12) or (4.15) to eliminate an integer solution $\bar{x}^l$. This method is referred to as *enhanced integer L-shaped* (EILS) method. In the third experiment, we have EILS settings plus mixing inequalities (4.17) or (4.18) and only look at the gap closed at the root node. If the root gaps closed in the third experiment show no substantial improvement with mixing inequalities, we immediately conclude that these inequalities may not be so effective in this class of problems. However, in our results we observed that the root gap closed experienced a noticeable reduction. This reduction may be because of the mixing inequalities, or because of the extra EILS cuts added at the extra points. We conducted the fourth experiment which investigates whether the extra EILS cuts have caused the gap reduction or not. To do this, we run EILS method with extra solution generations and add all relevant EILS cuts that are violated.

We run the experiments on two selected applications: *Stochastic Generalized Assignment Problem* (SGAP), where the component-wise ordering holds, and *Stochastic Capacity Planning Problem* (SCPP), where the multiple choice ordering holds.

All the methods were implemented in C++ and all MILPs were solved by Cplex 12.5.

4.5.1 Applications

In example 4.2, we presented the component-wise order and how we can model statement (4.8). The SGAP which has the component-wise ordering structure is as follows. For a set of scenarios $k = 1, \dots, K$ with associated probabilities p_k , suppose the first-stage problem is

$$\begin{aligned} \min \quad & \sum_{j \in M} c_j x_j + \mathcal{Q}(x) \\ \text{s.t.} \quad & x_j \in \{0, 1\}, \quad j \in M. \end{aligned}$$

where c_j is the cost of opening facility j , and $\mathcal{Q}(x) = \sum_{k=1}^K p_k Q_k(x)$. For each scenario k , the second-stage problem is

$$Q_k(x) = \min \quad \sum_{i \in N} f_i s_i + \sum_{i \in N} \sum_{j \in M} h_{ijk} y_{ij} \quad (4.20a)$$

$$\text{s.t.} \quad 0 \leq y_{ij} \leq x_j, \quad i \in N, j \in N, \quad (4.20b)$$

$$\sum_{i \in N} d_{ik} y_{ij} \leq a_j x_j, \quad j \in M \quad (4.20c)$$

$$\sum_{j \in M} y_{ij} + s_i = 1, \quad i \in N \quad (4.20d)$$

$$y_{ij} \in \{0, 1\}, \quad i \in N, j \in M, \quad (4.20e)$$

where d_{ik} is the demand of customer i in scenario k , a_j is the capacity of facility j , f_i is the cost of not assigning customer i to any facility and h_{ijk} is the cost of assigning customer i to facility j in scenario k . Variable y_{ij} equals 1 if customer i is assigned to facility j ,

and 0 otherwise. Also, if customer i is not assigned to any facilities variable s_i becomes equal to 1. In the objective function (4.20a), the first and second terms capture the cost of not assigning customer i to any facility and cost of assigning customer i to facility j . Constraint (4.20b) ensures that customer i can be assigned to facility j if facility j is open. To prevent assigning total demands below the capacity of facility j , constraint (4.20c) is enforced. In constraint (4.20d) customer i is either assigned to a facility, or s_i becomes equal to 1. Note that $Q_k(\cdot)$ and $Q(\cdot)$ are monotone decreasing. We test the proposed method on this example. We generate random instances based on [65]

The second application is the capacity planning problem in which the multiple choice ordering assumption holds. Suppose a set of facilities $M = \{1, \dots, m\}$ is given each of which is associated with a set of given capacities $\{u_{j1}, \dots, u_{jm_j}\}$, with $u_{j1} \leq u_{j2} \leq \dots \leq u_{jm_j}$. The key decision in this model is to determine the capacity for each facility. We write M_j to denote the set of indexes of capacities for facility $j \in M$.

The problem we are interested in solving is

$$\min \sum_{j \in M} \sum_{r \in M_j} c_{jr} x_{jr} + Q(x) \quad (4.21a)$$

$$\text{s.t. } \sum_{r \in M_j} x_{jr} = 1, \quad j \in M \quad (4.21b)$$

$$x_{jr} \in \{0, 1\}, \quad r \in M_j, j \in M. \quad (4.21c)$$

Here c_{jr} represents the cost for setting capacity of facility j to u_{jr} and constraint (4.21b) ensures that for each facility j exactly one capacity is chosen.

We define the cost function as $\mathcal{Q}(x) = \sum_{k=1}^K p_k Q_k(x)$ where Q_k is the cost of second-stage problem in scenario k . The second-stage problem for scenario k is:

$$Q_k(x) = \min \sum_{i \in N} f_i s_i + \sum_{i \in N} \sum_{j \in M} h_{ijk} y_{ij} \quad (4.22a)$$

$$\text{s.t.} \sum_{i \in N} d_{ik} y_{ij} \leq \sum_{r \in M_j} u_{jr} x_{jr}, \quad j \in M \quad (4.22b)$$

$$\sum_{j \in M} y_{ij} + s_i = 1, \quad i \in N \quad (4.22c)$$

$$y_{ij} \in \{0, 1\}, \quad i \in N, j \in M, \quad (4.22d)$$

The description of the objective function and constraints is the same as those of problem (4.20) except that constraint (4.22b) ensures that chosen level of capacity of facility j is not exceeded. Here, constraint (4.20b) is also implied by (4.22b). Since SCPP is a generalization of SGAP, we again use the approach of [65] to generate instances. The only difference is that we need to generate levels of capacity. For facility $j \in M$, we just equally split $(0, a_j)$ into $|M_j|$ subintervals where a_j is the capacity of facility j .

4.5.2 Results

For SGAP instances, we generated three instances each for sizes (10, 30), (20, 40), (30, 50) and (40, 100), where size is given as $(|M|, |N|)$ with $|M|$ denoting the number of facilities and $|N|$ denoting the number of customers. Each instance has 1000 scenarios. For SCPP instances, we chose the following sizes: (3, 20), (4, 20), (5, 25), (8, 25), (10, 40) and (15, 50) each with 25, 50, 100 and 1500 possible levels of capacity. For each size, we generated three instances, each with 500 scenarios.

The first question we investigate is the impact of mixing inequalities on the root relaxation. To see this, we compare the results of processing the root node with all methods. For SGAP and SCPP instances, respectively, Tables 4.1 and 4.2 display CPU time in seconds, gap of LP relaxation to the best upper bound, number of integer L-shaped cuts (either

standard or enhanced version) added and number of Benders' cuts added. Other than SILS and EILS, the tables have two more columns, EILS+AP and EILS+Mixing. EILS+AP displays results of the EILS method with generating additional points and adding only relevant EILS cuts at those points whereas in the column labeled as EILS+Mixing we also add violated mixing inequalities. The tables show that for larger instances the root gap closed by EILS is smaller than that of SILS. Also, for larger instances EILS method adds more EILS cuts than SILS. Comparing EILS with EILS+Mixing, the gaps are consistently smaller when we add mixing inequalities. However, this reduction is apparently the result of the EILS cuts added at the additional points generated to strengthen the mixing inequalities. This can be observed from comparing the gap between EILS+Mixing and EILS+AP. The gaps of EILS+AP are very close to those of EILS+Mixing. This implies that mixing inequalities are not so effective at closing the root gap. Rather, the reduced gaps obtained in the EILS+Mixing setting are due to the EILS cuts added at the additional points generated. Because the time spent generating the additional EILS cuts is so large, and the gap reduction is not significant, the use of mixing inequalities does not appear to be a promising option. Therefore, in the sequel we only compare the use of SILS and EILS cuts for solving these problems to optimality.

$ M $	$ N $	SILS				EILS				EILS + AP				EILS+Mixing				
		Time	% G	ILS	Bend.	Time	% G	ILS	Bend.	Time	% G	ILS	Bend.	Time	% G	# Mix.	ILS	Bend.
10	30	32.3	21.66	6.5	52.9	132.6	22.21	34.4	163.1	6807.8	18.63	35.0	164.2	5747.0	19.23	8.6	28.5	147.8
20	40	84.1	8.06	10.1	63.9	183.6	10.96	30.6	150.2	61671.2	6.58	17.2	14.8	11348.4	6.42	5.4	15.6	15.3
30	50	10.0	8.40	12.5	75.3	76.2	5.61	10.0	12.0	23297.5	5.83	9.5	20.9	12655.4	5.88	7.3	9.5	11.5
40	100	21.3	7.39	13.3	79.9	173.3	4.99	9.7	11.7	67526.2	4.97	9.6	21.3	31126.4	5.00	10.2	10.1	12.4

Table 4.1 Time, gap, number of ILS cuts, number of mixing cuts and number of Benders' cuts to process root node of SGAP instances.

M	N	L	SILS				EILS				EILS + AP				EILS+Mixing				
			Time	% G	ILS	Bend.	Time	% G	ILS	Bend.	Time	% G	ILS	Bend.	Time	% G	# Mix	ILS	Bend.
3	20	25	17.7	3.35	6.5	13.3	5.1	12.98	5.9	9.6	15.1	12.98	5.9	9.6	12.6	12.93	2.0	5.9	10.0
		50	20.4	2.31	8.7	15.0	3.0	17.63	5.0	7.0	4.6	17.63	5.0	7.0	3.3	17.63	0.0	5.0	7.0
		1500	21.4	2.15	6.5	10.0	14.8	4.37	6.8	12.1	63.6	4.37	6.8	12.1	93.7	4.28	5.7	9.8	12.1
4	20	25	19.3	7.76	8.1	22.8	14.5	13.63	10.9	25.4	89.1	13.83	11.0	24.5	78.0	13.81	4.9	10.6	23.7
		50	29.4	8.18	9.5	28.2	20.2	16.47	11.4	25.0	349.6	7.94	14.4	43.0	75.0	15.65	3.9	11.0	24.8
		1500	41.1	7.06	9.6	21.2	45.2	7.01	13.8	40.0	428.1	7.06	14.1	41.6	397.8	7.06	8.5	14.5	41.0
5	20	25	94.6	10.25	22.1	73.9	157.7	8.92	44.9	152.7	1789.0	7.85	35.0	144.3	2168.0	7.82	6.2	37.6	145.1
		50	83.2	11.13	21.0	67.6	134.2	11.43	38.9	130.1	1645.4	8.34	32.4	126.7	1947.8	8.52	10.8	36.6	132.2
		100	80.9	9.44	18.7	60.0	135.9	9.88	37.3	115.8	1632.8	8.47	29.9	111.1	1867.9	8.47	8.6	31.4	114.9
		1500	88.0	12.98	8.3	43.1	99.9	13.01	13.3	86.5	1488.0	8.64	17.3	87.8	1632.4	8.25	13.3	16.9	85.8
8	25	25	110.9	34.10	19.0	112.7	172.8	29.99	43.0	234.6	4042.9	13.15	67.1	261.0	5951.7	14.13	21.3	39.4	226.2
		50	94.5	31.73	39.8	133.6	176.3	24.36	89.9	282.5	4805.7	13.18	79.1	267.6	5752.6	13.76	19.7	81.9	290.1
		100	106.4	32.96	39.9	128.5	190.7	30.93	80.9	263.0	4923.1	14.15	59.3	223.5	4282.5	13.95	21.1	62.7	239.6
		1500	185.1	38.38	4.9	82.1	200.4	38.36	10.9	167.9	5360.0	14.83	10.6	147.9	7839.2	14.77	21.9	11.3	155.4
10	40	25	177.3	14.88	39.2	163.9	314.8	14.86	87.3	341.9	6342.1	10.55	49.4	308.2	38082.6	10.24	17.3	53.1	314.2
		50	207.7	21.03	48.5	166.0	334.4	14.60	96.5	337.7	11732.0	10.90	68.0	248.0	13096.9	10.81	11.2	56.0	240.2
		100	308.8	19.21	67.0	188.4	462.3	17.51	113.7	349.2	7520.5	14.01	58.2	202.8	7227.3	15.51	20.9	47.3	181.5
		1500	419.8	36.94	4.0	94.2	517.7	35.20	12.5	188.5	7354.7	15.91	11.8	100.3	8209.2	14.57	17.7	11.8	126.8
15	50	25	118.1	24.41	23.1	122.1	343.7	16.98	111.9	349.3	8653.2	10.23	65.0	215.3	34182.0	9.90	19.0	65.0	241.0
		50	141.6	22.40	27.7	130.3	370.2	16.49	115.3	349.5	10560.0	10.97	56.0	176.0	9564.4	11.29	13.4	47.8	163.0
		100	168.3	15.65	30.6	134.6	364.8	16.03	110.4	321.0	7845.1	13.22	42.5	135.3	8108.4	12.76	11.9	44.2	147.6
		1500	321.6	57.94	3.0	65.4	368.7	52.92	9.7	135.6	7400.0	15.40	9.7	78.1	8298.8	11.89	11.5	9.7	98.3

Table 4.2 Summary results of the root node solves for SILS, EILS, EILS+Mixing and EILS+AP (EILS with additional points)

We present the results of the branch-and-cut method to solve the instances in one hour time limit in Table 4.3 and Table 4.4 for the SGAP and SCPP instances, respectively. Each table shows the number of nodes, time in seconds, and gap between lower and upper bound for SILS, EILS and the extensive formulation. The time limit is one hour. As we expected, EILS outperforms SILS as its gap is consistently smaller than the gap of SILS. EILS could terminate in optimality for three sets of instances whereas SILS only solved one of these sets. In the instance set that both solved to optimality, EILS required fewer number of nodes than SILS. The extensive formulation solved could not solve any of the instances to optimality. However, in some of the instances, its associated gap is smaller than both SILS and EILS. For these instances, it appears that the reduction in size by using a decomposition algorithm does not compensate for the loss of structure of the full extensive form, which the solver can take advantage of to yield stronger relaxations.

$ M $	$ N $	SILS			EILS			Extensive		
		Node	Time	% G	Node	Time	% G	Node	Time	% G
10	30	154.8	1091.2	0.00	317.8	930.1	0.00	172.5	70.6	0.00
20	40	1070.5	2431.6	6.05	889.6	2534.2	9.64	18127.0	2750.1	7.54
30	50	215.4	94.3	0.00	194.3	75.4	0.00	2564.3	1198.2	0.00
40	100	563.1	312.5	0.00	574.9	172.5	0.00	1312.2	2420.0	0.00

Table 4.3 Summary results of branch-and-bound for SGA including number of nodes, CPU time in seconds, % gap between lower and the best upper bound for one hour time limit of SILS, EILS and extensive formulation.

M	N	L	SILS			EILS			Extensive		
			Node	Time	% Gap	Node	Time	% Gap	Node	Time	% Gap
3	20	25	646.1	992.5	0.00	150.8	97.7	0.00	24185.1	3600.0	7.46
		50	2247.3	3600.0	1.92	529.0	575.0	0.00	27662.9	3600.0	10.22
		1500	2793.0	3600.0	2.15	457.4	3600.0	2.13	3597.1	3600.0	35.38
4	20	25	2955.9	3600.0	6.67	862.3	1421.0	0.00	3414.4	3600.0	5.08
		50	3124.0	3600.0	7.97	500.5	2580.2	3.84	3178.4	3600.0	4.44
		1500	2740.5	3600.0	7.06	1766.6	3600.0	7.01	2216.6	3600.0	41.63
5	25	25	2736.1	3600.0	8.90	3554.6	3600.0	8.33	809.2	3600.0	7.38
		50	2222.7	3600.0	9.39	6430.1	3600.0	9.37	625.1	3600.0	8.22
		100	2251.0	3600.0	9.02	3960.0	3600.0	8.74	941.7	3600.0	11.00
		1500	1565.9	3600.0	12.72	1782.3	3600.0	9.96	821.3	3600.0	17.34
8	25	25	9654.4	3600.0	27.84	7221.7	3600.0	19.47	697.4	3600.0	11.59
		50	9238.8	3600.0	20.58	7019.7	3600.0	16.43	584.2	3600.0	11.55
		100	6593.3	3600.0	27.30	8021.1	3600.0	23.08	605.0	3600.0	11.32
		1500	1340.1	3600.0	38.38	1360.0	3600.0	34.99	2622.1	3600.0	27.21
10	40	25	5642.3	3600.0	14.85	5089.9	3600.0	12.29	59.5	3600.0	11.77
		50	4735.7	3600.0	15.42	4349.2	3600.0	13.12	53.9	3600.0	13.49
		100	5363.0	3600.0	19.21	3634.0	3600.0	15.54	34.9	3600.0	15.36
		1500	849.5	3600.0	24.75	811.3	3600.0	35.20	968.3	3600.0	34.26
15	50	25	3674.1	3600.0	19.34	5359.6	3600.0	13.86	466.0	3600.0	14.09
		50	7138.5	3600.0	15.50	4490.3	3600.0	13.60	303.5	3600.0	12.92
		100	5729.2	3600.0	15.65	4703.6	3600.0	13.90	64.1	3600.0	16.75
		1500	573.3	3600.0	56.40	537.9	3600.0	50.92	1272.0	3600.0	27.66

Table 4.4 Summary results of branch-and-bound for SCPP including number of nodes, CPU time in seconds, % gap between lower and the best upper bound for one hour time limit of SILS, EILS and extensive formulation.

4.6 Conclusion

We studied a class of optimization problems with a nonconvex monotone function of binary variables in the objective function. Many two-stage integer programs, that are

typically very hard, fall into this class. Because the value function of the subproblems are no longer a convex function, standard approaches to solve two-stage problems, such as Benders' decomposition, fail to converge to the global optimal solution.

By exploiting the structure of the problem, we derived two valid inequalities to tighten the LP relaxation in the branch-and-bound approach. We compared the results of the proposed method against the existing standard integer L-Shaped method. In the first step, we just compared standard integer L-shaped method with our method which uses an enhanced version of the integer L-shaped method. The gap and solution time of the enhanced method show the domination of the enhanced version.

To further improve the method, we proposed to also add the *mixing inequalities*. For this one, we generated some additional points to elevate the mixing inequalities. We generated the points very aggressively such that all possible solutions that may have any impact on the mixing inequality for that particular LP relaxation solution were considered. This way, the method is very computationally expensive. Results showed that the root gap closed with mixing inequalities in small sizes is not significant. For larger sizes, the gaps of mixing inequalities have some meaningful differences. This difference may be because of two sources: mixing inequalities, or integer L-shaped cuts added at additional points. We conducted another experiment to identify which one of these sources was the real factor. In this experiment, we generated all possible solutions, and only added integer L-shaped cuts with no mixing inequalities. The results showed that the gap was close to the case where we had mixing inequalities. This suggests that mixing inequalities was not the real source of improving the gap.

LIST OF REFERENCES

- [1] K. Abhishek, S. Leyffer, and J. Linderoth. FilMINT: An outer-approximation-based solver for convex mixed-integer nonlinear programs. *INFORMS J. Comput.*, 22(4):555–567, 2010.
- [2] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42 – 54, 2005.
- [3] S. Ahmed, M. Tawarmalani, and N.V. Sahinidis. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Math. Program.*, 100(2):355–377, 2004.
- [4] E. Aiyoshi and A. Shimizu. Hierarchical decentralized systems and its new solution by a barrier method. *Sys. Man. Cyber. IEEE Trans.*, 11(6):444–449, June 1981.
- [5] E. Aiyoshi and K. Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *Auto. Control IEEE Trans.*, 29(12):1111–1114, Dec 1984.
- [6] F. Al-Khayyal, R. Horst, and P.M. Pardalos. Global optimization of concave functions subject to quadratic constraints: An application in nonlinear bilevel programming. *Ann. Oper. Res.*, 34(1):125–147, 1992.
- [7] I. P. Androulakis, C. D. Maranas, and C. A. Floudas. α BB : A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995.
- [8] G. Angulo, S. Ahmed, and S. Dey. Improving the integer l-shaped method, 2014.
- [9] A. Atamtürk, G. Nemhauser, and M. Savelsbergh. The mixed vertex packing problem. *Math. Program.*, 89:35–53, 2000.
- [10] J. Bard and J. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM J. Sci. Stat. Comp.*, 11(2):281–292, 1990.

- [11] J.F. Bard. Coordination of a multidivisional organization through two levels of management. *Omega*, 11(5):457 – 468, 1983.
- [12] J.F. Bard and J. Moore. An algorithm for the discrete bilevel programming problem. *Naval Res. Logist.*, 39:419–435, 1992.
- [13] M. BartholomewBiggs. One-variable optimization. In *Nonlinear Optimization with Engineering Applications*, volume 19 of *Springer Optimization and Its Applications*, pages 1–22. Springer US, 2008.
- [14] M.S. Bazaraa, H. Sherali, and C.M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, 2006.
- [15] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 5 2013.
- [16] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Endlewood Cliffs, NJ, 1987.
- [17] D. Bertsekas, A. Nedić, and A.E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.
- [18] W.F. Bialas, M.H. Karawa, and J. Shaw. A parametric complementarity pivot approach for two-level linear programming. Technical report, State University of New York at Buffalo, Operations Research Program, 1980.
- [19] W.F. Bialas and M.H. Karwan. Two-level linear programming. *Manag. Sci.*, 30(8):1004–1020, 1984.
- [20] J. Birge and F. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988.
- [21] J. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer, 2011.
- [22] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discret. Optim.*, 5:186 – 204, 2008.
- [23] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Math. Program.*, 119:331–352, 2009.
- [24] P. Bonami, M. Kiliç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In Jon Lee and Sven Leyffer, editors, *IMA Volumes in Mathematics and its Applications*, volume 154, pages 1 – 40. Berlin: Springer, 2012.

- [25] P. Bonami, J. Lee, S. Leyffer, and A. Wächter. More branch-and-bound experiments in convex nonlinear integer programming. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2011. Preprint ANL/MCS-P1949-0911.
- [26] R. Boorstyn and H. Frank. Large-scale network topological optimization. *IEEE Trans. Commun.*, 25:29–47, 1977.
- [27] B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programs. *Comput. Oper. Res.*, 21:359–368, 1994.
- [28] J. Bracken and J.T. McGill. Mathematical programs with optimization problems in the constraints. *Oper. Res.*, 21(1):pp. 37–44, 1973.
- [29] J. Bracken and J.T. McGill. Defense applications of mathematical programs with optimization problems in the constraints. *Oper. Res.*, 22(5):pp. 1086–1096, 1974.
- [30] J. Bracken and J.T. McGill. Production and marketing decisions with multiple objectives in a competitive environment. *J. Optim. Theory Appl.*, 24(3):449–458, 1978.
- [31] L. Brotcome, M. Labbe, P. Marcotte, and G. Savard. A bilevel model and solution algorithm for a freight tariff setting problem. *Transport. Sci.*, 44:289–302, 2000.
- [32] S. Burer and A. N. Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Sur. Oper. Res. Manag. Sci.*, 17:97 – 106, 2012.
- [33] M. R. Bussieck and A. Drud. SBB: A new solver for mixed integer nonlinear programming. In *OR 2001, Section: Continuous Optimization*, 2001.
- [34] M. R. Bussieck, A. S. Drud, and A. Meeraus. MINLPLib – a collection of test models for mixed-integer nonlinear programming. *INFORMS J. Comput.*, 15(1), 2003.
- [35] W. Candler and R. Townsley. A linear two-level programming problem. *Comput. Oper. Res.*, 9(1):59 – 76, 1982.
- [36] C.C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Oper. Res. Lett.*, 24:37–45, 1999.
- [37] I. Castillo, J. Westerlund, S. Emet, and T. Westerlund. Optimization of block layout design problems with unequal areas: A comparison of milp and minlp optimization methods. *Comput. Chem. Eng.*, 30:54–69, 2005.

- [38] CMU/IBM MINLP Project. <http://egon.cheme.cmu.edu/ibm/page.htm>.
- [39] B. Colson, P. Marcotte, and G. Savard. A trust-region method for nonlinear bilevel programming: Algorithm and computational experience. *Comput. Optim. Appl.*, 30(3):211–227, 2005.
- [40] J. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4OR*, 3(2):87–107, 2005.
- [41] I. Constantin and M. Florian. Optimizing frequencies in a transit network: a non-linear bi-level programming approach. *Internat. Trans. Oper. Res.*, 2(2):149 – 164, 1995.
- [42] K. Cormican, D. Morton, and R. Wood. Stochastic network interdiction. *Oper. Res.*, 46(2):184–197, 1998.
- [43] J.-P. Cote, P. Marcotte, and G. Savard. A bilevel modelling approach to pricing and fare optimization in the airline industry. *J. Rev. Pric. Manag.*, 2(1):23–36, 2003.
- [44] CPLEX Optimization, Inc., Incline Village, NV. *Using the CPLEX Callable Library, Version 9*, 2005.
- [45] R. J. Dakin. A tree search algorithm for mixed integer programming problems. *Comput. J.*, 8:250–255, 1965.
- [46] S.T. DeNegre and T.K. Ralphs. A branch-and-cut algorithm for integer bilevel linear programs. *Oper. Res. Cyber-Infras.*, 47:65–78, 2009.
- [47] X. Deng. Complexity issues in bilevel linear programming. In A. Migdalas, P.M. Pardalos, and P. Varbrand, editors, *Multilevel Optimization: Algorithms and Applications*, volume 20 of *Nonconvex Optimization and Its Applications*, pages 149–164. Springer US, 1998.
- [48] E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91:201–212, 2002.
- [49] Marco A. Duran and Ignacio E. Grossman. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Program.*, 36:307–339, 1986.
- [50] T.A. Edmunds and J.F. Bard. Algorithms for nonlinear bilevel mathematical programs. *Sys. Man. Cyber. IEEE Trans.*, 21(1):83–89, 1991.
- [51] S. Elhedhli. Service System Design with Immobile Servers, Stochastic Demand, and Congestion. *Manuf. Serv. Oper. Manag.*, 8(1):92–97, 2006.

- [52] J.E. Falk and J. Liu. On bilevel programming, part i: General nonlinear cases. *Math. Program.*, 70(1-3):47–72, 1995.
- [53] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Math. Program.*, 66:327–349, 1994.
- [54] R. Fletcher and S. Leyffer. User manual for filterSQP, 1998. University of Dundee Numerical Analysis Report NA-181.
- [55] A. Flores-Tlacuahuac and L. T. Biegler. Simultaneous mixed-integer dynamic optimization for integrated design and control. *Comput. Chem. Eng.*, 31:588–600, 2007.
- [56] C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications*. Kluwer Academic Publishers, 2000.
- [57] C.A. Floudas and P.M Pardalos. *Encyclopedia of Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.
- [58] J. Fortuny-Amat and B. McCarl. A representation and economic interpretation of a two-level programming problem. *J. Oper. Res. Soc.*, 32(9):783–792, 1981.
- [59] M. R. Garey and D. S. Johnson, editors. *Computers and Interactability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
- [60] A. Geoffrion. Generalized benders decomposition. *J. Optim. Theory Appl.*, 10(4):237–260, 1972.
- [61] P. M. Ghare, D. C. Montgomery, and W. C. Turner. Optimal interdiction policy for a flow network. *Naval Res. Logist.*, 18(1):37–45, 1971.
- [62] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002.
- [63] I. E. Grossmann and R. W. H. Sargent. Optimal design of multipurpose chemical plant. *Ind. Eng. Chem. Process Des. Dev.*, 18:343–348, 1979.
- [64] I.E. Grossmann, J. Viswanathan, A.V.R. Raman, and E. Kalvelagen. GAMS/DICOPT: A discrete continuous optimization package. *Math. Models Methods Appl. Sci.*, 11:649–664, 2001.
- [65] M. Guignard and M. Rosenwein. An improved dual based algorithm for the generalized assignment. *Oper. Res.*, 37:658–683, 1989.

- [66] O. Günlük, J. Lee, and R. Weismantel. MINLP strengthening for separable convex quadratic transportation-cost ufl. Technical report, IBM Research Division, 2007.
- [67] O. Günlük and J. Linderoth. Perspective relaxation of mixed integer nonlinear programs with indicator variables. *Mathematical Programming Series B*, 104:186–203, 2010.
- [68] O. Günlük and Jeff Linderoth. Perspective reformulation and applications. In Jon Lee and Sven Leyffer, editors, *IMA Volumes in Mathematics and its Applications*, volume 154, pages 61–92. Berlin: Springer, 2012.
- [69] O. Günlük and Y. Pochet. Mixing mixed-integer inequalities. *Math. Program.*, 90:429–457, 2001.
- [70] M. Guo, J. Zhu, and X. Zhao. A bi-level programming model for supplier selection in constructing logistics service supply chain. In *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1614–1618, Dec 2007.
- [71] M. Hamzeei and J. Luedtke. Linearization-based algorithms for mixed-integer nonlinear programs with convex continuous relaxation. *J. Glob. Opt.*, pages 1–23, 2014. <http://dx.doi.org/10.1007/s10898-014-0172-4>.
- [72] I. Harjunkski, R. Pörn, and T. Westerlund. MINLP: Trim-loss problem. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2190–2198. Springer, 2009.
- [73] H. Held and D.L. Woodruff. Heuristics for multi-stage interdiction of stochastic networks. *J. Heur.*, 11:483–500, 2003.
- [74] S. Huijun, G. Ziyu, and W. Jianjun. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Appl. Math. Mod.*, 32:610 – 616, 2008.
- [75] Yo Ishizuka and Eitaro Aiyoshi. Double penalty method for bilevel optimization problems. *Ann. Oper. Res.*, 34(1):73–88, 1992.
- [76] E. Israeli and R.K. Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.
- [77] V. Jain and I.E. Grossmann. Cyclic scheduling of continuous parallel-process units with decaying performance. *AIChE*, 44:1623–1636, 1998.

- [78] U. Janjarassuk and J. Linderoth. Reformulation and sampling to solve a stochastic network interdiction problem. *Networks*, 52(3):120–132, 2008.
- [79] J.J. Judice and A. Faustino. The solution of the linear bilevel programming problem by using the linear complementarity problem. *Investigacao Operacional*, 8:77–95, 1988.
- [80] J.J. Judice and A. Faustino. The linear-quadratic bilevel programming problem. *Info. Sys. Oper. Res.*, 32:87–98, 1994.
- [81] J.J. Judice and A.M. Faustino. A sequential lcp method for bilevel linear programming. *Ann. Oper. Res.*, 34(1):89–106, 1992.
- [82] P. Kall and W. Wallace. *Stochastic Programming*. John Wiley and Sons, Chichester, England, 1994.
- [83] A. Klose and A. Drexl. Facility location models for distribution system design. *European J. Oper. Res.*, 162:4–29, 2005.
- [84] S. Kongsomsaksakul, C. Yang, and A. Chen. Shelter location-allocation model for flood evacuation planning. *J. East. Asia Soc. Trasn. Stud.*, 6:4237 – 4252, 2005.
- [85] M. Labbe and P. Marcotte. A bilevel model of taxation and its application to optimal highway pricing. *Manag. Sci.*, 44:1608–1622, 1998.
- [86] M. Labbe and P. Marcotte. On a class of bilevel programs. In D. di Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 1–24. Kluwer Academic Publishers, 1998.
- [87] C. D. Laird, L. T. Biegler, and B. van Bloemen Waanders. A mixed integer approach for obtaining unique solutions in source inversion of drinking water networks. *J. of Water Resour. Plan. Manag.*, 132:242–251, 2006. Special Issue on Drinking Water Distribution Systems Security.
- [88] G. Laporte and F.V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Oper. Res. Lett.*, 13(3):133 – 142, 1993.
- [89] S. Leyffer. MacMINLP: Test problems for mixed integer nonlinear programming, 2003. <http://www.mcs.anl.gov/~leyffer/macminlp>.
- [90] Sven Leyffer. *User Manual for MINLP-BB*. University of Dundee, 1998.
- [91] A. Mahajan, S. Leyffer, and C. Kirches. Solving mixed-integer nonlinear programs by QP-diving. Technical Report Preprint ANL/MCS-2004-0112, Argonne National Laboratory, Mathematics and Computer Science Division, March 2012.

- [92] A. Mahajan, S. Leyffer, J. Linderoth, J. Luedtke, and T. Munson. MINOTAUR: A toolkit for solving mixed-integer nonlinear optimization. wiki-page, 2011. http://wiki.mcs.anl.gov/minotaur/index.php/Main_Page.
- [93] O. L. Mangasarian. Nonlinear Programming. In *Classics in Applied Mathematics*, number 10. Society for Industrial and Applied Mathematics, 1994. Originally published 1969 by McGraw-Hill, New York.
- [94] P. Marcotte and G. Savard. Bilevel programming: Algorithms. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 156–158. Springer US, 2001.
- [95] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.
- [96] A.W. McMasters and T.M. Mustin. Optimal interdiction of a supply network. *Naval Res. Logist.*, 17(3):261–268, 1970.
- [97] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2006.
- [98] F. Pan and D. Morton. Minimizing a stochastic maximum-reliability path. *Networks*, 52(3):111–119, 2008.
- [99] R. Pörn and T. Westerlund. A cutting plane method for minimizing pseudo-convex functions in the mixed integer case. *Comput. Chem. Eng.*, 24:2655–2665, 2000.
- [100] I. Quesada and I.E. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Comput. Chem. Eng.*, 16:937–947, 1992.
- [101] D. E. Ravemark and D. W. T. Rippin. Optimal design of a multi-product batch plant. *Comput. Chem. Eng.*, 22(1-2):177 – 183, 1998.
- [102] R.T. Rockafellar and R.J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.*, 16(1):119–147, 1991.
- [103] T. Roughgarden. The price of anarchy is independent of the network topology. *J. Comp. System. Sci.*, 67(2):341 – 364, 2003.
- [104] G.K. Saharidis and M.G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *J. Glob. Opt.*, 44(1):29–51, 2009.

- [105] N. Sawaya. *Reformulations, relaxations and cutting planes for generalized disjunctive programming*. PhD thesis, Chemical Engineering Department, Carnegie Mellon University, 2006.
- [106] R. Schultz, L. Stougie, and M. Vlerk. Solving stochastic programs with integer recourse by enumeration: A framework using gröbner basis. *Math. Program.*, 83:229–252, 1998.
- [107] H.D. Sherali and T.B. Carter. A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transpn. Res.-B*, 25B(6):439–452, 1991.
- [108] R. M. Van Slyke and Roger Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. App. Math.*, 17(4):pp. 638–663, 1969.
- [109] C. Still and T. Westerlund. Extended cutting plane algorithm. In C.A. Floudas and P.M Pardalos, editors, *Encyclopedia of Optimization*, pages 53–61. Kluwer Academic Publishers, Dordrecht, 2001.
- [110] E. Taniguchi, M. Noritake, T. Yamada, and T. Izumitani. Optimal size and location planning of public logistics terminals. *Transpn. Res.-E*, 35:207–222, 1999.
- [111] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Boston MA, 2002.
- [112] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.
- [113] M. Türkay and I. E. Grossmann. Logic-based MINLP algorithms for the optimal synthesis of process networks. *Comput. Chem. Eng.*, 20(8):959 – 978, 1996.
- [114] A. Vecchietti and I. E. Grossmann. LOGMIP: a disjunctive 0-1 non-linear optimizer for process system models. *Comput. Chem. Eng.*, 23(4-5):555 – 565, 1999.
- [115] L. Vicente, G. Savard, and J. Jdice. Descent approaches for quadratic bilevel programming. *J. Optim. Theory Appl.*, 81(2):379–399, 1994.
- [116] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Program.*, 106:25–27, 2006.

- [117] Tapio Westerlund and F. Pettersson. An extended cutting plane method for solving convex minlp problems. *Comp. Chem. Eng.*, 19:131–136, 1995.
- [118] R. Wollmer. Removing arcs from a network. *Oper. Res.*, 12(6):pp. 934–940, 1964.
- [119] L.A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.
- [120] R.K. Wood. Deterministic network interdiction. *Math. Comput. Model.*, 17(2):1–18, 1993.
- [121] Y. Zhang, O. Berman, P. Marcotte, and V. Verter. A bilevel model for preventive healthcare facility network design with congestion. *IIE Trans.*, 42:865 – 880, 2010.