EFFICIENT RESOURCE ALLOCATION IN A BILEVEL HIERARCHY WITH KNAPSACK AND ASSIGNMENT LOWER-LEVEL PROBLEMS

by

Behdad Beheshti

B.S., Sharif University of Technology, 2009

M.S., University of Pittsburgh, 2011

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Behdad Beheshti

It was defended on

November 6, 2014

and approved by

Oleg A. Prokopyev, Ph.D., Associate Professor, Department of Industrial Engineering

Andrew J. Schaefer, Ph.D., W.K. Whiteford Professor, Department of Industrial Engineering

Jayant Rajgopal, Ph.D., Professor, Department of Industrial Engineering

Osman Y. Özaltın, Ph.D., Assistant Professor, Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State University

Dissertation Director: Oleg A. Prokopyev, Ph.D., Associate Professor, Department of Industrial Engineering

EFFICIENT RESOURCE ALLOCATION IN A BILEVEL HIERARCHY WITH KNAPSACK AND ASSIGNMENT LOWER-LEVEL PROBLEMS

Behdad Beheshti, PhD

University of Pittsburgh, 2014

Bilevel optimization problems model a decision-making process with a two-level hierarchy of independent decision-makers, namely, the leader and the follower. The decisions are performed in a predetermined sequence with the leader acting first. Consequently, the follower solves an optimization problem which contains parameters (e.g., the right-hand sides of the follower's constraints) that are functionally dependent on the leader's decisions. On the other hand, the leader's objective and, possibly, constraints are also functions of both the leader's and follower's decision variables. Therefore, in the course of the decision-making process the leader should take into account the follower's rational response, i.e., optimal solutions to the follower's optimization problem.

This dissertation is focused on the development of exact solution approaches for bilevel programs with combinatorial structures in the lower-level problems. In particular, we consider models arising in resource distribution systems that involve bilevel decision-making hierarchies with knapsack and assignment constraints. We discuss design and implementation of novel solution techniques, which exploit structural properties of the underlying optimization problems. The superiority of the proposed approaches is demonstrated through extensive computational experiments.

Keywords: Operations research, bilevel programming, combinatorial optimization, exact solution approaches.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

to my parents, Susan and Jafar

This work would not have been possible without the guidance of Dr. Oleg Prokopyev, my mentor and advisor. I would like to thank him, first and foremost, for his constant support, encouragement and most importantly patience.

I would like to acknowledge Dr. Osman Özaltin, whose insights and collaborations contributed greatly to the success of this work. I would also like to thank the other members of my doctoral committee, Dr. Andrew Schaefer and Dr. Jayant Rajgopal, for their valuable suggestions and comments.

Finally, I am forever indebted to my parents, Susan and Jafar, for their endless love, inspiration, encouragement and unconditional support throughout my life.

1.0 INTRODUCTION

Many decision-making processes involve hierarchies of autonomous agents operating under joint resource and structural constraints. A realistic mathematical model would recognize the independent role of the lower-level agents as a part of the overall decision-making process rather than simply assuming that they execute the decisions of a central planner (i.e., the upper-level agent). Bilevel programming problems (BPPs) emerged as a broad class of optimization methods that are particularly suitable for modeling such decentralized decision-making scenarios. Bilevel optimization naturally arises in a number of application domains such as hazardous material transportation [33], network design [20] and interdiction [63, 65], revenue management [18], traffic planning [47, 58], energy [5], military [14, 37] and many other areas [48]. Thus, BPPs form an interesting and rich area of mathematical optimization problems with numerous research opportunities for development of novel results in theory, algorithms and applications.

Formally, BPPs model a decision-making process with a two-level hierarchy of independent decision-makers, namely, the leader and the follower. The decisions are performed in a predetermined sequence with the leader acting first, i.e., bilevel problems represent the leader's perspective. Consequently, the follower solves an optimization problem, which contains parameters (e.g., the right-hand sides of the follower's constraints) that are functionally dependent on the leader's decisions. On the other hand, the leader's objective and, possibly, constraints are also functions of both the leader's and follower's decision variables. Therefore, in the course of the decision-making process the leader should take into account the follower's rational response (i.e., optimal solutions to the follower's optimization problem), referred to as the *lower-level (rational) reaction set*.

In general, the lower-level reaction set is not necessarily a singleton, i.e., the follower's optimization problem may contain multiple optimal solutions for a given leader's decision [24]. If the follower implements a solution, which is the most favorable decision for the leader, then such BPPs are referred to as *optimistic*. On the contrary, *pessimistic* BPPs assume that the follower always selects the least favorable solution for the leader.

If the lower-level optimization problem is convex, then, perhaps, the most popular approach in the literature is to replace the lower-level problem by the corresponding Karush-Kuhn-Tucker (KKT) optimality conditions [6], thus yielding a *mathematical program with equilibrium constraints*, also often referred to as a *mathematical program with complementarity constraints* [21, 43]. One standard example of this approach is given by bilevel linear programs (bilevel LPs), which can be equivalently represented as single-level linear mixed integer programs (MIPs) [3]. Note that in contrast to single-level LPs, bilevel LPs are NP-hard [26]. Nevertheless, availability of effective modern MIP solvers such CPLEX [35] and Gurobi [36] allows solution of reasonably large-sized bilevel LPs. Unfortunately, when the lower-level problem is not convex, e.g., some of the follower's variables have integrality restrictions, such simplistic single-level MIP reformulations are typically not applicable (or become too large to handle explicitly), and the problem becomes notoriously difficult to solve.

This dissertation is focused on the development of exact solution techniques for bilevel programs with combinatorial structures in the lower-level optimization problem. In particular, we consider models arising in resource distribution systems that involve bilevel decision-making hierarchies with knapsack and assignment constraints. For example, in such systems the leader may reserve some part of the available resources to himself/herself, while simultaneously distributing the remaining resources to the followers, who, in turn, solve their own optimization problems using the resources allocated to them by the leader.

The remainder of this dissertation is organized as follows. In Chapter 2, we consider a class of nonlinear bilevel 0–1 knapsack problems, where the upper-level objective is a nonlinear integer function of both the leader's and the follower's decisions. At the lower level the follower solves a linear binary knapsack problem, where the right-hand side of the knapsack constraint depends on the capacity allocated by the leader. After discussing

computational complexity issues, we propose an exact solution approach using an equivalent single-level value function reformulation. To illustrate its performance, we conduct extensive computational experiments with quadratic and fractional binary objective functions.

In Chapter 3, we consider a class of bilevel assignment problems in which each decision-maker, i.e., the leader and the follower, has its own objective function and controls a distinct subset of edges in a given bipartite graph. The leader acts first by choosing some of his/her edges. Subsequently, the follower completes the assignment process. The subset of edges selected by the decision-makers is required to constitute a perfect matching. We propose an exact solution approach, which is based on a branch-and-bound framework and exploits structural properties of the assignment problem. Extensive computational experiments with linear sum and linear bottleneck objective functions are conducted to demonstrate the performance of the developed methods. Furthermore, the bilevel assignment problem is known to be NP-hard. Thus, we also describe some polynomially solvable classes of the problem.

In Chapter 4, we consider a class of the vaccine procurement policy problems, which can be formulated as a bilevel MIP. At the upper-level, the government orders antigens to minimize the maximum shortage of antigens subject to a budgetary constraint. At the lower-level, the manufacturer chooses a production plan so as to maximize its individual profit. We describe a tailored branch-and-bound algorithm along with a threshold algorithm to solve the proposed bilevel MIP. Our computational examples comparing solution of the bilevel model vs. its single-level modifications that represents a centralized decision-maker, e.g., the government, provide some interesting insights.

Finally, Chapter 5 concludes the discussion summarizing the contributions of this dissertation. Detailed description of the test instances for all considered problems can be found in Appendices A and B.

2.0 EXACT SOLUTION APPROACH FOR A CLASS OF NONLINEAR BILEVEL KNAPSACK PROBLEMS

In this chapter we consider a general class of *nonlinear bilevel knapsack problems (NBKPs)*, where the leader controls some resource and decides on the allocation of this resource between himself/herself and the follower. Consequently, both the leader and the follower solve 0–1 knapsack problems, where the right-hand sides of the linear knapsack constraints in both problems depends on the resource allocated by the leader. The follower's objective function is assumed to be linear, while the leader's overall objective is a sum of two general (possibly nonlinear) functions of the leader's and the follower's decisions, respectively. Specifically, we formulate NBKP as:

$$\text{[NBKP]} \quad \max_{\alpha,\mathbf{z},\mathbf{x}} \quad f(\alpha,\mathbf{z}) + g(\mathbf{x}), \tag{2.1a}$$

$$\text{subject to} \quad \sum_{i=1}^{m} w_i z_i \leq \alpha,\ \mathbf{z} \in \{0,1\}^m, \tag{2.1b}$$

$$\underline{b} \leq \alpha \leq \bar{b},\ \alpha \in \mathbb{Z}^1, \tag{2.1c}$$

$$\mathbf{x} \in \operatorname{argmax}\left\{ \sum_{j=1}^{n} c_j x_j :\ \sum_{j=1}^{n} a_j x_j \leq h(\alpha),\ \mathbf{x} \in \{0,1\}^n \right\}, \tag{2.1d}$$

where $h : [\underline{b}, \bar{b}] \to [0, \bar{h}]$, $f : \mathbb{Z}^1 \times \{0,1\}^m \to \mathbb{R}^1$ and $g : \{0,1\}^n \to \mathbb{R}^1$. The leader's decision variables are given by $\alpha \in \mathbb{Z}^1$ and $\mathbf{z} \in \{0,1\}^m$, while the follower's decision variables are given by $\mathbf{x} \in \{0,1\}^n$. Integer variable α corresponds to the resource allocation decision controlled by the leader (i.e., the value of α sets the resource capacity reserved by the leader for himself/herself), and binary variables $\mathbf{z}$ and $\mathbf{x}$ represent decisions of the leader's and follower's knapsack problems, respectively.

In the remainder of this chapter we make the following assumptions:

A1: Optimistic case of NBKP is considered.

A2: $w \in \mathbb{Z}^m_+$, $\underline{b} \in \mathbb{Z}^1_+$, $\bar{b} \in \mathbb{Z}^1_+$, $c \in \mathbb{Z}^n_+$, $a \in \mathbb{Z}^n_+$, and $\bar{h} \in \mathbb{Z}^1_+$; $h : [\underline{b}, \bar{b}] \cap \mathbb{Z}^1_+ \to [0, \bar{h}] \cap \mathbb{Z}^1_+$.

A3: There is an algorithm available for computing the value function:

$$\phi(\alpha) = \max_{\mathbf{z} \in \{0,1\}^m} \left\{ f(\alpha, \mathbf{z}) \ : \ \sum_{i=1}^{m} w_i z_i \leq \alpha \right\}, \quad \alpha \in [\underline{b}, \bar{b}] \cap \mathbb{Z}^1_+. \tag{2.2}$$

Assumption **A1** is typical in the hierarchical optimization literature [3, 48, 53]. Furthermore, in Section 2.5 we briefly discuss how to extend the proposed solution approach to the pessimistic case. Nonnegativity and integrality restrictions in **A2** are common for both single- and bilevel knapsack problems [13, 46, 60]. Assumption **A3** is necessary as our solution method is based on a single-level value function reformulation of NBKP (see Section 2.1). Value functions of binary and general integer optimization problems with low-degree polynomial (e.g., linear, quadratic) and fractional objective functions can be obtained (e.g., using dynamic programming) for a reasonably large set of right-hand sides, see examples in [40, 46, 54].

Bilevel programming problems with knapsack constraints are first introduced in [25]. One typical and often mentioned motivation for this class of optimization problems is in revenue management [13], where the leader sells some amount of a product by itself, and receives an additional profit from the items sold by an intermediary, i.e., the follower, who maximizes his/her own profit. Also, depending on the amount of the product transferred to the follower the leader may incur some transportation costs.

Bilevel knapsack problems with the leader's linear objective function (further referred to as BKPs) are often described to as bilevel extensions of the classical linear knapsack 0–1 problem [13, 25]. Similarly, general NBKPs form a class of bilevel extensions of nonlinear (e.g., quadratic [38, 46, 60]) knapsack 0–1 problems. Single-level linear and nonlinear knapsack problems naturally arise in various applications that involve resource allocation/scheduling with the centralized decision-maker. Thus, bilevel problems such BKPs and NBKPs are capable of modeling situations, where the upper-level DM, while completely determining the resource allocation decisions, does not have the full control over some parts of the decision-making process [17]. This concept is captured using the notion of the follower, his/her

decision variables and the lower-level linear 0–1 knapsack problem. Thus, there exists a term in the leader's objective function given by $g(\cdot)$ that is completely determined by the follower's decision variables.

Observe the follower's knapsack constraint in (2.1d) can be equivalently re-written as

$$\sum_{j=1}^{n} a_j x_j + \tilde{h}(\alpha) \leq \bar{h}, \tag{2.3}$$

where $\tilde{h}(\alpha) = \bar{h} - h(\alpha)$. Then (2.3) can be naturally interpreted as a "joint" knapsack constraint of the follower's and the leader's decision variables similar to the models in [12, 44].

In the literature, enumeration [25], dynamic [13, 44] and integer [12, 45] programming approaches are proposed for BKPs with the leader's linear objective function. Moreover, a stochastic extension of the linear bilevel knapsack problem is provided in [53]. Finally, some related computational complexity issues are discussed in [16].

The contributions presented in this chapter are as follows.

- We make two modeling extensions to bilevel programs with knapsack constraints discussed in the literature. First, we consider a class of problems, where the leader's objective is a general function in the form of (2.1a). Second, the capacity of the follower's knapsack constraint in (2.1d) depends on the leader's resource allocation decision α in a generic functional form $h(\cdot)$. Therefore, NBKP model is capable of capturing situations, where (i) the follower's decisions have a nonlinear effect on the leader's objective, and (ii) the follower's available capacity is an arbitrary function $h(\alpha)$ of the leader's resource allocation decision α (e.g., some of the allocated resources may be lost during the shipment process of the resource).
- We demonstrate that a class of BKPs from [13, 25] is a special case of NBKP. We provide theoretical computational complexity results for BKP establishing that (i) the problem remains difficult (NP-hard and not approximable in polynomial time) even if $\bar{b} = +\infty$ (i.e., unlimited resource for the leader) and (ii) the problem of checking whether a given feasible solution is locally optimal (with respect to a simple "plus/minus one" neighborhood of the leader's integer decision variable α) is also difficult.

- Exploiting an equivalent single-level value function reformulation, we propose an exact solution approach for solving general NBKPs. We also tailor the general method for two special classes of $g(\cdot)$, namely, quadratic and fractional 0–1 functions. Finally, we provide an extensive computational study demonstrating the performance of the developed algorithms.

The remainder of the chapter is organized as follows. Section 2.1 presents a single-level reformulation of NBKP using a value function based approach. Section 2.2 discusses related work in the literature and theoretical computational complexity issues. In Section 4.3, we propose an exact algorithm for solving NBKP. Section 2.4 presents the results of our computational experiments using randomly generated test instances, where the leader's objective function is either quadratic or fractional. Section 2.5 concludes the chapter highlighting possible directions for future work.

2.1 VALUE FUNCTION REFORMULATION

We reformulate NBKP using the follower's and leader's value functions. For $\alpha \in [\underline{b}, \bar{b}] \cap \mathbb{Z}^1_+$, the value function of the first term in the leader's objective function (2.1a) is given by (2.2). For $k \in \{1, 2, \ldots, n\}$ and $\beta \in [0, \bar{h}] \cap \mathbb{Z}^1_+$, denote the follower's value function by:

$$\psi_k(\beta) = \max_{\mathbf{x} \in \{0,1\}^k} \left\{ \sum_{j=1}^{k} c_j x_j : \sum_{j=1}^{k} a_j x_j \leq \beta \right\}.$$

Then, NBKP can be reformulated as the following single-level problem:

$$\begin{aligned}
[\text{VF-NBKP}] \quad \max_{\alpha, \mathbf{x}} \quad & \phi(\alpha) + g(\mathbf{x}) && \text{(2.4a)} \\
\text{subject to} \quad & \alpha \in \{\underline{b}, \ldots, \bar{b}\}, && \text{(2.4b)} \\
& \sum_{j=1}^{n} c_j x_j \geq \psi_n\left(h(\alpha)\right), && \text{(2.4c)} \\
& \sum_{j=1}^{n} a_j x_j \leq h(\alpha), && \text{(2.4d)}
\end{aligned}$$

$$x_j \in \{0,1\}, \quad j = 1,\ldots,n. \tag{2.4e}$$

Proposition 1 establishes the equivalency of NBKP and VF-NBKP.

Proposition 1. *Given an optimal solution $(\alpha^*, \mathbf{x}^*)$ to VF-NBKP , let $\mathbf{z}^* \in \arg\max \phi(\alpha^*)$. Then, $(\alpha^*, \mathbf{z}^*, \mathbf{x}^*)$ is an optimal solution to NBKP. Conversely, for any optimal solution $(\alpha^*, \mathbf{z}^*, \mathbf{x}^*)$ of NBKP, $(\alpha^*, \mathbf{x}^*)$ is an optimal solution of VF-NBKP. Consequently, the optimal objective function values of the two problems are equal.*

Proof. Constraint (2.4b) chooses an integer capacity α between $\underline{b}$ and $\bar{b}$. Constraints (2.4c) and (2.4d) ensure that the follower maximizes his/her own objective function subject to the capacity constraint. The result follows from Assumption **A1**. □

For $\alpha \in \{\underline{b},\ldots,\bar{b}\}$ define

$$\lambda(\alpha) = \max_{\mathbf{x}\in\{0,1\}^n} \{g(\mathbf{x}) : (2.4c),\ (2.4d)\}. \tag{2.5}$$

Then, Proposition 1 implies that problem (2.1) can be equivalently reformulated as:

$$\max_{\alpha\in\{\underline{b},\ldots,\bar{b}\}} \mathcal{F}(\alpha) = \phi(\alpha) + \lambda(\alpha). \tag{2.6}$$

Value function based reformulations are often applied in the literature. For example, Kong et al. [40] and Özaltın et al. [54] used such reformulations for solving stochastic integer programs. Furthermore, Özaltın et al. [53] used a similar idea to reformulate the bilevel knapsack problem with stochastic right-hand sides as a two-stage stochastic program, while Brotcorne et al. [12] applied such reformulation for a class of bilevel linear knapsack problems.

2.2 COMPUTATIONAL COMPLEXITY ISSUES

2.2.1 Links to Previous Work

The discrete bilevel knapsack problem studied by Dempe and Richter [25] and Brotcorne et al. [13] is a special case of NBKP. Specifically, it is given by:

$$\text{[BKP]} \quad \max_{\alpha,\mathbf{x}} \quad f^1(\alpha,\mathbf{x}) = t\alpha + \sum_{i=1}^{n} d_i x_i, \tag{2.7a}$$

$$\text{subject to} \quad \alpha \in \{\underline{b},\dots,\bar{b}\}, \tag{2.7b}$$

$$\mathbf{x} \in \operatorname{argmax}\left\{ f^2(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j : \ \sum_{j=1}^{n} a_j x_j \le \alpha, \ \mathbf{x} \in \{0,1\}^n \right\}. \tag{2.7c}$$

If there is no integrality restriction for α in (2.7b), then the problem may not have an optimal solution for $t > 0$; however, if the solution exists, then it is also optimal for the discrete version of the problem [13]. Moreover, if $t \le 0$, then any optimal solution of the discrete bilevel knapsack problem (2.7) is also optimal for the problem without integrality restrictions for α [13]. These results motivate a typical assumption in the literature that $\alpha \in \mathbb{Z}^1$, which is also followed in this chapter (see Assumption **A2**).

Solving bilevel knapsack problems is computationally difficult (specifically, $\sum_2^p$-hard [16]). For NBKP given by (2.1), let $g(\mathbf{x}) = \sum_{i=1}^{n} d_i x_i$, $h(\alpha) = \alpha$, and $f(\alpha,\mathbf{z}) = t\alpha$, where t is some constant. Then under assumption **A2**, constraint (2.1b) is satisfied for any $\alpha \in \{\underline{b},\dots,\bar{b}\}$, and the leader's variables $\mathbf{z}$ can be discarded because function $f(\cdot)$ does not depend on $\mathbf{z}$. In this case, NBKP reduces to BKP, which implies that NBKP is at least as hard as BKP.

We note that if $t \ge 0$, and $d_i = c_i$ for all $i = 1,\dots,n$, then it is optimal to have $\alpha = \bar{b}$ and BKP reduces to a single-level optimization problem, namely, the linear 0–1 knapsack problem. This simplification clearly demonstrates that both BKP and NBKP are computationally hard due (at least in part) to NP-hardness of the linear 0–1 knapsack problem [29]. Recall that difficulty of solving either single-level or bilevel linear 0–1 knapsack problems directly depends on the value of the right-hand side in the knapsack constraint as both problems admit a pseudo-polynomial time solution method [13, 46]. On the other hand, it is also well-known that the single-level linear 0–1 knapsack problem is "easy" to solve for

sufficiently large right-hand sides, as all items can be added to the solution. Thus, it is interesting to investigate how the computational complexity of BKP is influenced by the value of the parameter $\bar{b}$, which determines the amount of the resource available to the leader and, consequently, through the value of $h(\cdot)$, to the follower.

2.2.2 Complexity of BKP with $\bar{b} = +\infty$

Consider the PARTITION problem: Given a set of positive integers $S = \{s_1, s_2, \ldots, s_n\}$, $n \geq 2$, does there exist a subset $S' \subseteq S$ such that:

$$\sum_{i:s_i \in S'} s_i = \sum_{i:s_i \in S \setminus S'} s_i = \frac{1}{2} \sum_{i=1}^{n} s_i ?$$

This problem is known to be NP-complete [29]. Given an instance of the PARTITION problem, define the following instance of BKP:

$$\max_{\alpha, \mathbf{x}} \quad f^1(\alpha, \mathbf{x}) = -\frac{3}{2}\alpha + \sum_{i=1}^{n} s_i x_i + \left(2 \sum_{i=1}^{n} s_i\right) x_{n+1} \tag{2.8a}$$

$$\text{subject to} \quad \alpha \in \mathbb{Z}^1_+, \tag{2.8b}$$

$$\max_{\mathbf{x}} \quad f^2(\mathbf{x}) = 2 \sum_{i=1}^{n} s_i x_i + \left(\sum_{i=1}^{n} s_i - \frac{1}{2}\right) x_{n+1} + M x_{n+2} \tag{2.8c}$$

$$\text{subject to} \quad 2 \sum_{i=1}^{n} s_i x_i + \left(\sum_{i=1}^{n} s_i\right) x_{n+1} + \left(\sum_{i=1}^{n} s_i + 1\right) x_{n+2} \leq \alpha, \tag{2.8d}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n+2,$$

where $M \geq 3 \sum_{i=1}^{n} s_i$. We assume that $s_i \geq 1$ for all $i = 1, \ldots, n$.

Lemma 1. *Let $(\alpha^*, \mathbf{x}^*)$ be an optimal solution of (2.8). Then $f^1(\alpha^*, \mathbf{x}^*) \geq 0$, and $f^1(\alpha^*, \mathbf{x}^*) = 0$ iff the PARTITION problem has a solution.*

Proof. First, note that $\alpha = 0$ and $x_1 = \ldots = x_{n+2} = 0$ is a feasible solution of (2.8) and $f^1(0, \mathbf{0}) = 0$. Next, we show that if $\alpha \neq \sum_{i=1}^{n} s_i$, then $f^1(\alpha, \mathbf{x}) \leq 0$. Furthermore, if $\alpha = \sum_{i=1}^{n} s_i$, then $f^1(\alpha, \mathbf{x}) > 0$ only if the PARTITION problem does not have a solution.

Formally, consider the following four cases:

(a) Let $0 \leq \alpha \leq \sum_{i=1}^{n} s_i - 1$, then $x_{n+1} = x_{n+2} = 0$ and $2\sum_{i=1}^{n} s_i x_i \leq \alpha$. Therefore:

$$f^1(\alpha, \mathbf{x}) = -\frac{3}{2}\alpha + \sum_{i=1}^{n} s_i x_i \leq -\frac{3}{2}\alpha + \frac{1}{2}\alpha = -\alpha \leq 0.$$

(b) If $\alpha = \sum_{i=1}^{n} s_i$, then $x_{n+2} = 0$. Furthermore, there are two possible situations:

(1b) If the PARTITION problem has a solution, then $x_{n+1} = 0$, $f^2(\mathbf{x}) = \sum_{i=1}^{n} s_i$ and $f^1(\alpha, \mathbf{x}) = -\sum_{i=1}^{n} s_i < 0$.

(2b) If the PARTITION problem does not have a solution, then $x_{n+1} = 1$, $x_1 = \ldots = x_n = 0$, $f^2(\mathbf{x}) = \sum_{i=1}^{n} s_i - \frac{1}{2}$ and $f^1(\alpha, \mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} s_i > 0$.

(c) If $\sum_{i=1}^{n} s_i + 1 \leq \alpha \leq 2\sum_{i=1}^{n} s_i$, then $x_{n+2} = 1$, $x_{n+1} = 0$, and $2\sum_{i=1}^{n} s_i x_i \leq \alpha - \sum_{i=1}^{n} s_i - 1$. Therefore, we have:

$$f^1(\alpha, \mathbf{x}) \leq -\frac{3}{2}\alpha + \frac{1}{2}(\alpha - \sum_{i=1}^{n} s_i - 1) = -\alpha - \frac{1}{2}(\sum_{i=1}^{n} s_i + 1) \leq 0.$$

(d) Finally, if $2\sum_{i=1}^{n} s_i + 1 \leq \alpha$, then we have:

$$f^1(\alpha, \mathbf{x}) \leq -\frac{3}{2}\alpha + \sum_{i=1}^{n} s_i + 2\sum_{i=1}^{n} s_i \leq 3\sum_{i=1}^{n} s_i - \frac{3}{2}(2\sum_{i=1}^{n} s_i + 1) \leq -\frac{3}{2} \leq 0.$$

Clearly, the discussion above implies that the optimal solution of (2.8) is equal to zero iff the PARTITION problem has a solution. □

A direct observation from Lemma 1 is that BKP remains NP-hard even if α is not bounded from above, i.e., we may assume that $\bar{b} \geq \sum_{j=1}^{n} a_j$, or, equivalently (and with a slight abuse of notation) $\bar{b} = +\infty$.

Proposition 2. *BKP remains NP-hard even if* $\bar{b} = +\infty$.

The intuition behind this result is that the computational difficulty of BKP and NBKP, as its generalization, can be contributed to a considerable extent not to the resource limitations of the leader (given by $\bar{b}$), but to the hierarchical (bilevel) structure of the overall decision-making process. This fact is in line with similar results on the theoretical computational complexity of bilevel linear programs, which cannot be solved or approximated in polynomial time unless $P = NP$ [26], while single-level linear programs are known to be "easy" (i.e., polynomially solvable).

Recent work in [16] demonstrates that BKP does not allow polynomial time approximation algorithms with finite worst case guarantee. Another simple observation from Lemma 1, which complements this result, is that BKP does not admit a polynomial time approximation scheme even if $\bar{b} = +\infty$. Specifically, suppose there exists a polynomial time approximation algorithm for solving BKP that returns a solution with the objective function value of at least $\epsilon \times OPT$, where $0 < \epsilon \leq 1$ is a fixed parameter and OPT denotes the optimal objective function value. Then, whenever this algorithm returns zero (positive solution), by Lemma 1 we would be able to conclude that the PARTITION problem has a solution (does not have a solution). Thus, we establish that:

Corollary 1. *Assuming $P \neq NP$, for any fixed ϵ, $0 < \epsilon \leq 1$, there is no polynomial time ϵ-approximation algorithm for BKP, even if $\bar{b} = +\infty$.*

2.2.3 Complexity of checking local optimality

In practice, it is typical that computationally challenging problems are solved (not necessarily to optimality) by applying algorithms that exploit some local search based ideas. Unfortunately, we show that checking whether a particular feasible solution is locally optimal for BKP is still NP-hard. Examples of similar results on complexity of checking local optimality can be found in [55, 57]. Our definition of local optimality is given with respect to the neighborhood of "adjacent" (follower's) knapsack problems with consecutive right-hand sides (as defined by Blair [10]), i.e., α and $\alpha + 1$ (or $\alpha - 1$ and α) in (2.7c).

Definition 1. *Define $\tilde{\alpha}$ to be a locally optimal solution of BKP if exactly one of the conditions below holds:*

- *if $\underline{b} < \tilde{\alpha} < \bar{b}$ then $\mathcal{F}(\tilde{\alpha}) \geq \mathcal{F}(\tilde{\alpha}-1)$ and $\mathcal{F}(\tilde{\alpha}) \geq \mathcal{F}(\tilde{\alpha}+1)$;*
- *if $\tilde{\alpha} = \underline{b}$ then $\mathcal{F}(\tilde{\alpha}) \geq \mathcal{F}(\tilde{\alpha}+1)$;*
- *if $\tilde{\alpha} = \bar{b}$ then $\mathcal{F}(\tilde{\alpha}) \geq \mathcal{F}(\tilde{\alpha}-1)$.*

Consider any instance of the PARTITION problem such that $\sum_{i=1}^{n} s_i$ is even, $\sum_{i=1}^{n} s_i \geq 6$ and $s_i \geq 2$ for $i = 1, \ldots, n$. It is rather easy to show that this restricted version of the PARTITION problem remains NP-hard. Define a corresponding instance of BKP as follows:

$$\max_{\alpha,\mathbf{x}} \quad f^1(\alpha, \mathbf{x}) = -2\alpha + \sum_{i=1}^{n} s_i x_i + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) x_{n+1} + \left(2\sum_{i=1}^{n} s_i\right) x_{n+2} \tag{2.9a}$$

$$\text{subject to} \quad \frac{1}{2}\sum_{i=1}^{n} s_i - 1 \leq \alpha \leq \frac{1}{2}\sum_{i=1}^{n} s_i + 1, \alpha \in \mathbb{Z}^1_+, \tag{2.9b}$$

$$\max_{\mathbf{x}} \quad f^2(\mathbf{x}) = \sum_{i=1}^{n} s_i x_i + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - \frac{5}{3}\right) x_{n+1} + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - \frac{1}{2}\right) x_{n+2} \tag{2.9c}$$

$$\text{subject to} \quad \sum_{i=1}^{n} s_i x_i + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) x_{n+1} + \left(\frac{1}{2}\sum_{i=1}^{n} s_i\right) x_{n+2} \leq \alpha, \tag{2.9d}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n+2.$$

Lemma 2. *For problem (2.9):*

1. *$\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$ is a locally optimal solution iff the PARTITION problem has a solution;*
2. *$\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i$ is a locally optimal solution iff the PARTITION problem does not have a solution;*
3. *if $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i + 1$ is a locally optimal solution then the PARTITION problem has a solution.*

Proof. From (2.9b) it follows that α can take only three possible values. Next, we consider each of them. Specifically:

(a) If $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$, then there are two possible cases:

(a-1) If equation $\sum_{i=1}^{n} s_i x_i = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$ has a 0–1 solution, then $x_{n+1} = x_{n+2} = 0$, $f^2(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$ and

$$f^1(\alpha, \mathbf{x}) = -2\left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) = -\frac{1}{2}\sum_{i=1}^{n} s_i + 1.$$

(a-2) If equation $\sum_{i=1}^{n} s_i x_i = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$ does not have a 0–1 solution, then $x_1 = \ldots = x_n = 0$, $x_{n+1} = 1$, $x_{n+2} = 0$, $f^2(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} s_i - \frac{5}{3}$ and

$$f^1(\alpha, \mathbf{x}) = -2\left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) = -\frac{1}{2}\sum_{i=1}^{n} s_i + 1.$$

Thus, $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i - 1) = -\frac{1}{2}\sum_{i=1}^{n} s_i + 1$ for both of the above cases.

(b) If $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i$, then there are two possible cases:

(b-1) If the PARTITION problem has a solution, then $x_{n+1} = x_{n+2} = 0$, $f^2(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} s_i$ and

$$f^1(\alpha, \mathbf{x}) = -2\left(\frac{1}{2}\sum_{i=1}^{n} s_i\right) + \left(\frac{1}{2}\sum_{i=1}^{n} s_i\right) = -\frac{1}{2}\sum_{i=1}^{n} s_i.$$

(b-2) If the PARTITION does not have a solution, then $x_1 = \ldots = x_{n+1} = 0$, $x_{n+2} = 1$, $f^2(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} s_i - \frac{1}{2}$, and

$$f^1(\alpha, \mathbf{x}) = -2\left(\frac{1}{2}\sum_{i=1}^{n} s_i\right) + \left(2\sum_{i=1}^{n} s_i\right) = \sum_{i=1}^{n} s_i.$$

Thus, $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i) = \sum_{i=1}^{n} s_i$ iff the PARTITION does not have a solution.

(c) If $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i + 1$, then there are two possible situations based on the value of x_{n+2}:

(c-1) If $x_{n+2} = 0$, then

$$f^1(\alpha, \mathbf{x}) \leq -2\alpha + \alpha = -\frac{1}{2}\sum_{i=1}^{n} s_i - 1.$$

(c-2) If $x_{n+2} = 1$, then $\sum_{i=1}^{n} s_i x_i + \left(\frac{1}{2}\sum_{i=1}^{n} s_i - 1\right) x_{n+1} \leq 1$. Based on our assumptions for the PARTITION problem, it implies that $x_1 = \ldots = x_{n+1} = 0$, and

$$f^1(\alpha, \mathbf{x}) \leq -2\left(\frac{1}{2}\sum_{i=1}^{n} s_i + 1\right) + \left(2\sum_{i=1}^{n} s_i\right) = \sum_{i=1}^{n} s_i - 2$$

Thus, if $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i + 1$, then $\mathcal{F}(\alpha) < \sum_{i=1}^{n} s_i$ for both of the above cases.

Next, we prove the claims of the lemma as follows:

1. If $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$ is a locally optimal solution, then the PARTITION problem must have a solution, because otherwise $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i - 1) < \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i)$, which follows from (a) and (b-2). On the other hand, if the PARTITION problem has a solution, then $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$ is a locally optimal solution, because $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i - 1) > \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i)$, which follows from (a) and (b-1).

2. If $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i$ is a locally optimal solution, then the PARTITION problem must have no solution, because otherwise $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i - 1) > \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i)$, which follows from the discussion above. Furthermore, if the PARTITION problem does not have a solution, then $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i$ is a locally optimal solution, because $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i) > \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i - 1)$ and $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i) > \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i + 1)$, which follows from (b-2) and (c), respectively.

3. If $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i + 1$ is a locally optimal solution, then $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i + 1) \geq \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i)$. Hence, the PARTITION problem must have a solution, because otherwise we have that $\mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i + 1) < \mathcal{F}(\frac{1}{2}\sum_{i=1}^{n} s_i)$, which follows from (b-2) and (c).

□

From the statement (i) of Lemma 2, the following result holds:

Proposition 3. *Checking whether a given solution is locally optimal for BKP is NP-hard.*

Moreover, suppose there exists a polynomial time algorithm for solving BKP that returns a locally optimal solution. Then, from Lemma 2, we would conclude that the answer for the PARTITION problem is "yes" iff the algorithm returns either $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i + 1$ or $\alpha = \frac{1}{2}\sum_{i=1}^{n} s_i - 1$. This observation implies the following result.

Corollary 2. *Assuming $P \neq NP$, there is no polynomial time algorithm that finds a locally optimal solution for BKP according to Definition 1.*

2.3 EXACT SOLUTION APPROACH

In this section, we propose an exact solution approach for NBKP based on its value function reformulation (2.6). First, we compute and store the leader's value function $\phi(\alpha)$ for all $\alpha \in \{\underline{b}, \ldots, \bar{b}\}$, as well as the follower's value function $\psi_k(\beta)$ for all $k \in \{1, 2, \ldots, n\}$ and $\beta \in \{0, \ldots, \bar{h}\}$ by using dynamic programming. Then, given α, evaluating $\mathcal{F}(\alpha)$ requires solving problem (2.5), which is a nonlinear binary integer program. Özaltın et al. [53] presented the branch-and-backtrack algorithm to solve problem (2.5) when $g(\cdot)$ is linear. We generalize this approach for an arbitrary 0–1 function $g(\cdot)$.

2.3.1 Generic Branch-and-Backtrack Algorithm (GBBA)

The key idea of a generic branch-and-backtrack algorithm (GBBA) is to select or reject an item only if one of these choices maximizes the follower's profit. Specifically, GBBA incrementally constructs feasible solutions by considering one item at a time within a backtracking procedure. If selecting or rejecting an item has the same benefit for the follower, then a decision is given in favor of the leader's objective after branching and considering both options. (Recall that under assumption **A1** we focus on the optimistic case. However, GBBA can be easily modified to handle the pessimistic case in a similar manner, see discussion in Section 2.5.)

Given α, let $\mathcal{M}$ be the set of unprocessed nodes, and L be the current lower bound on $\lambda(\alpha)$. For each node $\mathcal{P}^m \in \mathcal{M}$, k_m is the number of items that still need to be backtracked, β_m is the remaining knapsack capacity, and U_m is an upper bound on the leader's objective.

Algorithm 1. *Generic branch-and-backtrack algorithm (GBBA) for computing $\lambda(\alpha)$*

Step 0: (Initialization) Create a node $\mathcal{P}^0$ with $k_0 \leftarrow n$, $\beta_0 \leftarrow h(\alpha)$. Initialize list $\mathcal{M} \leftarrow \{\mathcal{P}^0\}$. Initialize lower bound $L \leftarrow -\infty$.

Step 1: (Node selection) If $\mathcal{M} = \emptyset$, terminate with the optimal objective function value L; otherwise, select and delete node $\mathcal{P}^m$ from $\mathcal{M}$.

Step 2: (Pruning) Calculate U_m. If $U_m \leq L$, go to Step 1.

Step 3: (Backtracking) While $k_m \geq 1$

(3a) If $a_{k_m} > \beta_m$, set $k_m \leftarrow k_m - 1$ and $x^m_{k_m} \leftarrow 0$, go to Step 3.

(3b) **Case 1:** If $\psi_{k_m-1}(\beta_m) < \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, set $\beta_m \leftarrow \beta_m - a_{k_m}$, $x^m_{k_m} \leftarrow 1$, $k_m \leftarrow k_m - 1$, go to Step 3.

Case 2: If $\psi_{k_m-1}(\beta_m) > \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, $x^m_{k_m} \leftarrow 0$, set $k_m \leftarrow k_m - 1$, go to Step 3.

Case 3: (Branching) If $\psi_{k_m-1}(\beta_m) = \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, create two nodes $\mathcal{P}^{m_1}$ and $\mathcal{P}^{m_2}$ such that

$\mathcal{P}^{m_1}$: $\beta_{m_1} \leftarrow \beta_m - a_{k_m}$, $x^{m_1} \leftarrow x^m$ and $x^{m_1}_{k_m} \leftarrow 1$. Set $k_{m_1} \leftarrow k_m - 1$.

$\mathcal{P}^{m_2}$: $\beta_{m_2} \leftarrow \beta_m$, $x^{m_2} \leftarrow x^m$ and $x^{m_2}_{k_m} \leftarrow 0$. Set $k_{m_2} \leftarrow k_m - 1$.

Update $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{P}^{m_1}, \mathcal{P}^{m_2}\}$, go to Step 1.

Step 4: (Update Lower bound) If $L < g(\mathbf{x}^m)$, then $L \leftarrow g(\mathbf{x}^m)$, go to Step 1.

In Step 0, the algorithm initializes the root node with capacity $h(\alpha)$. In Step 1, a node is chosen from $\mathcal{M}$ according to a node selection heuristic. Step 2 checks whether the current node is promising based on its upper bound and the current value of the lower bound. If so, Step 3a checks whether selecting item k_m is feasible based on the remaining capacity β_m. If it is feasible, then Step 3b compares the follower's benefit from selecting or not selecting item k_m. If selecting is more profitable, then item k_m is selected, the remaining capacity as well as the current solution is updated, and backtracking is resumed by considering item $k_m - 1$. If selecting is not profitable, then the algorithm continues backtracking from the next item. If these decisions are equally profitable, the algorithm branches by creating two new nodes. On one of these nodes item k_m is selected, and on the other one it is not. In Step 4, the current value of the lower bound L is updated.

In general, GBBA can handle any 0–1 function, e.g., polynomials of the form:

$$g(\mathbf{x}) = \sum_{S \subseteq \{1,2,\ldots,n\}} p_S \prod_{j \in S} x_j, \tag{2.10}$$

where $\prod_{j \in \emptyset} x_j = 1$ and $p_S \in \mathbb{R}^1$ for all $S \subseteq \{1, 2, \ldots, n\}$. Note that any pseudo-boolean function can be uniquely represented in form (3.7) [11].

Branch-and-backtrack performs well when there is a small number of alternative solutions to the follower's problem. Otherwise, its performance depends on the quality of the upper bound generated in Step 2, which clearly depends on $g(\cdot)$. Next, we tailor the branch-and-backtrack algorithm for two special cases of $g(\cdot)$.

2.3.1.1 Quadratic Case

In this section, we consider $g(\mathbf{x})$ given by:

$$g(\mathbf{x}) = \sum_{j=1}^{n} d_j x_j + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j. \tag{2.11}$$

We assume that $d \in \mathbb{Z}_+^n$, $Q = (q_{ij})_{n \times n} \in \mathbb{Z}_+^{n \times n}$ and Q is symmetric. Furthermore, we let $q_{ii} = 0$ for all $i \in \{1, \ldots, n\}$, which is due to the fact that for 0–1 variables $x_i^2 = x_i$, and the presence of the linear term.

Upper Bound (U_m). We use an approach similar to the one proposed by Gallo et al. [28] for the quadratic 0–1 knapsack problem. Specifically, let U_m be an upper bound for $g(\mathbf{x})$ at node $\mathcal{P}^m$. Note that at $\mathcal{P}^m$ variables $x_{k_m+1}, \ldots, x_n$ are fixed; we denote their values as $\bar{x}_{k_m+1}^m, \ldots, \bar{x}_n^m$, respectively. Then:

$$U_m = \max \quad \sum_{j=1}^{k_m} (\frac{1}{2}\pi_j^m + d_j) x_j + \sum_{j=k_m+1}^{n} (\frac{1}{2}\pi_j^m + d_j) \bar{x}_j^m \tag{2.12a}$$

$$\text{s.t.} \quad \sum_{j=1}^{k_m} c_j x_j \geq \psi_{k_m}(\beta_m), \tag{2.12b}$$

$$\sum_{j=1}^{k_m} a_j x_j \leq \beta_m, \tag{2.12c}$$

$$x_j \in \{0, 1\}, \ j = 1, \ldots, k_m, \tag{2.12d}$$

where

$$\pi_j^m = \begin{cases} \sum_{i=k_m+1}^{n} q_{ij} \bar{x}_i^m + \\ \max \left\{ \sum_{i=1}^{k_m} q_{ij} x_i : \sum_{i=1}^{k_m} a_i x_i \leq \beta_m - a_j, \ x_i \in \{0,1\}, \ i = 1, \ldots, k_m \right\} & \text{if } j \leq k_m, \\ \sum_{i=k_m+1}^{n} q_{ij} \bar{x}_i^m & \text{if } j > k_m. \end{cases}$$

Note that updating π_j^m at each node m requires solving a knapsack problem for each variable $j = 1, \ldots, n$. The value function of the knapsack problem associated with each variable might be computed and stored a priori to improve runtime efficiency. However, this approach requires excessive memory storage as the number variables increases. Alternatively, we use a version of π_j^m that is valid over all nodes, specifically:

$$\pi_j = \max \left\{ \sum_{i=1, i\neq j}^{n} q_{ij} x_i : \sum_{i=1, i\neq j}^{n} a_i x_i \leq h(\alpha) - a_j,\ x_i \in \{0,1\},\ i = 1, \ldots, n \right\}.$$

Note that π_j is independent from x_i^m, $i = k_m + 1, \ldots, n$. Thus, problem (2.12) can be solved efficiently using a dynamic programming algorithm [13].

Lower Bound. After fixing variables $x_{k_m+1}, \ldots, x_n$ we have that

$$g(x_1, \ldots, x_{k_m}, \bar{x}_{k_m+1}, \ldots, \bar{x}_n) = \sum_{j=k_m+1}^{n} d'_j \bar{x}_j + \sum_{j=1}^{k_m} d'_j x_j + \frac{1}{2} \sum_{i=1}^{k_m} \sum_{j=1}^{k_m} q_{ij} x_i x_j,$$

where $d'_j = d_j + \frac{1}{2} \sum_{i=k_m+1}^{n} q_{ij} \bar{x}_i$ for all $j = 1, \ldots, n$. Hence, instead of keeping fixed values of $\mathbf{x}$ we can update the value of d'_j in Step 3b. We also maintain a lower bound η_m on $g(\mathbf{x})$.

Modification of GBBA for the quadratic case (we initialize $\eta_0 \leftarrow 0$ and $d^0 \leftarrow d$)

(3b) **Case 1:** If $\psi_{k_m-1}(\beta_m) < \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, $\beta_m \leftarrow \beta_m - a_{k_m}$, $\eta_m \leftarrow \eta_m + d_{k_m}^m$, $d_i^m \leftarrow d_i^m + q_{i,k_m}$ for $i = 1, \ldots, k_m - 1$, and set $k_m \leftarrow k_m - 1$, go to Step 3.

Case 2: If $\psi_{k_m-1}(\beta_m) > \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, set $k_m \leftarrow k_m - 1$, go to Step 3.

Case 3: (Branching) If $\psi_{k_m-1}(\beta_m) = \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, create two nodes $\mathcal{P}^{m_1}$ and $\mathcal{P}^{m_2}$ such that

$\mathcal{P}^{m_1}$: $\beta_{m_1} \leftarrow \beta_m - a_{k_m}$, $\eta_{m_1} \leftarrow \eta_m + d_{k_m}^m$, and $d_i^{m_1} \leftarrow d_i^m + q_{i,k_m}$ for $i = 1, \ldots, k_m - 1$. Set $k_{m_1} \leftarrow k_m - 1$.

$\mathcal{P}^{m_2}$: $\beta_{m_2} \leftarrow \beta_m$, $\eta_{m_2} \leftarrow \eta_m$, and $d_i^{m_2} \leftarrow d_i^m$ for $i = 1, \ldots, k_m - 1$. Set $k_{m_2} \leftarrow k_m - 1$.

Update $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{P}^{m_1}, \mathcal{P}^{m_2}\}$. Go to Step 1.

Note that $\eta_m \leq g(\mathbf{x}^m)$ when $k_m \geq 1$, and $\eta_m = g(\mathbf{x}^m)$ when $k_m = 0$. Thus, after each update of η_m at node m, all other unprocessed nodes $m' \in \mathcal{M}$ with upper bound $U_{m'} \leq \eta_m$ can be fathomed before completely backtracking node m.

2.3.1.2 Fractional Case In this section, we consider $g(\mathbf{x})$ given by a fractional 0–1 function as:

$$g(\mathbf{x}) = \frac{p_0 + \sum_{j=1}^{n} p_j x_j}{q_0 + \sum_{j=1}^{n} q_j x_j},$$

where $p_j,\ q_j \in \mathbb{Z}_+^1, j = 0, 1, \ldots, n$. We refer the reader to [11] for a related discussion on fractional 0–1 programming problems and their applications.

Upper Bound (U_m). Let $p'_0 = p_0 + \sum_{j=k_m+1}^{n} p_j \bar{x}_j^m$ and $q'_0 = q_0 + \sum_{j=k_m+1}^{n} q_j \bar{x}_j^m$. Moreover, let $\mathcal{K} = \{j = 1, \ldots, k_m : a_j \leq \beta_m\}$. Then, we compute U_m as:

$$U_m = \max \left\{ \frac{p'_0 + \sum_{j\in\mathcal{K}} p_j x_j}{q'_0 + \sum_{j\in\mathcal{K}} q_j x_j} : x_j \in \{0,1\},\ \forall j \in \mathcal{K} \right\} \tag{2.13}$$

Problem (2.13) can be solved in linear time using a threshold algorithm [34].

Lower Bound. After fixing variables $x_{k_m+1}, \ldots, x_n$ we have that

$$g(x_1, \ldots, x_{k_m}, \bar{x}_{k_m+1}, \ldots, \bar{x}_n) = \frac{p'_0 + \sum_{j=1}^{k_m} p_j x_j}{q'_0 + \sum_{j=1}^{k_m} q_j x_j}. \tag{2.14}$$

Hence, instead of keeping fixed values of $\mathbf{x}$ we can update the values of p'_0 and q'_0 in Step 3b. We define a lower bound η_m on $g(\mathbf{x}^m)$ as:

$$\eta_m = \frac{p'_0 + \min \left\{ \sum_{j=1}^{k_m} p_j x_j : \sum_{j=1}^{k_m} c_j x_j \geq \psi_{k_m}(\beta_m),\ \sum_{j=1}^{k_m} a_j x_j \leq \beta_m \right\}}{q'_0 + \max \left\{ \sum_{j=1}^{k_m} q_j x_j : \sum_{j=1}^{k_m} c_j x_j \geq \psi_{k_m}(\beta_m),\ \sum_{j=1}^{k_m} a_j x_j \leq \beta_m \right\}}. \tag{2.15}$$

Optimization problems that appear in denominator and numerator of (2.15) can be solved efficiently (e.g., using dynamic programming [13]), when computing the follower's value function $\psi_k(\cdot)$.

Modification of GBBA for the fractional case (we initialize $\eta_0 \leftarrow 0$, $p^0 \leftarrow p$ and $q^0 \leftarrow q$)

(3b) **Case 1:** If $\psi_{k_m-1}(\beta_m) < \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, $\beta_m \leftarrow \beta_m - a_{k_m}$, $p_0^m \leftarrow p_0^m + p_{k_m}$ and $q_0^m \leftarrow q_0^m + q_{k_m}$. Update η_m, set $k_m \leftarrow k_m - 1$, go to Step 3.

Case 2: If $\psi_{k_m-1}(\beta_m) > \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, set $k_m \leftarrow k_m - 1$, go to Step 3.

Case 3: (Branching) If $\psi_{k_m-1}(\beta_m) = \psi_{k_m-1}(\beta_m - a_{k_m}) + c_{k_m}$, create two nodes $\mathcal{P}^{m_1}$ and $\mathcal{P}^{m_2}$ such that

$\mathcal{P}^{m_1}$: $\beta_{m_1} \leftarrow \beta_m - a_{k_m}$, $p_0^{m_1} \leftarrow p_0^m + p_{k_m}$ and $q_0^{m_1} \leftarrow q_0^m + q_{k_m}$. Set $k_{m_1} \leftarrow k_m - 1$.

$\mathcal{P}^{m_2}$: $\beta_{m_2} \leftarrow \beta_m$, $p_0^{m_2} \leftarrow p_0^m$ and $q_0^{m_2} \leftarrow q_0^m$. Set $k_{m_2} \leftarrow k_m - 1$.

Update $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{P}^{m_1}, \mathcal{P}^{m_2}\}$. Go to Step 1.

Note that $\eta_m \leq g(\mathbf{x}^m)$ when $k_m \geq 1$, and $\eta_m = g(\mathbf{x}^m) = \frac{p_0^m}{q_0^m}$ when $k_m = 0$. After each update of η_m at node m, all other unprocessed nodes $m' \in \mathcal{M}$ with upper bound $U_{m'} \leq \eta_m$ can be fathomed before completely backtracking node m.

2.3.2 Multi-Pass Bounding Algorithm

Exhaustive search solves problem (2.6) by computing $\phi(\alpha) + \lambda(\alpha)$ for each $\alpha \in \{\underline{b}, \ldots, \bar{b}\}$, which can be burdensome if $\bar{b} \gg \underline{b}$. In this section, we develop an exact algorithm (referred to as the multi-pass bounding algorithm) to omit the calculation of $\lambda(\alpha)$ for unfavorable values of α by generating lower and upper bounds on $\lambda(\alpha)$.

Lemma 1([46]). $\psi_n(\beta)$ is piecewise constant and nondecreasing over β. Moreover, it can have discontinuities only at integer values of β.

Let $\kappa = (\bar{b} - \underline{b}) + 1$. First, we sort all values in $\{\underline{b}, \ldots, \bar{b}\}$ in increasing $h(\alpha)$ order $\{\alpha_{(1)}, \alpha_{(2)}, \ldots, \alpha_{(\kappa)}\}$ such that $h(\alpha_{(1)}) \leq h(\alpha_{(2)}) \leq \cdots \leq h(\alpha_{(\kappa)})$. Note that $\psi_n(h(\alpha))$ is piecewise constant and nondecreasing in $h(\alpha)$. Let $1 \leq \ell_1 < \ell_2 < \cdots < \ell_p = \kappa$ be such that $\psi_n(h(\alpha))$ is constant over $\{\alpha_{(1)}, \ldots, \alpha_{(\ell_1)}\}$ and $\{\alpha_{(\ell_i+1)}, \ldots, \alpha_{(\ell_{i+1})}\}$ for $i = 1, 2, \ldots, p-1$. Note that $\lambda(\alpha)$ is not monotone over $\{\alpha_{(1)}, \alpha_{(2)}, \ldots, \alpha_{(\kappa)}\}$. However, it is nondecreasing when $\psi_n(h(\alpha))$ is constant. Proposition 4 formalizes this observation.

Proposition 4. *$\lambda(\alpha)$ is piecewise constant and nondecreasing over $\{\alpha_{(1)}, \ldots, \alpha_{(\ell_1)}\}$ and $\{\alpha_{(\ell_i+1)}, \ldots, \alpha_{(\ell_{i+1})}\}$ for $i = 1, 2, \ldots, p-1$.*

The pseudo-code of the multi-pass bounding algorithm for solving (2.6) is outlined next.

Algorithm 2. *Multi-pass bounding algorithm*

Step 0 (Initialization) Call GBBA to calculate $\lambda(\alpha_{(\kappa)})$. Let s be the slack of constraint (2.4d). Initialize $L \leftarrow \lambda(\alpha_{(\kappa)}), F \leftarrow \psi_n(h(\alpha_{(\kappa)}))$, $\Gamma \leftarrow \{\alpha_{(1)}, \alpha_{(2)}, \ldots, \alpha_{(\kappa-1)}\}$, $LB \leftarrow \phi(\alpha_{(\kappa)}) + \lambda(\alpha_{(\kappa)})$.

Step 1 If $\Gamma \neq \emptyset$, set $\ell \leftarrow \max\{i : \alpha_{(i)} \in \Gamma\}$.

Step 1.1 If $\psi_n\left(h(\alpha_{(\ell)})\right) = F$,

(1.1a) **(Fixing)** If $s \geq h(\alpha_{(\ell+1)}) - h(\alpha_{(\ell)})$,

- Set $\lambda(\alpha_{(\ell)}) \leftarrow L$, $\Gamma \leftarrow \Gamma \setminus \{\alpha_{(\ell)}\}$, and $LB \leftarrow \max\{LB, \phi(\alpha_{(\ell)}) + \lambda(\alpha_{(\ell)})\}$.
- Set $s \leftarrow s - (h(\alpha_{(\ell+1)}) - h(\alpha_{(\ell)}))$, and $\ell \leftarrow \ell - 1$.
- If $\ell > 0$, go to Step 1.1, else go to Step 1.

(1.1b) **(Upperbounding)** Else (i.e., $s < h(\alpha_{(\ell+1)}) - h(\alpha_{(\ell)})$),

Step 1.2 (Pruning) If $\phi(\alpha_{(\ell)}) + \lambda(\alpha_{(\ell)}) \leq LB$,

- Set $\Gamma \leftarrow \Gamma \setminus \{\alpha_{(\ell)}\}$, $\ell \leftarrow \ell - 1$. If $\ell > 0$, go to Step 1.2, else go to Step 1.

Step 1.3 (Calculating) Call GBBA to calculate $\lambda(\alpha_{(\ell)})$ and s.

- Set $F \leftarrow \psi_n(h(\alpha_{(\ell)}))$, $L \leftarrow \lambda(\alpha_{(\ell)})$, and $\Gamma \leftarrow \Gamma \setminus \{\alpha_{(\ell)}\}$.
- Set $LB \leftarrow \max\{LB, \phi(\alpha_{(\ell)}) + \lambda(\alpha_{(\ell)})\}$, $\ell \leftarrow \ell - 1$. If $\ell > 0$, go to Step 1.1, else go to Step 1.

Step 0 calculates $\lambda(\alpha_{(\kappa)})$, and stores it as L. Set Γ contains all α values that need to be processed. Note that each $\alpha \in \{\alpha_{(1)}, \alpha_{(2)}, \ldots, \alpha_{(\kappa)}\}$ is a feasible solution, thus $\phi(\alpha) + \lambda(\alpha)$ constitutes a lower bound on the optimal objective function value of (2.6). Step 1 traverses over all unprocessed α values in nonincreasing $h(\alpha)$ order. Note that L is the exact value of $\lambda(\alpha_{(\ell)})$ in Step 1.1a, because the capacity decrease from $h(\alpha_{(\ell+1)})$ to $h(\alpha_{(\ell)})$ is less than s; thus, the follower must have the same optimal solutions for right-hand sides $\psi_n(h(\alpha_{(\ell+1)})$ and $\psi_n(h(\alpha_{(\ell)})$ in (2.4c).

Algorithm 2 assigns L as an upper bound on $\lambda(\alpha_{(\ell)})$ in Step 1.1b, when the follower has the same optimal objective value in $\psi_n(h(\alpha_{(\ell+1)})$ and $\psi_n(h(\alpha_{(\ell)})$, but different optimal solutions. In this case, the optimal solution to $\psi_n(h(\alpha_{(\ell)})$ is still optimal in $\psi_n(h(\alpha_{(\ell+1)})$, but not every optimal solution to $\psi_n(h(\alpha_{(\ell+1)})$ is feasible in $\psi_n(h(\alpha_{(\ell)})$. Therefore, in the optimistic case, the leader has more flexibility of choosing the most favorable lower-level solution among the optimal solutions to problem $\psi_n(h(\alpha_{(\ell+1)})$. As a result, L is an upper bound on $\lambda(\alpha_{(\ell)})$.

The value of L is updated when the follower's optimal value decreases. Step 1.2 checks whether the current solution $\alpha_{(\ell)}$ is promising. If so, Step 1.3 calls GBBA to update L. In each pass, the algorithm either finds optimal value, prunes, or updates upper bounds on unprocessed values of α in Γ. An illustration of multiple passes of Algorithm 2 with respect to $\lambda(\cdot)$ and $\psi_n(h(\cdot))$ is provided in Figure 1.

Note that Algorithm 2 does not make any assumption on the monotonicity of $h(\cdot)$. Corollary 3 shows that when $h(\alpha)$ is nondecreasing in α, i.e. $\alpha_{(1)} \leq \cdots \leq \alpha_{(\kappa)}$, the optimal solution to problem (2.6) can be found by calculating $\lambda(\alpha)$ only at the break points of $\psi(h(\alpha))$ over $\{\alpha_{(1)}, \alpha_{(2)}, \ldots, \alpha_{(\kappa)}\}$.

Corollary 3. *If $h(\cdot)$ is non-decreasing, then*

$$\max_{\alpha \in \{\underline{b}, \ldots, \bar{b}\}} \mathcal{F}(\alpha) = \max_{\alpha \in \{\alpha_{(\ell_1)}, \ldots, \alpha_{(\ell_p)}\}} \mathcal{F}(\alpha).$$

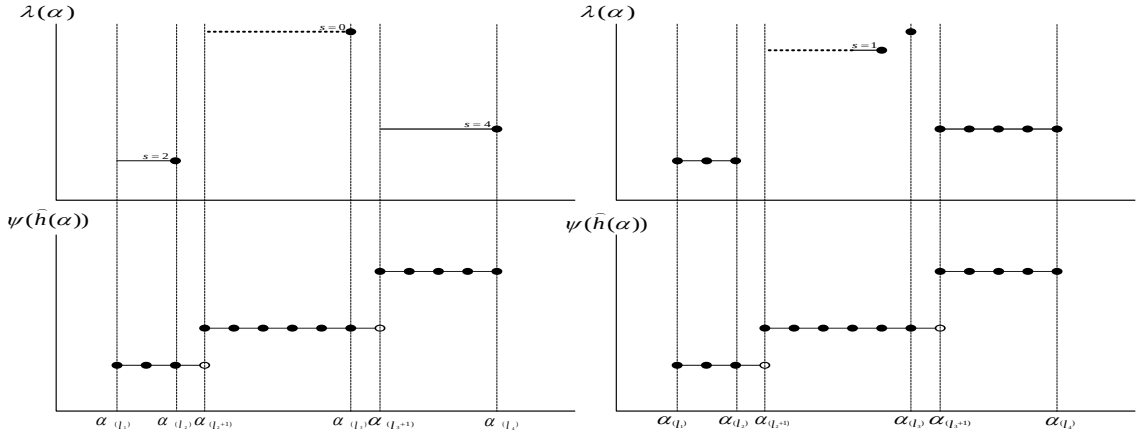
2.4 COMPUTATIONAL EXPERIMENTS

2.4.1 Test Instances and Setup

First, we describe the characteristics of our test instances, which are available online [8]. More details of test instances can be found in Appendix A.

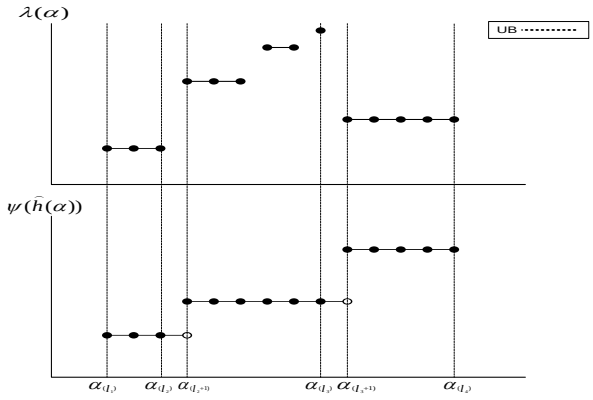
Follower's problem. Test instances of the follower's problem have different number of variables $n \in \{25, 50, 75, 100, 125, 150\}$. Computational difficulty of solving a knapsack problem is greatly affected by the correlation between profits and weights, i.e., the value of $\mathbf{c}$ and $\mathbf{a}$ parameters. Following the literature on knapsack-like problems (see, e.g., [46, 53]), we generate the follower's knapsack problem with varying degrees of correlation:

- uncorrelated: $a_j \sim U[1, 1000]$ and $c_j \sim U[1, 1000]$;
- correlated: $a_j \sim U[1, 1000]$ and $c_j = a_j + \max\{0, U[-100, 100]\}$;
- highly correlated: $a_j \sim U[1, 1000]$ and $c_j = a_j + 100$.



(a) First pass

(b) Second pass



(c) Final pass

Figure 1: An illustration of the multi-pass bounding algorithm.

The right-hand side function $h(\alpha)$ is generated in two different ways: (i) class S: $h(\alpha) \sim U[0.25, 0.75] \sum_{j=1}^{n} a_j$ and (ii) class C: $h(\alpha) = \bar{h} - \alpha$, where $\bar{h}$ is the largest right-hand side in the corresponding class S instance.

Leader's problem. We assume that $f(\alpha, \mathbf{z}) = \sum_{i=1}^{m} t_i z_i$, $\mathbf{t} \in \mathbb{R}_+^m$. We fix the number of leader's variables $m = 50$, and generate $w_i \sim U[1, 100]$. We consider 500 different right-hand sides between $\underline{b} = 500$ and $\bar{b} = 1000$ in constraint (2.1b). Then the leader's objective function $f(\alpha, \mathbf{z}) + g(\mathbf{x})$ is generated as follows:

- **Quadratic $g(\cdot)$.** We generate $t_i \sim U[1, 1000]$ and $d_j \sim U[1, 1000]$. The entries of matrix Q are generated according to uniform distribution between $[1, R]$ for $R = 10$ and 100. The density of Q is given by $\mu \in \{0, 0.1, 0.5, 1\}$.
- **Fractional $g(\cdot)$.** We generate $t_i \sim U[1, 1000] \times 10^{-6}$ and $p_j \sim U[1, 1000]$. The $\mathbf{q}$ vector is generated according to uniform distribution between $[1, R]$ for $R = 10$ and 100.

We generate 5 instances from each class, and report the solution time as well as the number of calls to GBBA. We refer to our test instances as $ICX - Y$, where $X \in \{Q, F\}$, i.e., quadratic or fractional, and $Y \in \{u, c, h\}$, i.e., uncorrelated, correlated, and highly correlated. We use a Windows 7 PC with 2.4GHz CPU and 2GB of RAM.

2.4.2 Results and Discussion

Tables 1 and 2 report the solution times of quadratic instances, and Table 3 presents that of fractional instances. Each entry in Tables 1, 2, and 3 is an average of five instances. The lower-level problem of all instances (either quadratic or fractional) of the same size are the same. All instances with less than 50 variables are solved within a second; hence, they are not reported. In addition, lengthy solution times are required for those instances with $n \geq 150$ variables due to extensive enumeration.

Not surprisingly, the computational difficulty of solving our test instances increases in the degree of correlation due to a larger-sized follower's reaction set. Furthermore, the solution times often increase with the density of matrix Q for the quadratic instances as the quality of the bounds in GBBA decreases. As seen in Tables 1 and 2, the solution times of the ICQ-h instances with $R = 10$ increase significantly when μ is increased from 0.1 to 0.5. The

effect of μ on the solution time is not that significant when $R = 100$. The effect of R on the solution time is most significant for highly correlated instances. In Table 3, the solution time of the ICF-h instances in class C with $n = 125$ variables increase by seven-fold when R is increased from 10 to 100.

Table 4 reports the number of calls to GBBA for quadratic and fractional instances. Observe that quadratic instances require fewer calls to GBBA than fractional instances, but still they have larger solution times. This is due to the fact that the upper bounding technique used in GBBA for the fractional instances generates tighter bounds than the approach available for the quadratic instances. As a result, execution time of GBBA is often longer for a quadratic instance in comparison to a fractional instance of the same size.

Note that in Table 4, the benefit of using *the multi-pass bounding algorithm* over exhaustive search is more significant in class C instances. The bounding procedure in the multi-pass bounding algorithm is more effective if the follower's optimal objective stays constant for many different right-hand sides (see Figure 1). The right-hand sides of class C instances are closer to each other in comparison to those of class S instances. Hence, a greater number of unfavorable right-hand sides are pruned for class C instances.

2.5 CONCLUDING REMARKS

We consider a general class of nonlinear bilevel knapsack problems, where the follower's decisions have a nonlinear effect on the leader's objective, and the follower's available capacity is an arbitrary function of the leader's capacity allocation decision.

We propose an exact solution approach, which runs efficiently as long as good upper and lower bounds can be generated in *the generic branch-and-backtrack algorithm*. In the computational experiments we consider quadratic and fractional objective functions for the leader. The results are encouraging as sizes and solution times of our nonlinear test instances are comparable to the results reported in the literature for the simpler linear case [13]. The key underlying factor is the fact that the follower's problem (2.1d) is still linear, which is exploited within the developed solution framework.

Table 1: Solution time for class C of quadratic instances (in seconds)

			R = 10			R = 100		
	n	$\mu = 0$	$\mu = 0.1$	$\mu = 0.5$	$\mu = 1$	$\mu = 0.1$	$\mu = 0.5$	$\mu = 1$
	75	1.4	1.4	1.6	1.4	1.6	1.0	1.4
	100	3.2	3.4	3.2	3.4	3.4	3.4	3.4
ICQ-u	125	6.6	6.6	6.4	6.2	6.4	6.6	6.2
	150	11.0	11.0	11.0	11.0	11.0	11.0	11.0
	75	1.6	1.4	1.6	1.6	1.6	1.4	1.2
	100	3.4	3.6	3.2	3.4	3.4	3.4	3.4
ICQ-c	125	6.6	6.4	6.6	6.6	6.2	6.2	6.2
	150	11.2	11.0	11.2	10.8	11.2	11.2	11.0
	75	1.6	1.6	2.2	2.4	2.4	2.4	2.6
	100	3.4	5.6	21	23.4	24.0	25.0	25.0
ICQ-h	125	6.6	22.4	573.2	675.8	653.4	720.6	719.4
	150	11.4	151.4	5,352.8	6,182.2	6,113.4	6,468.0	6,479.8

Table 2: Solution time for class S of quadratic instances (in seconds)

			R = 10			R = 100		
	n	$\mu = 0$	$\mu = 0.1$	$\mu = 0.5$	$\mu = 1$	$\mu = 0.1$	$\mu = 0.5$	$\mu = 1$
	75	1.6	1.4	1.4	1.4	1.4	1.4	1.4
	100	3.4	3.4	3.2	3.4	3.4	3.6	3.4
ICQ-u	125	6.2	6.4	6.6	6.4	6.6	6.2	6.6
	150	11.2	11.0	11.2	11.2	11.0	11.2	11.2
	75	1.4	1.6	1.4	1.4	1.6	1.4	1.4
	100	3.4	3.4	3.4	3.2	3.4	3.6	3.6
ICQ-c	125	6.2	6.6	6.4	6.6	6.4	6.6	6.6
	150	11.2	11.2	10.8	11.2	11.0	11.2	11.2
	75	1.4	1.6	4.8	6.8	6.6	8.4	8.4
	100	3.2	3.8	72.0	100.0	91.2	108.0	108.0
ICQ-h	125	6.2	10.2	1,658.8	2,371.0	2,027.0	2,570.2	2,575.8
	150	10.8	68.0	21,936.0	32,070.4	31,585.6	34,463.2	34,342.6

Table 3: Solution time for fractional instances (in seconds)

		C		S	
	n	R = 10	R = 100	R = 10	R = 100
ICF-u	75	< 1	< 1	< 1	< 1
	100	< 1	< 1	< 1	< 1
	125	< 1	< 1	< 1	< 1
	150	< 1	< 1	< 1	< 1
ICF-c	75	< 1	< 1	< 1	< 1
	100	< 1	< 1	< 1	< 1
	125	< 1	< 1	< 1	< 1
	150	< 1	< 1	< 1	< 1
ICF-h	75	< 1	< 1	1.8	1.8
	100	6.2	5.4	33.0	31.4
	125	21.6	155.0	225.2	936.4
	150	974.8	1,476.0	11,686.6	12,771.6

Table 4: Number of calls to GBBA (max = 500)

		X = Q		X = F	
	n	C	S	C	S
ICX-u	75	2.5	129.6	3.9	249.4
	100	5.5	180.4	8.3	313.6
	125	5.5	208.1	7.8	344.6
	150	7.1	216.1	9.1	382.4
ICX-c	75	39.6	192.0	45.2	431.8
	100	68.9	237.0	77.8	452.4
	125	87.2	267.2	98.7	471.6
	150	97.3	267.5	115.2	475.2
ICX-h	75	344.4	344.5	392.8	485.6
	100	329.8	343.3	375.6	491.0
	125	349.3	355.1	398.0	492.8
	150	414.1	331.2	486.6	495.6

A potential drawback of our solution procedure for large instances might be the explicit storage of value functions in computer memory. One approach is to calculate value functions when they are needed. However, this would adversely impact the solution times due to repetitive optimization steps that do not exploit the information gained from former calculations.

Model (2.4) formulates the optimistic NBKP, which might arise in a collaborative environment. To formulate the pessimistic NBKP, which corresponds to an adversarial environment, $\lambda(\alpha)$ should be redefined as:

$$\lambda(\alpha) = \min_{\mathbf{x}\in\{0,1\}^n} \{g(\mathbf{x}) : (2.4c),\ (2.4d)\} \qquad \alpha \in \{\underline{b}, \dots, \bar{b}\}. \tag{2.16}$$

Then, the single level value function reformulation proposed in (2.6) is still valid. We need to modify GBBA according to the redefinition of $\lambda(\alpha)$ in (2.16). Specifically, we should maintain a global upper bound in Step 0 and Step 4. Then, a node will be pruned if its lower bound is greater than the global upper bound in Step 2. The multi-pass bounding algorithm (as well as Proposition 4) will require similar modifications, i.e., taking into account the fact the follower prefers a solution that is not favorable to the leader.

Another interesting modification is to consider a class of bilevel knapsack problems, where the leader does not completely control the resource allocation through variable α, and the follower's knapsack constraint depends on the amount of the resource used by the leader (i.e., $\sum_{i=1}^m w_i z_i$), or, equivalently, on the amount of the unused resource (i.e., $\bar{b} - \sum_{i=1}^m w_i z_i$). This assumption results in the model similar to NBKP with (2.1d) replaced by:

$$\mathbf{x} \in \operatorname{argmax}\left\{\sum_{j=1}^n c_j x_j : \sum_{j=1}^n a_j x_j + \tilde{h}\left(\sum_{i=1}^m w_i z_i\right) \le \bar{h},\ \mathbf{x} \in \{0,1\}^n\right\}, \tag{2.17}$$

where $\tilde{h}(\cdot)$ is defined as in (2.3). Next, if we replace value function (2.2) by

$$\phi(\alpha) = \max_{\mathbf{z}\in\{0,1\}^m} \left\{f(\alpha, \mathbf{z}) \ : \ \sum_{i=1}^m w_i z_i = \alpha\right\}, \quad \alpha \in [\underline{b}, \bar{b}] \cap \mathbb{Z}^1_+, \tag{2.18}$$

then it is rather easy to show that similar approach (with some modifications) can be applied as well.

A possible direction of future research is to consider the leader's and follower's problems with multiple knapsack constraints. In this case, the upper- and lower-level value functions

$\phi(\cdot)$ and $\psi_k(\cdot)$ have to be computed over a multidimensional space. However, explicit storage of value functions requires substantially more memory in the multidimensional case. Furthermore, both GBBA and the multi-pass bounding algorithm have to be generalized to handle this extension.

2.6 ACKNOWLEDGMENT

The content of this chapter is reproduced with kind permission from Springer Science+Business Media: B. Beheshti, O. Y. Özaltın, M. H. Zare, and O. A. Prokopyev. Exact solution approach for a class of nonlinear bilevel knapsack problems. *Journal of Global Optimization*, accepted for publication, 2014. In addition, I am grateful to Gabriel Zenarosa and Austin Buchanan for their helpful comments.

This chapter is a part of research which was partially supported by AFOSR Grant FA9550-08-1-0268, NSF Grant CMMI-0825993 and DoD DURIP Grant FA2386-12-1-3032.

3.0 EXACT SOLUTION APPROACH FOR THE BILEVEL ASSIGNMENT PROBLEM

The *bilevel assignment problem* (BAP) is a class of assignment problems [15] that models two autonomous (conflicting or collaborative) decision makers, namely, the *leader* and the *follower*, who act in a bilevel hierarchy [31]. Each decision maker controls a distinct set of edges. The leader acts first by choosing a subset of its edges; consequently, the follower completes the assignment process. The edges selected by the leader and the follower are required to form a *perfect matching*. Decision makers pursue their own individual objectives. Therefore, the leader decides by considering the follower's reaction, i.e., an optimal solution to the lower-level optimization problem. In general, BAP belongs to a family of discrete bilevel (hierarchical) optimization problems, see surveys in [22, 48] and references therein.

Formally, let $G = (U \cup V, E)$ be a balanced bipartite graph with $|U| = |V| = n$. The edge set E is partitioned into two disjoint subsets E_ℓ and E_f, i.e. $E_\ell \cap E_f = \emptyset$ and $E_\ell \cup E_f = E$, which are controlled by the leader and the follower, respectively. Let $|E_\ell| = m_\ell$ and $|E_f| = m_f$. For all $i \in U$, define

$$V_\ell^{(i)} = \{j \in V : (i,j) \in E_\ell\} \quad \text{and} \quad V_f^{(i)} = \{j \in V : (i,j) \in E_f\}.$$

Likewise, we define $U_\ell^{(j)}$ and $U_f^{(j)}$ for all $j \in V$. Then BAP is formulated as follows:

$$\begin{aligned}
[\text{BAP}] \quad & \min_{\mathbf{x}} \quad g_\ell(\mathbf{x}, \mathbf{y}) && && (3.1a)\\
& \text{s.t.} \quad \min_{\mathbf{y}} \quad g_f(\mathbf{y}) && && (3.1b)\\
& \qquad \text{s.t.} \quad \sum_{j \in V_\ell^{(i)}} x_{ij} + \sum_{j \in V_f^{(i)}} y_{ij} = 1 && \forall i \in U, && (3.1c)
\end{aligned}$$

$$\sum_{i \in U_\ell^{(j)}} x_{ij} + \sum_{i \in U_f^{(j)}} y_{ij} = 1 \qquad \forall j \in V, \tag{3.1d}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E_f, \tag{3.1e}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E_\ell, \tag{3.1f}$$

where $g_\ell : \{0,1\}^{m_\ell + m_f} \to \mathbb{R}^1$ and $g_f : \{0,1\}^{m_f} \to \mathbb{R}^1$.

For a given feasible leader's decision $\mathbf{x}$, the lower-level optimization problem (3.1b)-(3.1e) may have alternative optimal solutions. If the follower always implements the most (least) favorable solution for the leader, then we refer to the obtained problem as the *optimistic* (*pessimistic*) bilevel program. In the remainder of the chapter we make the following assumptions:

A1: Optimistic case of BAP is considered.

A2: There exists a perfect matching in $G(U \cup V, E)$.

Assumption **A1** is typical in the bilevel (hierarchical) optimization literature [22]. Assumption **A2** is technical in order to ensure the existence of a feasible solution in BAP.

In conventional assignment problems, e.g., the linear sum assignment problem, there exists a single decision maker who fully controls the assignment processes [15, 56]. In BAP there are two autonomous decision makers. However, one player, namely, the leader, whose perspective is modeled in BAP, has an advantage of acting first and, hence, influencing the decision of the other player, i.e., the follower. Naturally, BAP can be generalized for the case of multiple followers. Thus, BAP is particularly appropriate for modeling a decentralized assignment process with a bilevel hierarchy of independent decision makers.

In this chapter we focus on the objective functions considered by Gassner and Klinz [31], namely, *linear sum* and *linear bottleneck* functions, formally defined for the leader as:

$$g_\ell(\mathbf{x},\mathbf{y}) = \sum_{(i,j) \in E_\ell} d_{ij} x_{ij} + \sum_{(i,j) \in E_f} d_{ij} y_{ij} \quad \text{and} \quad g_\ell(\mathbf{x},\mathbf{y}) = \max \Big\{ \max_{(i,j) \in E_\ell} \{d_{ij} x_{ij}\}, \max_{(i,j) \in E_f} \{d_{ij} y_{ij}\} \Big\}, \tag{3.2}$$

Table 5: BAP complexity for the optimistic case / pessimistic case [31].

		Leader	
		Sum	Bottleneck
Follower	Sum	NP-hard / NP-hard	NP-hard / NP-hard
	Bottleneck	NP-hard / NP-hard	Open / NP-hard

respectively, where $\mathbf{d} \in \mathbb{R}^{m_\ell + m_f}$. Similarly, for the follower we have:

$$g_f(\mathbf{y}) = \sum_{(i,j)\in E_f} c_{ij} y_{ij} \quad \text{and} \quad g_f(\mathbf{y}) = \max_{(i,j)\in E_f} \{c_{ij} y_{ij}\}, \tag{3.3}$$

where $\mathbf{c} \in \mathbb{R}^{m_f}$. In [31] the authors establish that, in contrast to the single-level linear sum and linear bottleneck assignment problems, all variants of BAP are NP-hard when the leader's and the follower's objective functions are in either a linear sum or a bottleneck form, except for the optimistic case when both decision makers have linear bottleneck objective functions (see Table 5). These results are not surprising and, in fact, in line with similar results on the theoretical computational complexity of bilevel linear programs, which cannot be solved in polynomial time unless $P = NP$ [26].

This chapter is focused on developing exact solution methods for BAP. In particular, we introduce a combinatorial branch-and-bound approach (Section 3.1.2), in which the branching process is based on imposing and forbidding the follower's edges on each node of the search tree, and pruning nodes that do not provide feasible solutions. Moreover, the lower bounds are constructed using a single-level relaxation by removing the follower's objective function. The general method is also tailored for BAP, where the follower's objective function is in either linear sum or linear bottleneck form (Section 3.1.3). Furthermore, we introduce other solution techniques for special classes of BAP. Specifically, we describe a mixed integer programming (MIP) reformulation when the follower's objective is a linear sum function (Section 3.2.1). We also show that BAP admits polynomial time solution algorithms if the leader's set of feasible actions is polynomially bounded (Section 3.2.2), or if the special

structure of the leader's and the follower's objective functions allows reduction of BAP to a single-level problem (Section 3.2.3). Finally, we provide an extensive computational study demonstrating the performance of the developed algorithms (Section 3.3) and conclude the chapter highlighting possible directions for future work (Section 3.4).

3.1 COMBINATORIAL BRANCH-AND-BOUND ALGORITHM

Ranking assignments according to the objective function value is a well-studied topic in the literature [19, 50, 59]. A typical application arises when the decision maker has to solve an assignment problem with constraints that are difficult to specify formally. In this case an optimal assignment can be found by enumerating suboptimal assignments until a solution satisfying the complicating constraints is found. Similarly, BAP can be viewed as an assignment problem in which a feasible assignment needs to satisfy the optimality criteria of the lower-level problem and, hence, can be solved using one of the ranking algorithms. The downside of this somewhat naïve approach is that the rank of the assignment for which optimality of the lower-level problem is satisfied, can be exponentially large. Nevertheless, this observation motivates our combinatorial branch-and-bound (B&B) algorithm for solving BAP, which we describe in detail next.

3.1.1 Preliminaries

A *matching* μ in $G(U \cup V, E)$ is a subset of edges such that each vertex is incident to at most one member of μ. A matching is *perfect* if every vertex in $U \cup V$ is incident to exactly one edge in μ. Let $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{m_\ell + m_f}$ be a feasible solution to (3.1c)-(3.1f), which induces a perfect matching $\mu(\mathbf{x}, \mathbf{y}) = \{(i,j) \in E_\ell \cup E_f : x_{ij} = 1 \text{ or } y_{ij} = 1\}$ in G. Also, let $\mu_f(\mathbf{y}) = \{(i,j) \in E_f : y_{ij} = 1\} \subseteq \mu(\mathbf{x}, \mathbf{y})$, which is also a matching in G; however, it is not necessarily perfect. Define $\mathcal{M}(G)$ to be the set of all perfect matchings (assignments) in G, i.e.,

$$\mathcal{M}(G) = \Big\{ \mu(\mathbf{x}, \mathbf{y}) \subseteq E_\ell \cup E_f \ : \ (3.1c) - (3.1f) \text{ hold } \Big\}.$$

For a given feasible leader's solution $\mathbf{x}$, we have the *follower's reaction set* defined as:

$$\mathcal{R}(\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmin}} \Big\{ g_f(\mathbf{y}) : \mu(\mathbf{x}, \mathbf{y}) \in \mathcal{M}(G) \Big\}.$$

Next, denote by $\mathcal{F}(G)$ the set of all perfect matchings in G that correspond to feasible solutions of BAP:

$$\mathcal{F}(G) = \Big\{ \mu(\mathbf{x}, \mathbf{y}) \in \mathcal{M}(G) : \ \mathbf{y} \in \mathcal{R}(\mathbf{x}) \Big\}.$$

Thus, $\mathcal{F}(G) \subseteq \mathcal{M}(G)$, and the optimistic case of BAP can be reformulated as:

$$\min_{\mathbf{x},\mathbf{y}} \Big\{ g_\ell(\mathbf{x}, \mathbf{y}) : \mu(\mathbf{x}, \mathbf{y}) \in \mathcal{F}(G) \Big\}.$$

3.1.2 General Algorithm

The B&B algorithm keeps a list of assignment problems obtained by ignoring the follower's objective function along with imposing and forbidding some of the follower's edges. Each such problem corresponds to an unfathomed node of the search tree. In particular, the root node represents all possible perfect matchings in the bipartite graph G. In the following we discuss the details of each step of the algorithm separately and conclude the subsection by presenting the outline of the approach.

Let $\mathcal{E}$ and $\bar{\mathcal{E}}$ be two disjoint subsets of the follower's edges, i.e., $\mathcal{E} \cup \bar{\mathcal{E}} \subseteq E_f$ and $\mathcal{E} \cap \bar{\mathcal{E}} = \emptyset$. Define $\mathcal{P}$ to be a nonempty subset of $\mathcal{M}(G)$ of the form

$$\mathcal{P} = \Big\{ \mu \in \mathcal{M}(G) : \mu \cap \mathcal{E} = \mathcal{E} \text{ and } \mu \cap \bar{\mathcal{E}} = \emptyset \Big\}, \tag{3.4}$$

where for each matching $\mu \in \mathcal{P}$ it is required that edges in $\mathcal{E}$ are contained in μ, i.e., $\mathcal{E} \subseteq \mu$, while edges in $\bar{\mathcal{E}}$ are excluded from μ. Then for notational brevity we re-write (3.4) as

$$\mathcal{P} = \big\{ \mathcal{E}; \bar{\mathcal{E}} \big\},$$

where $\mathcal{E}$ and $\bar{\mathcal{E}}$ are often referred to as the sets of *imposed* and *forbidden* edges, respectively, see, e.g., [15].

Within our branch-and-bound algorithm each node of the search tree is associated with a subset of perfect matchings $\mathcal{P}$. Hence, we define a single level-relaxation of BAP with respect to $\mathcal{P}$ as:

$$\big[\mathrm{AP}(\mathcal{P})\big] \qquad \min_{\mathbf{x},\mathbf{y}} \quad \Big\{ g_\ell(\mathbf{x},\mathbf{y}) : \mu(\mathbf{x},\mathbf{y}) \in \mathcal{P} \Big\},$$

which is obtained by removing the follower's objective function and enforcing that each matching should belong to $\mathcal{P}$. The root node of the B&B tree corresponds to $\mathcal{P}^0 = \{\emptyset; \emptyset\}$. Note that $\mathcal{P}^0 = \mathcal{M}(G)$. Thus, $\mathrm{AP}(\mathcal{M}(G))$ is simply a single-level relaxation of BAP.

Bounding. Let $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ be an optimal solution to $\mathrm{AP}(\mathcal{P})$. Clearly, $g_\ell(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ serves as a lower bound for BAP at the B&B node associated with $\mathcal{P}$. If $\bar{\mathbf{y}} \in \mathcal{R}(\bar{\mathbf{x}})$, then $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a feasible solution to (3.1), and $g_\ell(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is an upper bound on the optimal value of BAP. Otherwise (i.e., $\bar{\mathbf{y}} \notin \mathcal{R}(\bar{\mathbf{x}})$), one can always find another $\widetilde{\mathbf{y}} \in \mathcal{R}(\bar{\mathbf{x}})$ such that $(\bar{\mathbf{x}}, \widetilde{\mathbf{y}})$ is a feasible solution to (3.1), which also results in a valid upper bound given by $g_\ell(\bar{\mathbf{x}}, \widetilde{\mathbf{y}})$. Moreover, the following result holds:

Proposition 5. *For a given $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \{0,1\}^{m_\ell + m_f}$, if $\bar{\mathbf{y}} \notin \mathcal{R}(\bar{\mathbf{x}})$, then $\bar{\mathbf{y}}$ cannot belong to any feasible solution of BAP, i.e., the following inequality is valid:*

$$\sum_{e \in \mu_f(\bar{\mathbf{y}})} y_e \leq |\mu_f(\bar{\mathbf{y}})| - 1$$

or, equivalently,

$$\mathcal{F}(G) \subseteq \mathcal{M}(G) \setminus \Gamma(\bar{\mathbf{y}}) \tag{3.5}$$

where $\Gamma(\bar{\mathbf{y}}) = \Big\{ \mu \in \mathcal{M}(G) : \mu_f(\bar{\mathbf{y}}) \subseteq \mu \Big\}$.

Next, we discuss our branching technique that induces $\Gamma(\mathbf{y})$ as one of the branches, which is immediately pruned.

Branching. Murthy [50] describes an algorithm for determining a ranked set of solutions to the assignment problem using partitioning with respect to a minimum cost assignment. Our branching technique follows the same idea. However, we branch with respect to the follower's matching, which allows us to create a fewer number of branches and prune a larger set of perfect matchings.

Formally, let $\mu \in \mathcal{P} = \{\mathcal{E}; \bar{\mathcal{E}}\}$ be a perfect matching obtained by solving AP($\mathcal{P}$) and $\mu_f \subseteq \mu$ be the corresponding follower's matching, i.e., $\mu_f = \mu \cap E_f$. If $\mu_f \setminus \mathcal{E}$ is a nonempty subset of edges, i.e.,

$$\mu_f \setminus \mathcal{E} = \Big\{(t_1, s_1), \ldots, (t_q, s_q)\Big\}$$

for some integer $q \geq 1$, then $\mathcal{P}$ can be partitioned with respect to μ_f as the union of subsets $\mathcal{P}_1, \mathcal{P}_2, \ldots$ and $\mathcal{P}_{q+1}$ that are mutually disjoint, where

$$\begin{aligned}
\mathcal{P}_1 &= \Big\{\mathcal{E}; \bar{\mathcal{E}} \cup (t_1, s_1)\Big\},\\
\mathcal{P}_2 &= \Big\{\mathcal{E} \cup (t_1, s_1); \bar{\mathcal{E}} \cup (t_2, s_2)\Big\},\\
\mathcal{P}_3 &= \Big\{\mathcal{E} \cup (t_1, s_1) \cup (t_2, s_2); \bar{\mathcal{E}} \cup (t_3, s_3)\Big\},\\
&\vdots\\
\mathcal{P}_q &= \Big\{\mathcal{E} \cup (t_1, s_1) \cup (t_2, s_2) \cup \ldots \cup (t_{q-1}, s_{q-1}); \bar{\mathcal{E}} \cup (t_q, s_q)\Big\},\\
\mathcal{P}_{q+1} &= \Big\{\mathcal{E} \cup (t_1, s_1) \cup (t_2, s_2) \cup \ldots \cup (t_q, s_q); \bar{\mathcal{E}}\Big\},
\end{aligned}$$

or, equivalently, $\mathcal{P}_v = \{\mathcal{E}_v; \bar{\mathcal{E}}_v\}$, $v = 1, \ldots, q+1$, and

$$\mathcal{E}_v = \mathcal{E} \cup \big\{(t_i, s_i) \mid i = 1, \ldots, v-1\big\} \quad \text{and} \quad \bar{\mathcal{E}}_v = \bar{\mathcal{E}} \cup (t_v, s_v), \quad v = 1, \ldots, q+1, \tag{3.6}$$

where it is assumed that $(t_{q+1}, s_{q+1}) = \emptyset$. Then

$$\mathcal{P} = \cup_{v=1}^{q+1} \mathcal{P}_v.$$

For a given $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \{0,1\}^{m_\ell + m_f}$, if $\bar{\mathbf{y}} \notin \mathcal{R}(\bar{\mathbf{x}})$, then we define $\mu_f = \mu_f(\bar{\mathbf{y}})$ and create q new nodes of the B&B tree with the corresponding sets of imposed and forbidden edges given by $\mathcal{E}_v$ and $\bar{\mathcal{E}}_v$, respectively, for each $v = 1, \ldots, q$. Note that $\mathcal{P}_{q+1} = \Gamma(\bar{\mathbf{y}})$. Thus, based on Proposition 5, $\mathcal{P}_{q+1}$ is pruned immediately. Figure 2 provides an illustration of the branching process.

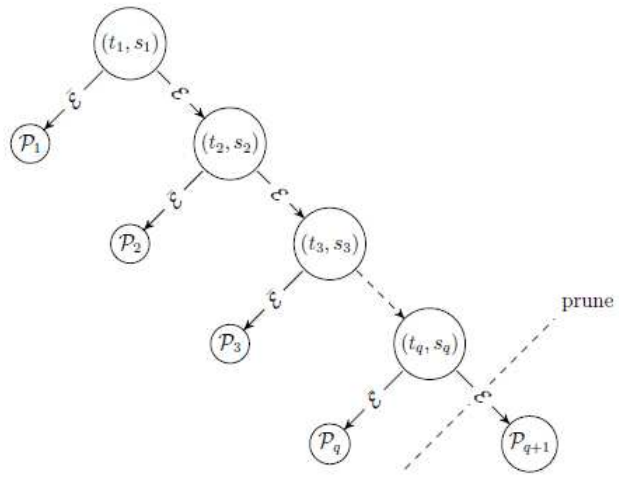


Figure 2: An illustration of q nodes created for given matching $\{(t_1, s_1), \ldots, (t_q, s_q)\}$.

Algorithm Outline. For a node $\mathcal{N}^\nu$ of the branch-and-bound tree, let z^ν denote the optimal value of the corresponding assignment problem AP($\mathcal{P}^\nu$), where $\mathcal{P}^\nu = \left\{\mathcal{E}^\nu; \bar{\mathcal{E}}^\nu\right\}$. Root node $\mathcal{N}^0$ is associated with $\mathcal{P}^0 = \{\emptyset; \emptyset\}$, i.e., $\mathcal{P}^0 = \mathcal{M}(G)$. Let $\mathcal{Q}$ be a set of unprocessed nodes (i.e., nodes that have not been pruned nor branched on) of the B&B tree, and z_u be the current upper bound on the optimal value z^*.

Algorithm 1. *Combinatorial branch-and-bound algorithm to solve BAP*

Step 0: (Initialization) Create a node $\mathcal{N}^0$, let $\mathcal{E}^0 = \bar{\mathcal{E}}^0 = \emptyset$ and $\mathcal{P}^0 = \{\emptyset; \emptyset\}$. Initialize list $\mathcal{Q} \leftarrow \{\mathcal{N}^0\}$ and the global upper bound $z_u = +\infty$.

Step 1: (Node selection) If $\mathcal{Q} = \emptyset$, terminate with the optimal objective function value z_u; otherwise, select and delete node $\mathcal{N}^\eta$ from $\mathcal{Q}$.

Step 2: (Bounding) Solve $AP(\mathcal{P}^\eta)$ for node $\mathcal{N}^\eta$. Denote by $(\mathbf{x}^\eta, \mathbf{y}^\eta)$ and z^η its optimal solution and the objective function value, respectively.

Step 3: (Pruning)

(3a) If $z^\eta \geq z_u$, go to Step 1.

(3b) If $\mathbf{y}^\eta \in \mathcal{R}(\mathbf{x}^\eta)$, then let $z_u = z^\eta$ and go to Step 1.

(3c) If $\mathbf{y}^\eta \notin \mathcal{R}(\mathbf{x}^\eta)$, then find $\widetilde{\mathbf{y}} \in \mathcal{R}(\mathbf{x}^\eta)$ and update (if necessary) the global upper bound. Go to Step 4.

Step 4: (Branching) From $\mathcal{P}^\eta = \{\mathcal{E}^\eta; \bar{\mathcal{E}}^\eta\}$ construct $\mathcal{P}_1^\eta = \{\mathcal{E}_1^\eta; \bar{\mathcal{E}}_1^\eta\}, \ldots, \mathcal{P}_q^\eta = \{\mathcal{E}_q^\eta; \bar{\mathcal{E}}_q^\eta\}$ with respect to $\mu_f(\mathbf{y}^\eta)$ using (3.6). Add the corresponding new nodes $\mathcal{N}_1^\eta, \ldots, \mathcal{N}_q^\eta$ to $\mathcal{Q}$ and go to Step 1.

In Step 0, the algorithm initializes the root node by $\mathcal{E} = \bar{\mathcal{E}} = \emptyset$. In Step 1, a node $\mathcal{N}^\eta$ is chosen from $\mathcal{Q}$ according to some node selection rule. Step 2 solves AP($\mathcal{P}^\eta$). Step 3a checks whether the considered node is promising based on the obtained solution of AP($\mathcal{P}^\eta$) and the current value of the global upper bound. If this is the case, then Step 3b verifies whether the follower's solution $\mathbf{y}^\eta$ is optimal for the lower-level problem. Specifically, if it is optimal, then the algorithm updates the global upper bound. Otherwise, in Step 4 the algorithm branches by creating new nodes of the search tree using to the follower's matching $\mu_f(\mathbf{y}^\eta)$.

Generally speaking, our combinatorial branch-and-bound algorithm can handle any 0–1 function, e.g., polynomials, for the leader's and the follower's objective functions of the form:

$$g(\mathbf{z}) = \sum_{S \subseteq E} p_S \prod_{e \in S} z_e \tag{3.7}$$

where $\prod_{e \in \emptyset} z_e = 1$, $p_S \in \mathbb{R}^1$ and $z_e \in \{0, 1\}$ for all $S \subseteq E$. Note that any pseudo-boolean function can be uniquely represented in form (3.7) [11].

The performance of our B&B approach is affected by the tractability of AP($\mathcal{P}$) in Step 2 and the efficiency of checking whether the obtained follower's solution is optimal for the lower-level optimization problem in Step 3b. Clearly, all of the above depend on the types of the leader's and the follower's objective functions considered. Next, we describe implementation issues for the objective functions given by (3.2) and (3.3) as well as several enhancements that improve the overall performance of the algorithm.

3.1.3 Implementation and Enhancements

If $g_\ell(\mathbf{x}, \mathbf{y})$ is in either a linear sum or linear bottleneck form (3.2), then AP($\mathcal{M}(G)$) is polynomially solvable as it reduces to the linear sum assignment problem (LSAP) or linear bottleneck assignment problem (LBAP), respectively [15]. In order to solve AP($\mathcal{P}$) for general $\mathcal{P} = \{\mathcal{E}; \bar{\mathcal{E}}\}$ one simple approach is to set $d_{ij} = +\infty$ for all $(i, j) \in \bar{\mathcal{E}}$ and remove the

vertices in the set $\{i \in V, j \in U \mid \exists (i,j) \in \mathcal{E}\}$ from G together with their incident edges. Then the obtained problem becomes either LSAP or LBAP, which can be solved efficiently using known algorithms.

Given a solution of AP($\mathcal{P}$), one way to check its optimality with respect to the follower's objective is to solve the lower-level optimization problem (3.1b)-(3.1e) for a given leader's decision and compare the objective function values. Also, the number of branches created in Step 4 depends on the size of the follower's matching, and based on Proposition 5 we prune one of those branches. However, if optimality of the current solution can be verified with less effort than solving the lower-level problem and/or branching can be done with respect to a smaller size matching, then the overall solution time can be improved. In the following, we discuss how to achieve these goals for BAPs with the follower's objective function in either a linear sum or a linear bottleneck form.

In the literature (see, e.g., [15]), primal approaches for LSAP and LBAP often start with a perfect matching (i.e., a feasible solution) and incrementally improve the objective function value by finding an *alternating cycle* with respect to the current perfect matching until it reaches the global optimum. Recall that an alternating cycle with respect to some perfect matching μ is a cycle, the edges of which are alternately in μ and not in μ, and swapping the edges in μ with edges not in μ results in a better feasible solution [15]. Next, we exploit alternating cycles to strengthen our result from Proposition 5 as follows:

Proposition 6. *For a given $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \{0,1\}^{m_\ell + m_f}$, if there exists an alternating cycle with respect to $\mu_f(\bar{\mathbf{y}})$, say $C_{\mu_f(\bar{\mathbf{y}})}$, then let $S = \mu_f(\bar{\mathbf{y}}) \cap C_{\mu_f(\bar{\mathbf{y}})}$, and the following inequality is valid:*

$$\sum_{e \in S} x_e \leq |S| - 1.$$

or, equivalently,

$$\mathcal{F}(G) \subseteq \mathcal{M}(G) \setminus \Gamma(S),$$

where $\Gamma(S) = \Big\{\mu \in \mathcal{M}(G) : S \subseteq \mu\Big\}$.

Note that $2 \leq |S| \leq |\mu_f(\mathbf{y})|$ and $\Gamma(\mu_f(\mathbf{y})) \subseteq \Gamma(S)$. Proposition 6 implies that the problem of checking optimality of the lower-level optimization problem reduces to the problem of finding an alternating cycle. If an alternating cycle does not exist, then the current solution is optimal. Otherwise, an alternating cycle is used to create a fewer number of branches and prune a larger set of perfect matchings than the earlier approach based on Proposition 5. Observe that one should prefer finding smaller sets S, as it results in larger sets $\Gamma(S)$.

We use the labeling algorithm in [4] for LSAP and the threshold algorithm in [30] for LBAP to construct set S. Specifically, for the threshold algorithm let $\mathbf{y}$ be a follower's decision and $t = \max_{(i,j)\in E_f}\{c_{ij}y_{ij}\}$. If there exists a matching, say $\mu_f(\mathbf{y}')$, that saturates the same set of vertices saturated by $\mu_f(\mathbf{y})$, with edges that cost strictly less than t, then $\mathbf{y}$ is not optimal for the lower-level optimization problem. Thus, if we denote $S = \mu_f(\mathbf{y}) \setminus \mu_f(\mathbf{y}')$, then a similar result to Proposition 6 holds here.

3.2 SPECIAL CASES

3.2.1 MIP Formulation

For a general bilevel optimization problem, if the lower-level problem is convex and the regularity conditions are satisfied, then one standard solution method is based on reformulating the original bilevel problem as a single-level mixed integer program by exploiting the Karush-Kuhn-Tucker (KKT) optimality conditions [22]. Next, we follow this approach to present an MIP reformulation of BAPs with the follower's objective function (3.1b) in a linear sum form, i.e., $g_f(\mathbf{y}) = \sum_{(i,j)\in E_f} c_{ij}y_{ij}$. Note that in this case the lower-level problem is LSAP. Its constraint matrix is known to be totally unimodular and the integrality restrictions can be relaxed. Thus, the lower-level problem becomes a simple linear program (LP), which can be represented in terms of its dual as follows:

$$\max_{\lambda,\gamma} \quad \sum_{i\in U}(1 - \sum_{j\in V_\ell^{(i)}} x_{ij})\lambda_i + \sum_{j\in V}(1 - \sum_{i\in U_\ell^{(j)}} x_{ij})\gamma_j \tag{3.8a}$$

$$\text{subject to} \quad \lambda_i + \gamma_j \leq c_{ij} \qquad (i,j) \in E_f. \tag{3.8b}$$

Consequently, BAP can be equivalently rewritten as:

$$\min_{\lambda,\gamma,\mathbf{x},\mathbf{y}} \quad g_\ell(x,y) \tag{3.9a}$$

$$\text{subject to} \quad \sum_{j\in V_\ell^{(i)}} x_{ij} + \sum_{j\in V_f^{(i)}} y_{ij} = 1 \qquad \forall i \in U, \tag{3.9b}$$

$$\sum_{i\in U_\ell^{(j)}} x_{ij} + \sum_{i\in U_f^{(j)}} y_{ij} = 1 \qquad \forall j \in V, \tag{3.9c}$$

$$\lambda_i + \gamma_j \le c_{ij} \qquad \forall (i,j) \in E_f, \tag{3.9d}$$

$$\sum_{(i,j)\in E_f} c_{ij} y_{ij} = \sum_{i=1}^{n} (1 - \sum_{j\in V_\ell^{(i)}} x_{ij})\lambda_i + \sum_{j=1}^{n} (1 - \sum_{i\in U_\ell^{(j)}} x_{ij})\gamma_j, \tag{3.9e}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E_\ell, \tag{3.9f}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E_f, \tag{3.9g}$$

where (3.9e) ensures optimality of the lower-level problem by the LP strong duality property.

MIP formulation (3.9) is nonlinear due to the presence of nonlinear terms of the form $x_{ij}\lambda_i$ and $x_{ij}\gamma_j$ in constraint (3.9e). However, since x_{ij} is binary, it can be easily linearized using a standard Big-M method, see, e.g., [32]. We compare the performances of our B&B algorithm and an off-the-shelf MIP solver in Section 3.3 for instances with the follower's linear sum objective.

3.2.2 Follower's Problem and Polynomially Solvable Classes of BAP

If the leader's feasible set of actions is relatively small, then one solution approach for BAP is to solve the follower's problem for each leader's feasible solution and pick the one that returns the most favorable objective function value for the leader. In general, there are $O(2^{m_\ell})$ possible leader's decisions. However, this number can be reduced depending on the graph topology. For example, consider the case when all leader's edges emanate from exactly one node, which implies that the number of feasible leader's action is bounded by $O(m_\ell)$.

Formally, given a feasible leader's solution $\hat{\mathbf{x}}$, define

$$U_f(\hat{\mathbf{x}}) = \{i \in U : \forall j \in V_\ell^{(i)}, \hat{x}_{ij} = 0\} \quad \text{and} \quad V_f(\hat{\mathbf{x}}) = \{j \in V : \forall i \in U_\ell^{(j)}, \hat{x}_{ij} = 0\},$$

i.e., all vertices of G that are not assigned by the leader. Note that $U_f(\hat{\mathbf{x}}) \subseteq U$, $V_f(\hat{\mathbf{x}}) \subseteq V$ and $|U_f(\hat{\mathbf{x}})| = |V_f(\hat{\mathbf{x}})|$. For all $i \in U_f(\hat{\mathbf{x}})$, define $V_f^{(i)}(\hat{\mathbf{x}}) = \{j \in V_f(\hat{\mathbf{x}}) : (i,j) \in E_f\}$. Likewise, we define $U_f^{(j)}(\hat{\mathbf{x}})$ for all $j \in V_f(\hat{\mathbf{x}})$. Denote $E_f(\hat{\mathbf{x}}) = \{(i,j) \in E_f : i \in U_f(\hat{\mathbf{x}}), j \in V_f(\hat{\mathbf{x}})\}$, where $E_f(\hat{\mathbf{x}}) \subseteq E_f$. Then evaluating the leader's objective function $g_\ell(\hat{\mathbf{x}}, \mathbf{y})$ requires solution of the follower's problem:

$$
\begin{aligned}
[\mathrm{BAP}(\hat{\mathbf{x}})] \quad \min_{\mathbf{y}} \quad & g_\ell(\hat{\mathbf{x}}, \mathbf{y}) && \text{(3.10a)}\\
\text{s.t.} \quad \min_{\mathbf{y}} \quad & g_f(\mathbf{y}) && \text{(3.10b)}\\
\text{s.t.} \quad & \sum_{j \in V_f^{(i)}(\hat{\mathbf{x}})} y_{ij} = 1 \quad \forall i \in U_f(\hat{\mathbf{x}}), && \text{(3.10c)}\\
& \sum_{i \in U_f^{(j)}(\hat{\mathbf{x}})} y_{ij} = 1 \quad \forall j \in V_f(\hat{\mathbf{x}}), && \text{(3.10d)}\\
& y_{ij} \in \{0,1\} \quad \forall (i,j) \in E_f(\hat{\mathbf{x}}). && \text{(3.10e)}
\end{aligned}
$$

Observe that problem (3.10b)-(3.10e) is a single-level assignment problem that can be solved by an appropriate method depending on $g_f(\cdot)$; the objective of (3.10a) ensures that if there are multiple optimal solutions for the follower, then we select the most favorable one for the leader, which is consistent with Assumption **A1**, i.e., the optimistic case for BAP. Naturally, the pessimistic case can be handled by replacing minimization with maximization in (3.10a).

Any ranking algorithm [15] can be used to solve (3.10) by enumerating over all optimal solutions for (3.10b)-(3.10e). However, this approach may be computationally expensive as the number of such solutions can be exponentially large. In the following we show that if the follower's objective is in either linear sum or linear bottleneck form, then problem (3.10) can be solved efficiently as long as its single-level relaxation obtained by removing (3.10b) is also efficiently solvable. *This also implies that if m_ℓ is fixed then these classes of BAP are polynomially solvable.* It is interesting to contrast this observation with the result in [42], where it is shown that if the number of variables controlled by the follower is constant, then bilevel linear programs are polynomially solvable.

3.2.2.1 Follower's Objective: Linear Sum

For $g_f(\mathbf{y}) = \sum_{(i,j)\in E_f(\hat{\mathbf{x}})} c_{ij}y_{ij}$, let $(\boldsymbol{\lambda}, \boldsymbol{\gamma})$ be a dual optimal solution to (3.10b)-(3.10e). Define $E(\boldsymbol{\lambda}, \boldsymbol{\gamma}) = \{(i,j) \in E_f(\hat{\mathbf{x}}) : \lambda_i + \gamma_j = c_{ij}, i \in U_f(\hat{\mathbf{x}}), j \in V_f(\hat{\mathbf{x}})\}$. The following result from [27] characterizes all optimal solutions of LSAP:

Lemma 3. *A feasible solution* $\mathbf{y}$ *is optimal to (3.10b)-(3.10e) if and only if* $\mu_f(\mathbf{y}) \subseteq E(\boldsymbol{\lambda}, \boldsymbol{\gamma})$.

Then problem (3.10) can be solved using the following two-step procedure. First, we solve LSAP given by (3.10b)-(3.10e); specifically, we need to obtain a dual optimal solution. Second, we re-define the leader's cost coefficients for edges in $E_f(\hat{\mathbf{x}})$ as:

$$d'_{ij} = \begin{cases} d_{ij} & \text{if } (i,j) \in E(\boldsymbol{\lambda}, \boldsymbol{\gamma}), \\ \infty & \text{otherwise;} \end{cases}$$

and solve a single-level relaxation of (3.10) by removing the follower's objective in (3.10b). Clearly, by Lemma 3 the obtained solution is optimal for (3.10).

Note that this procedure does not depend on a type of the leader's objective. Furthermore, one interesting observation links the above result for another class of assignment problems, namely, singly constrained LSAP (SC-LSAP) [1, 41]. Suppose $g_\ell(\mathbf{y}) = \sum_{(i,j)\in E_f} d_{ij}y_{ij}$. Then (3.10) is reformulated as the following IP:

$$\min_{\mathbf{y}} \quad \sum_{(i,j)\in E_f} d_{ij}y_{ij} \tag{3.11a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E_f} c_{ij}y_{ij} \le c^*, \tag{3.11b}$$

$$\sum_{j\in V_f^{(i)}(\hat{\mathbf{x}})} y_{ij} = 1 \quad \forall i \in U_f(\hat{\mathbf{x}}), \tag{3.11c}$$

$$\sum_{i\in U_f^{(j)}(\hat{\mathbf{x}})} y_{ij} = 1 \quad \forall j \in V_f(\hat{\mathbf{x}}), \tag{3.11d}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in E_f(\hat{\mathbf{x}}), \tag{3.11e}$$

where

$$c^* = \min\Big\{ \sum_{(i,j)\in E_f} c_{ij}y_{ij} : (3.11c) - (3.11e) \Big\}. \tag{3.12}$$

If equation (3.12) is not required to be satisfied, i.e., one considers a general class of SC-LSAPs, then the problem is known to be NP-hard. However, in our case (3.12) holds, which implies that (3.11)-(3.12) forms a polynomially solvable class of SC-LSAPs. We refer the reader to [41] for other examples of efficiently solvable classes of SC-LSAP.

3.2.2.2 Follower's Objective: Linear Bottleneck Characterizing the set of alternative optimal solutions for LBAP is rather straightforward as one should simply omit all edges with their costs greater than the optimal value of LBAP. Formally, denote by c^* the optimal objective function value of (3.10b)-(3.10e) with $g_f(\mathbf{y}) = \max_{(i,j)\in E_f(\hat{\mathbf{x}})}\{c_{ij}y_{ij}\}$. Similar to the previous section, we redefine the leader's cost coefficients for edges in $E_f(\hat{\mathbf{x}})$ as:

$$d'_{ij} = \begin{cases} \infty & \text{if } c_{ij} > c^*, \\ d_{ij} & \text{otherwise.} \end{cases}$$

Then solving single-level relaxation of (3.10) by removing the follower's objective in (3.10b) yields the optimal solution to the problem (3.10) with the follower's bottleneck objective function.

3.2.3 Other Polynomially Solvable Classes

In this section we describe two other classes of BAP that can be solved efficiently. In particular, we exploit the following simple observation. If for all feasible leader's decisions both the leader's and follower's objective function values coincide (subject to an addition of a constant value), i.e., $g_\ell(\hat{\mathbf{x}}, \mathbf{y}) = g_f(\mathbf{y}) + const$, for all feasible $\hat{\mathbf{x}}$, then the follower's objective function can be omitted and BAP reduces to a single-level assignment problem. Furthermore, for linear sum objective functions in BAP there exists a more general sufficient condition:

Proposition 7. *If there exists $k \in \mathbb{R}^1$ such that $d_{ij} = c_{ij} + k$ for all $(i,j) \in E_f$, then BAP with the leader's and follower's linear sum objective functions reduces to LSAP.*

The proof of the result can be derived using the assignment constraints (3.1c)-(3.1d). A similar in spirit result for BAP with bottleneck objective functions is given by:

Proposition 8. *Suppose there exists an ordering of edges in E_f given by $(i_1, j_1), (i_2, j_2), \ldots, (i_{m_f}, j_{m_f})$ such that $c_{i_1,j_1} \leq c_{i_2,j_2} \leq \ldots c_{i_{m_f},j_{m_f}}$ and $d_{i_1,j_1} \leq d_{i_2,j_2} \leq \ldots d_{i_{m_f},j_{m_f}}$. Then BAP with the leader's and follower's linear bottleneck objective functions reduces to LBAP.*

In particular, the latter result follows from the fact the optimal objective function value of LBAP is exactly equal to one of the edge's cost coefficients and the optimal assignment depends only on the relative order of the coefficients but not on their actual numerical values.

Finally, we note the results of Propositions 7 and 8 hold under the assumption that the leader's and the follower's objective functions are in the same form (specifically, either both in a linear sum or both in a linear bottleneck form). In the next section we explore computationally test instances with the objectives' coefficients as in Proposition 7 (referred to as *strongly-correlated* instances) for BAPs, where the leader's and the follower's objectives are not in the same form.

3.3 COMPUTATIONAL EXPERIMENTS

3.3.1 Test Instances and Setup

In our computational study we focus on four classes of BAP, where the leader's and the follower's objective functions are in either a linear sum or a linear bottleneck form. We refer to the resulting optimization problems as *Sum-Sum*, *Bottleneck-Sum*, *Sum-Bottleneck* and *Bottleneck-Bottleneck*, where for each problem the first and second terms correspond to the considered types of the leader's and the follower's objective functions, respectively.

Our test instances are constructed as follows. The leader's and follower's cost matrices are generated with varying degrees of correlation:

- un-correlated: $d_j \sim U[1, 100]$ and $c_j \sim U[1, 100]$,
- weakly-correlated: $d_j \sim U[1, 100]$ and $c_j = \max\{0,\ d_j + U[-10, 10]\}$,
- strongly-correlated: $d_j \sim U[1, 100]$ and $c_j = d_j + 10$,

where U denotes a coefficient generated according to a uniform distribution over the specified interval. We consider test instances with complete bipartite graphs G, i.e., $m_\ell + m_f = n^2$, where $n \in \{25,\ 50,\ 75,\ 100,\ 150,\ 200,\ 250,\ 300\}$. The number of edges controlled by the follower (m_f) is assumed to be some fraction, say ρ, of the total number of edges in G and modeled using a Bernoulli distribution, i.e., $m_f \approx \rho n^2$, where ρ varies over $\{0.1,\ 0.2,\ 0.3, \ldots,\ 0.9\}$. We generate 5 instances for each n, ρ and degree of correlation. More details of test instances can be found in Appendix B. Our test instances are available online [7].

In the computational experiments, we consider three different algorithmic approaches:

(i) CPLEX 12.4 applied to the linearized version of MIP (3.9) (only for the *Sum-Sum* and *Bottleneck-Sum* problems);

(ii) the general branch-and-bound (B&B) algorithm described in Section 3.1.2 (see Algorithm 1); and

(iii) its improved version referred to as IB&B that uses the algorithmic enhancements described in Section 3.1.3.

The algorithms in our study are coded in C++, compiled using Microsoft Visual Studio 2010. For solving the linear sum assignment problem, the *dlib* open source library is used [39]. All experiments are performed on a single thread of a Windows 7 PC with 3.6GHz CPU and 32GB of RAM. Note that in our implementation of the branch-and-bound algorithms (i.e., B&B and IB&B), in Step 1 nodes are selected according to the breadth-first rule.

3.3.2 Results and Discussion

Our computational results are summarized in Tables 6-15. The reported average running times are in seconds. The time limit is set to 3600 seconds. If an algorithm does not find an optimal solution within the time limit, we report the obtained average optimality gap. The superscripts over average time and gap values indicate the number of instances for each type of possible outcome (note that the superscript is dropped if its value is exactly 5). For example, consider an un-correlated class of test instances of the *Sum-Sum* problem in Table 6, where $n = 25$ and $\rho = 0.6$. CPLEX solves four instances to optimality with average

running time of 276 seconds, and for one instance it does not find an optimal solution within the time limit and the optimality gap is 5.6%. On the other hand, B&B solves three instances to optimality with average running time 112 seconds, while for the remaining two instances the gap is 26.8%. Finally, IB&B solves all instances to optimality with average running time of 1 second. The best result for a given test instance is in bold. An entry of "-" in our tables indicates that the corresponding algorithm can not find a feasible integer solution within the time limit for any of the instances in the considered class. Note that for strongly-correlated instances, the results for the *Sum-Sum* and *Bottleneck-Bottleneck* problems are not presented; recall that they are polynomially solvable as shown in Section 3.2.3.

As one would expect, the computational difficulty of the problem decreases in the degree of correlation (between the leader's and follower's cost coefficients), i.e., the un-correlated test instances seems to be the most challenging ones, while the strongly-correlated are much easier to solve. For example, compare the performance of IB&B for the *Bottleneck-Sum* problem with $n = 100$, $\rho = 0.6$ and different degrees of correlation given in Tables 7, 11 and 14. None of the un-correlated instances can be solved within the time limit and the average optimality gap is 90.8%, while four out of five weakly-correlated instances can be solved with the average runtime of 112 seconds and exactly one instance has the optimality gap of 33.3%. All strongly-correlated instances can be solved within 7 seconds. These observations are rather intuitive because the leader's and the follower's objectives are more in line when the degree of correlation increases, which results in a better quality of bounds obtained via a single-level relaxation of BAP.

Another general (and also somewhat expected) observation is that the computational difficulty of the problem typically increases with ρ. This is due to the fact that larger values of ρ result in larger-sized follower's reaction sets, which evidently make BAP more challenging.

Regarding the algorithms developed in this study, IB&B turns out to be the best approach among the considered solutions methods in almost all cases (except for some un-correlated instances discussed below). Specifically, IB&B either reaches the optimal solution faster or

has a tighter optimality gap at the time limit than B&B and CPLEX. Advantages of IB&B in comparison to B&B are rather clear due the nature of the proposed enhancement described in Section 3.1.3. Moreover, the smaller values in the column *#nodes* (i.e., the number of nodes in the search tree) for IB&B provides another supporting argument for its superior performance. As a side note, we should mention that these values are reported only for instances that are solved within the time limit.

Finally, we note that for some un-correlated instances CPLEX obtains better results than IB&B, in particular, see Table 6 with $n = 25$ and $\rho \in \{0.8, 0.9\}$, $n = 50$ and $\rho \in \{0.6, 0.7, 0.9\}$, $n = 75$ and $\rho \in \{0.4, 0.5, 0.6\}$ as well as $n = 100$ and $\rho \in \{0.3, 0.4\}$. As mentioned above, for un-correlated instances the lower bound computed by ignoring the follower's objective can be arbitrarily loose as there is no correlation between the leader's and the follower's cost coefficients. This hinders the performance of IB&B. However, as the number of edges controlled by the follower increases, see instances in Table 6 for $n = 100$ and $\rho \geq 0.5$, IB&B is able to find a better quality feasible solution than CPLEX, which improves the tightness of IB&B's optimality gaps. We attribute this observation to the fact that IB&B constructs feasible solutions at every node of the search tree (see Step 3(c) of Algorithm 1).

3.4 CONCLUDING REMARKS

In this chapter we describe exact solution methods for the bilevel assignment problem. Our approach is based on a branch-and-bound framework and tailored to exploit structural properties of the assignment problem and specific classes of the considered objective functions. Our extensive computational experiments (including comparisons with an off-the-shelf MIP solver) demonstrate both advantages and limitations of the developed algorithms. Furthermore, the bilevel assignment problem is known to be NP-hard. Thus, we also describe some polynomially solvable classes of the problem.

One possible direction of the future research is to relax assumption **A1** and extend our solution methods for the pessimistic case of the bilevel assignment problem. Also, it would be interesting to consider bilevel extensions of other well-known classes of assignment problems.

For instance a bilevel extension of multi-index assignment problems can be appropriate to model a decentralized multi-target tracking [49]. Finally, we note that our combinatorial branch-and-bound framework can potentially handle more general types of the objective functions than those considered in this study.

3.5 ACKNOWLEDGMENT

The content of this chapter is based upon work supported by the AFRL Mathematical Modeling and Optimization Institute.

Table 6: Results for un-correlated instances - *Sum-Sum* problem.

		Sum-Sum							
		CPLEX		B&B			IB&B		
n	ρ	time	gap	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		< 1		1	< 1		
	0.2	< 1		< 1		1	< 1		1
	0.3	< 1		< 1		14	< 1		2
	0.4	< 1		< 1		121	< 1		8
25	0.5	72		23		4061	$< \mathbf{1}$		638
	0.6	276[4]	5.6%[1]	112[3]	28.6%[2]	59535	**1**		14834
	0.7	1399[1]	12.8%[4]		38.6%		**105**[4]	**0.3**%[1]	473071
	0.8		**16.9**%		48.0%			19.5%	
	0.9	**1982**[1]	**32.1**%[4]		83.3%			55.0%	
	0.1	< 1		< 1		22	< 1		1
	0.2	< 1		< 1		120	< 1		2
	0.3	388		421[4]	21.4%[1]	45482	$< \mathbf{1}$		258
	0.4	2891[1]	8.5%[4]		43.0%		**44**		172057
50	0.5		19.0%		80.3%		**1557**[1]	**38.1**%[4]	2291362
	0.6		**33.8**%		83.3%			58.4%	
	0.7		**53.6**%		86.6%			68.7%	
	0.8		73.7%[4]		88.4%			**79.3**%	
	0.9		**76.8**%		90.3%			85.1%	
	0.1	< 1		< 1		1	< 1		1
	0.2	9		4		904	$< \mathbf{1}$		8
	0.3		4.5%		60.8%		**61**		120271
	0.4		**13.7**%		**81.5**%			41.3%	
75	0.5		**47.5**%		85.6%			69.4%	
	0.6		**74.6**%		87.3%			76.7%	
	0.7		86.4%		89.0%			**82.2**%	
	0.8		89.1%		90.9%			**86.1**%	
	0.9		-		92.1%			**89.3**%	
	0.1	< 1		< 1		4	< 1		1
	0.2	721[4]	0.8%[1]	631[2]	69.7%[3]	1015	$< \mathbf{1}$		304
	0.3		**9.2**%		79.4%			35.8%	
	0.4		**43.4**%		84.8%			62.8%	
100	0.5		78.3%		87.1%			**75.0**%	
	0.6		89.9%		89.2%			**82.4**%	
	0.7		92.5%		90.2%			**86.4**%	
	0.8		94.2%[1]		92.1%			**89.7**%	
	0.9		-		92.9%			**91.1**%	

Table 7: Results for un-correlated instances - *Bottleneck-Sum* problem.

		Bottleneck-Sum							
		CPLEX		B&B			IB&B		
n	ρ	time	gap	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		< 1		1	< 1		1
	0.2	< 1		< 1		1	< 1		1
	0.3	1		$< \mathbf{1}$		124	$< \mathbf{1}$		5
	0.4	5		7		5229	$< \mathbf{1}$		35
25	0.5	113		7^{3}	17.9%^{2}	6614	**2**		1114
	0.6	1052^{3}	11.9%^{2}	3570^{1}	42.3%^{4}	497540	**38**		32305
	0.7	2027^{2}	23.6%^{3}		61.5%		$\mathbf{176}^{2}$	**26.5**%^{3}	133773
	0.8	$\mathbf{2267}^{2}$	**29.0**%^{3}		67.3%			42.5%	
	0.9	$\mathbf{1525}^{3}$	**8.7**%^{2}		74.8%			68.7%	
	0.1	2		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.2	15		1		19	$< \mathbf{1}$		5
	0.3	122^{4}	20.0%^{1}	12^{4}	84.3%^{1}	459	**1**		48
	0.4		24.1%		79.7%		**362**		42309
50	0.5		84.1%		84.9%			**80.1**%	
	0.6		86.9%		86.5%			**83.8**%	
	0.7		87.3%		88.2%			**86.4**%	
	0.8		88.8%		89.3%			**87.9**%	
	0.9		87.0%^{3}		90.2%			**89.6**%	
	0.1	21		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.2	587		134		1745	**1**		13
	0.3	3485^{1}	53.2%^{4}		84.7%^{4}		$\mathbf{7}^{4}$	**77.8**%^{1}	141
	0.4		58.8%		87.7%		$\mathbf{516}^{2}$	**83.4**%^{3}	20862
75	0.5		89.5%		89.9%			**87.9**%	
	0.6		90.6%		90.4%			**89.5**%	
	0.7		92.6%		90.7%			**90.1**%	
	0.8		92.3%^{3}		90.8%			**90.5**%	
	0.9		90.9%^{1}		91.4%			**91.1**%	
	0.1	100		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.2		90.9%	68^{2}	87.1%^{2}	1498	**5**		32
	0.3		90.4%	2997^{1}	88.4%^{4}	40811	$\mathbf{37}^{3}$	**72.7**%^{2}	266
	0.4		90.1%		90.3%		$\mathbf{1075}^{2}$	**87.6**%^{3}	15952
100	0.5		92.1%		91.3%			**89.9**%	
	0.6		93.8%		92.0%			**90.8**%	
	0.7		95.3%^{4}		92.1%			**91.7**%	
	0.8		92.5%^{3}		92.3%			**92.1**%	
	0.9		-		92.4%			**92.2**%	

Table 8: Results for un-correlated instances - *Sum-Bottleneck* problem.

		Sum-Bottleneck					
		B&B			IB&B		
n	ρ	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		13	< 1		1
	0.4	< 1		69	< 1		9
25	0.5	3		3290	$<$ **1**		511
	0.6	36[4]	34.2%[1]	23663	**94**		79595
	0.7		36.1%		**1197**[1]	**24.8**%[4]	551018
	0.8		56.4%			**42.3**%	
	0.9		82.8%			**66.0**%	
	0.1	< 1		1	< 1		1
	0.2	< 1		4	< 1		2
	0.3	4		1550	$<$ **1**		133
	0.4	1067[4]	54.9%[1]	82388	**23**[4]	**39.1**%	17218
50	0.5		80.2%			**51.2**%	
	0.6		84.1%			**67.3**%	
	0.7		85.0%			**75.2**%	
	0.8		87.9%			**81.8**%	
	0.9		90.7%			**88.0**%	
	0.1	< 1		1	< 1		1
	0.2	1		146	$<$ **1**		8
	0.3	17[1]	76.0%[4]	2954	**16**		6137
	0.4		81.2%			**60.9**%	
75	0.5		84.5%			**76.6**%	
	0.6		88.2%			**82.8**%	
	0.7		89.0%			**84.7**%	
	0.8		90.5%			**87.4**%	
	0.9		92.0%			**90.2**%	
	0.1	< 1		2	< 1		1
	0.2	11[3]	72.3%[2]	1182	$<$ **1**		53
	0.3		79.0%			**56.6**%	
	0.4		83.2%			**71.5**%	
100	0.5		86.8%			**79.5**%	
	0.6		89.3%			**84.4**%	
	0.7		90.6%			**87.9**%	
	0.8		91.7%			**90.0**%	
	0.9		92.9%			**91.8**%	

Table 9: Results for un-correlated instances - *Bottleneck-Bottleneck* problem.

		Bottleneck-Bottleneck					
		B&B			IB&B		
n	ρ	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		9	< 1		2
	0.4	1		518	$< \mathbf{1}$		152
25	0.5	622		75869	**4**		2142
	0.6	271^{2}	$45.8\%^{3}$	111914	**1054**		427064
	0.7		56.3%			**54.9**%	
	0.8		**62.5**%			63.1%	
	0.9		72.1%			**71.7**%	
	0.1	< 1		2	< 1		1
	0.2	< 1		6	< 1		4
	0.3	3		82	**1**		38
	0.4	$\mathbf{23}^{2}$	$77.2\%^{3}$	791	68^{2}	$\mathbf{75.6\%}^{\mathbf{3}}$	4700
50	0.5		**81.1**%			84.0%	
	0.6		86.1%			**85.2**%	
	0.7		87.6%			**87.0**%	
	0.8		**88.6**%			88.7%	
	0.9		90.1%			90.1%	
	0.1	< 1		1	< 1		1
	0.2	121		1356	1		12
	0.3	22^{2}	$84.5\%^{3}$	477	31^{3}	$86.7\%^{2}$	1093
	0.4	3083^{1}	$88.1\%^{4}$	94353	$\mathbf{284}^{1}$	$\mathbf{87.6\%}^{4}$	4264
75	0.5		89.6%			**89.2**%	
	0.6		90.2%			**90.0**%	
	0.7		90.3%			**90.2**%	
	0.8		90.8%			**90.5**%	
	0.9		91.3%			**91.2**%	
	0.1	< 1		1	< 1		1
	0.2	107		2178	**7**		41
	0.3	953^{2}	$86.7\%^{3}$	16357	$\mathbf{428}^{4}$	$\mathbf{91.9\%}^{1}$	8326
	0.4		90.0%		$\mathbf{338}^{1}$	$\mathbf{89.3\%}^{4}$	1872
100	0.5		91.0%			**90.9**%	
	0.6		91.5%			**91.4**%	
	0.7		91.9%			91.9%	
	0.8		92.3%			**92.2**%	
	0.9		92.3%			92.3%	

Table 10: Results for weakly-correlated instances - *Sum-Sum* problem.

		Sum-Sum							
		CPLEX		B&B			IB&B		
n	ρ	time	gap	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		< 1		1	< 1		1
	0.2	< 1		< 1		1	< 1		1
	0.3	< 1		< 1		1	< 1		1
	0.4	< 1		< 1		1	< 1		1
25	0.5	< 1		< 1		1	< 1		1
	0.6	< 1		< 1		1	< 1		1
	0.7	< 1		< 1		3	< 1		2
	0.8	< 1		< 1		17	< 1		3
	0.9	2		$\mathbf{< 1}$		431	$\mathbf{< 1}$		65
	0.1	< 1		< 1		1	< 1		1
	0.2	< 1		< 1		1	< 1		1
	0.3	< 1		< 1		1	< 1		1
	0.4	< 1		< 1		1	< 1		1
50	0.5	< 1		< 1		7	< 1		1
	0.6	31		1		993	$\mathbf{< 1}$		26
	0.7	511		426		36408	$\mathbf{< 1}$		577
	0.8		10.1%[3]		4.5%		**743**		424018
	0.9		15.3%		14.1%			**7.3**%	
	0.1	< 1		< 1		1	< 1		1
	0.2	< 1		< 1		1	< 1		1
	0.3	1		$\mathbf{< 1}$		1	$\mathbf{< 1}$		1
	0.4	1		$\mathbf{< 1}$		6	$\mathbf{< 1}$		1
75	0.5	6		2		654	$\mathbf{< 1}$		14
	0.6	662[3]	3.1%[2]	1[2]	3.6%[3]	158	**5**		7090
	0.7		7.3%	1201[1]	8.3%[4]	64531	**338**[4]	**4.3**%[1]	215096
	0.8		49.0%[3]		12.9%		**229**[1]	**8.0**%[4]	246395
	0.9		84.2%[1]		19.9%			**16.0**%	
	0.1	< 1		< 1		1	< 1		1
	0.2	1		$\mathbf{< 1}$		1	$\mathbf{< 1}$		1
	0.3	2		$\mathbf{< 1}$		9	$\mathbf{< 1}$		2
	0.4	8		$\mathbf{< 1}$		73	$\mathbf{< 1}$		8
100	0.5	622[4]	1.6%[1]	111[4]	2.2%[1]	7638	$\mathbf{< 1}$		357
	0.6		6.0%		4.3%		**153**[4]	**2.5**%[1]	110674
	0.7		40.2%		13.9%			**9.3**%	
	0.8		84.0%		20.8%			**16.3**%	
	0.9		-		26.5%			**24.3**%	

Table 11: Results for weakly-correlated instances - *Bottleneck-Sum* problem.

		Bottleneck-Sum							
		CPLEX		B&B			IB&B		
n	ρ	time	gap	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		< 1		1	< 1		1
	0.2	< 1		< 1		1	< 1		1
	0.3	< 1		< 1		2	< 1		1
	0.4	< 1		< 1		2	< 1		1
25	0.5	1		$< \mathbf{1}$		5	$< \mathbf{1}$		2
	0.6	2		$< \mathbf{1}$		10	$< \mathbf{1}$		6
	0.7	3		$< \mathbf{1}$		29	$< \mathbf{1}$		10
	0.8	3		1		740	$< \mathbf{1}$		83
	0.9	8		10^4	11.5%[1]	7034	**6**		3544
	0.1	1		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.2	6		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.3	8		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.4	34		$< \mathbf{1}$		3	$< \mathbf{1}$		1
50	0.5	63		$< \mathbf{1}$		9	$< \mathbf{1}$		3
	0.6	107		8		269	**1**		92
	0.7	1133[3]	10.0%[2]	78[2]	14.4%[3]	6276	**28**		2236
	0.8		74.7%		29.9%		**512**[3]	**21.5**%[2]	42869
	0.9		44.9%		32.5%			**26.2**%	
	0.1	14		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.2	53		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.3	181		$< \mathbf{1}$		3	$< \mathbf{1}$		2
	0.4	248		1		3	$< \mathbf{1}$		3
75	0.5	776		41		982	**2**		25
	0.6	805[2]	39.4%[3]	207[3]	20.2%[2]	3695	**16**		222
	0.7	1637[2]	85.5%[3]	1111[2]	27.9%[3]	37685	**92**[3]	**27.3**%[2]	1662
	0.8		92.5%[4]	1066[2]	25.9%[3]	100894	**192**[2]	**21.9**%[3]	3889
	0.9		-		27.2%		**681**[1]	**24.2**%[4]	13473
	0.1	58		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.2	361		$< \mathbf{1}$		1	$< \mathbf{1}$		1
	0.3	1378		1		2	$< \mathbf{1}$		2
	0.4	1872[3]	92.7%[2]	20		189.8	**2**		9
100	0.5	2293[2]	93.8%[3]	503[4]	25.0%[1]	7551	**36**		180
	0.6		94.0%		25.6%		**112**[4]	**33.3**%[1]	672
	0.7		94.0%		34.8%		**1008**[1]	**29.9**%[4]	6732
	0.8		93.9%[3]		30.3%		**123**[1]	**34.9**%[4]	585
	0.9		94.4%[2]		36.6%			**33.8**%	

Table 12: Results for weakly-correlated instances - *Sum-Bottleneck* problem.

		Sum-Bottleneck					
		B&B			IB&B		
n	ρ	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
25	0.5	< 1		1	< 1		1
	0.6	< 1		1	< 1		1
	0.7	< 1		5	< 1		3
	0.8	< 1		12	< 1		8
	0.9	8		3384	**5**		2540
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
50	0.5	< 1		2	< 1		1
	0.6	1		205	$< \mathbf{1}$		92
	0.7	2		281	**1**		231
	0.8	195^{4}	22.2%^{1}	15903	$\mathbf{97}^{4}$	**21.9**%^{1}	13123
	0.9	4^{1}	27.2%^{4}	383	$\mathbf{3}^{1}$	**22.4**%^{4}	305
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
75	0.5	< 1		3	< 1		3
	0.6	26		1607	**13**		1151
	0.7	453		15389	**119**		8676
	0.8	1179^{2}	16.8%^{3}	27711	$\mathbf{1476}^{4}$	**24.0**%^{1}	59547
	0.9	5^{1}	33.5%^{4}	201	5^{1}	33.5%^{4}	214
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		4	< 1		3
100	0.5	7		427	**2**		135
	0.6	134		4249	**45**		2454
	0.7	95^{4}	25.3%^{1}	1084	$\mathbf{35}^{4}$	25.3%^{1}	3644
	0.8	1^{1}	32.5%^{4}	15	1^{1}	**32.2**%^{4}	20
	0.9		43.1%			**42.8**%	

Table 13: Results for weakly-correlated instances - *Bottleneck-Bottleneck* problem.

		Bottleneck-Bottleneck					
		B&B			IB&B		
n	ρ	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
25	0.5	< 1		1	< 1		1
	0.6	< 1		3	< 1		2
	0.7	< 1		153	< 1		26
	0.8	1		172	1		217
	0.9	195		39579	**127**		35660
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		2	< 1		2
50	0.5	< 1		2	< 1		2
	0.6	44		3218	**1**		22
	0.7	289^{4}	23.1%^{1}	16042	$\mathbf{54}^{4}$	23.1%^{1}	3473
	0.8	4^{2}	34.1%^{3}	121	$\mathbf{2}^{2}$	**33.5**%^{3}	37
	0.9	$< 1^{1}$	45.8%^{4}	2	$< 1^{1}$	45.8%^{4}	2
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		3	< 1		1
75	0.5	14		270	**1**		7
	0.6	140		3317	**2**		16
	0.7	8^{4}	30.8%^{1}	106	$\mathbf{8}^{4}$	**25.0**%^{1}	69
	0.8	272^{2}	31.9%^{3}	5648	$\mathbf{120}^{3}$	**36.8**%^{2}	1492
	0.9		42.3%			**41.5**%	
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	2		9	1		3
100	0.5	148		2607	**34**		105
	0.6	452^{4}	38.5%^{1}	6213	$\mathbf{369}^{4}$	**36.4**%^{1}	1810
	0.7	1892^{1}	44.9%^{4}	2321	$\mathbf{277}^{2}$	**45.1**%^{3}	13750
	0.8		45.7%			**44.1**%	
	0.9		47.6%			47.6%	

Table 14: Results for strongly-correlated instances - *Bottleneck-Sum* problem.

		Bottleneck-Sum							
		CPLEX		B&B			IB&B		
n	ρ	time	gap	time	gap	#nodes	time	gap	#nodes
	0.1	74		< 1		1	< 1		1
	0.2	338		< 1		1	< 1		1
	0.3	454		< 1		1	< 1		1
	0.4	1508		< 1		2	< 1		1
100	0.5	2483[3]	93.3%[2]	2		10	**1**		5
	0.6		91.5%	10		293	**7**		45
	0.7		91.1%[4]	72		2803	**19**		108
	0.8		92.9%[2]	541		24588	**12**		57
	0.9		89.8%[1]	2[4]	20.0%[1]	8	**82**		493
	0.1	345		< 1		1	< 1		1
	0.2	1360		< 1		1	< 1		1
	0.3	1947[4]	98.0%[1]	< 1		1	< 1		1
	0.4	2488[3]	98.8%[2]	25		10	**1**		2
150	0.5		98.8%[3]	26		15	**7**		9
	0.6		98.8%[4]	43		24	**24**		28
	0.7		98.9%[3]	**12**		6	17		21
	0.8		-	45		29	**11**		12
	0.9		-	19		17	**6**		7
	0.1	1622[4]	97.8%[1]	< 1		1	< 1		1
	0.2	2175[3]	98.9%[2]	< 1		1	< 1		1
	0.3		99.3%	< 1		1	< 1		1
	0.4		99.0%[4]	< 1		1	< 1		1
200	0.5		99.0%[4]	47		23	**46**		22
	0.6		99.0%[3]	164		185	**60**		25
	0.7		99.0%[1]	**212**		234	397		185
	0.8		-	169[4]	20.0%[1]	167	**334**		134
	0.9		-	40[3]	20.0%[2]	13	**66**[4]	**20.0**%[1]	27
	0.1	3071		< 1		1	< 1		1
	0.2		99.0%	< 1		1	< 1		1
	0.3		99.0%	< 1		1	< 1		1
	0.4		99.0%	10		1	**1**		2
250	0.5		99.0%	525		133	**61**		16
	0.6		99.0%[3]	438[2]	20.0%[3]	144	**995**		259
	0.7		99.0%[1]	1422		431	**182**		36
	0.8		-	723[3]	20.0%[2]	42	**294**		56
	0.9		-	68[4]	20.0%[1]	12	**395**		83
	0.1		99.4%	**1**		1	**1**		1
	0.2		99.0%	**1**		1	**1**		1
	0.3		99.0%	**1**		1	**1**		1
	0.4		99.0%	391		23	**13**		2
300	0.5	3065[1]	99.0%[4]	1380[4]	20.0%[1]	189	**447**		60
	0.6		99.0%[2]	851[3]	22.5%[2]	188	**1165**[4]	**20.0**%[1]	142
	0.7		99.0%[1]	1408[3]	25.0%[2]	126	**318**[3]	**22.5**%[2]	34
	0.8		-	565[3]	17.5%[2]	57	**78**[4]	**25.0**%[1]	9
	0.9		-	452[4]	25.0%[1]	19	**169**		24

Table 15: Results for strongly-correlated instances - *Sum-Bottleneck* problem.

		Sum-Bottleneck					
		B&B			IB&B		
n	ρ	time	gap	#nodes	time	gap	#nodes
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
100	0.5	< 1		1	< 1		1
	0.6	< 1		1	< 1		1
	0.7	< 1		1	< 1		1
	0.8	367		5719	**174**		3928
	0.9	450		5717	**410**		6464
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
150	0.5	< 1		1	< 1		1
	0.6	< 1		1	< 1		1
	0.7	< 1		1	< 1		1
	0.8	4		32	$< \mathbf{1}$		2
	0.9	6		33	**2**		12
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
200	0.5	< 1		1	< 1		1
	0.6	< 1		2	< 1		2
	0.7	1		2	$< \mathbf{1}$		2
	0.8	2		7	**1**		5
	0.9	2^4	17.3%[1]	4	$\mathbf{1}^4$	**17.3**%[1]	4
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
250	0.5	< 1		1	< 1		1
	0.6	< 1		1	< 1		1
	0.7	139		1752	**60**		182
	0.8	**13**		20	25		50
	0.9	83		120	**60**		81
	0.1	< 1		1	< 1		1
	0.2	< 1		1	< 1		1
	0.3	< 1		1	< 1		1
	0.4	< 1		1	< 1		1
300	0.5	< 1		1	< 1		1
	0.6	< 1		1	< 1		1
	0.7	1		2	$< \mathbf{1}$		2
	0.8	2		2	$< \mathbf{1}$		1
	0.9	31^4	12.1%[1]	27	$\mathbf{21}^4$	**12.1**%[1]	19

4.0 A BILEVEL APPROACH TO THE VACCINE PROCUREMENT POLICY

As defined by the World Health Organization (WHO), vaccine procurement "describes the activities of a country that are necessary to acquire vaccines domestically or internationally, utilizing specific procurement procedures and/or mechanisms" [51]. In particular, the main objective of vaccine procurement "is to ensure that country programs receive products of assured quality at affordable prices in a timely manner in order to optimize immunization program performance and achieve their goals of reducing mortality and morbidity caused by vaccine preventable diseases" [51].

There are numerous studies of the vaccine procurement policy problem (see, e.g., [61, 62]), where a centralized optimization model is proposed to capture both the societal and the manufacturers' goals. This objective is typically achieved by minimizing the vaccine purchasing costs [62] or maximizing social welfare [61], while ensuring some predetermined level of profitability for the manufacturers [61, 62].

In this chapter, we introduce a bilevel programming model to address the vaccine procurement policy in which the government exploits its monopsonistic power (see, e.g., [62] for a more detailed discussion) to influence the decisions of the involved manufacturers. However, our model also recognizes the autonomy of the manufacturers. Specifically, there is no obligation for the manufacturers to satisfy the government's order and moreover, the manufacturers are free to choose their own production plans.

One notable example of bilevel optimization models in the related literature can be found in [52]. The author proposes a bilevel multi-stage stochastic mixed-integer program for designing influenza vaccine, which combines the strain selection by an advisory committee and production decisions of the manufacturers through a two-level decision hierarchy.

In this chapter, we consider a hypothetical vaccine production dynamics with a bilevel decision-making process. At the upper-level, the government's goal is to minimize the maximum shortage of antigens subject to a budgetary constraint. At the lower-level, the manufacturers choose the production plans so as to maximize their individual profits. In our setting, the manufacturers are able to increase their profit margins by producing combination vaccines (see, e.g., [62]). However, since the research and development costs of combination vaccines are significantly higher than traditional vaccines, there should be a sufficiently large demand to make the manufacturers move toward production of combination vaccines.We describe our bilevel model in Section 4.1.

Furthermore, in Section 4.2 we introduce two central planner formulations, specifically, one from the leader's perspective and the other from the follower's perspective. In Section 4.3 we develop a branch-and-bound algorithm along with a threshold-type algorithm to solve the bilevel model. Section 4.4 presents computational examples of our bilevel model to demonstrate some interesting insights with possible practical applications. Lastly, we conclude the discussion in Section 4.5.

4.1 BILEVEL FORMULATION

We formulate the vaccine procurement policy problem as a bilevel mixed-integer program. To simplify the exposition, we consider one manufacturer. Following [61], we define a combination vaccine as an indivisible bundle of antigens. Let $\mathcal{A}$ and $\mathcal{B}$ be the sets of all antigens and all bundles produced by the manufacturer, respectively. Parameters of our model include:

- b: the amount of budget available to the government;
- d_k: the number of doses of antigen $k \in \mathcal{A}$ required by the government;
- p_k: the price per dose of antigen $k \in \mathcal{A}$;
- r_j: the capacity of producing bundle $j \in \mathcal{B}$ by the manufacturer;
- s_j: the setup cost of producing bundle $j \in \mathcal{B}$ by the manufacturer;
- c_j: the cost of producing one unit of bundle $j \in \mathcal{B}$ by the manufacturer;
- a_{jk}: the number of doses of antigen $k \in \mathcal{A}$ in one dose of bundle $j \in \mathcal{B}$.

We introduce a non-negative integer variable x_k, which indicates the order size of antigen $k \in \mathcal{A}$. We also define a non-negative integer variable w_k, $k \in \mathcal{A}$, which determines the number of antigens sold by the manufacturer. Additionally, we introduce a non-negative integer variable y_j, which indicates the number of doses of bundle $j \in \mathcal{B}$ produced by the manufacturer. Finally, we define a binary variable z_j for each bundle $j \in \mathcal{B}$, where $z_j = 1$ implies that the bundle is produced by the manufacturer.

Based on the notation defined above, the government problem can be formulated as:

$$\min_{\mathbf{x}} \quad \psi \tag{4.1a}$$

$$\text{s.t.} \quad \psi \geq d_k - \sum_{j \in \mathcal{B}} a_{jk} y_j \qquad \forall k \in \mathcal{A}, \tag{4.1b}$$

$$\sum_{k \in \mathcal{A}} p_k x_k \leq b, \tag{4.1c}$$

$$x_k \in \mathbb{Z}^1_+ \qquad \forall k \in \mathcal{A}, \tag{4.1d}$$

and the profit maximization problem of the manufacturer can be formulated as:

$$\max_{\mathbf{w},\mathbf{y},\mathbf{z}} \quad \sum_{k \in \mathcal{A}} p_k w_k - \sum_{j \in \mathcal{B}} s_j z_j - \sum_{j \in \mathcal{B}} c_j y_j \tag{4.2a}$$

$$\text{s.t.} \quad w_k \leq x_k \qquad \forall k \in \mathcal{A}, \tag{4.2b}$$

$$w_k \leq \sum_{j \in \mathcal{B}} a_{jk} y_j \qquad \forall k \in \mathcal{A}, \tag{4.2c}$$

$$y_j \leq r_j z_j \qquad \forall j \in \mathcal{B}, \tag{4.2d}$$

$$w_k, y_j \in \mathbb{Z}^1_+, \quad z_j \in \{0,1\} \qquad \forall k \in \mathcal{A}, \forall j \in \mathcal{B}. \tag{4.2e}$$

The government objective (4.1a) along with constraint (4.1b) minimizes the maximum shortage of the antigens. Constraint (4.1c) ensures that the total cost of ordering is less than the available budget. The manufacturer objective (4.2a) maximizes its profit, which is the difference of the total revenue and total costs (i.e., the summation of fixed charges and variable costs). Constraint (4.2b) ensures that number of antigens sold does not ex-

ceed the order quantity specified by the government. Constraint (4.2c) guarantees that the manufacturer does not sell more antigens than it produces. Constraint (4.2d) describes the manufacturer's production capacity for each bundle.

Note that the lower-level optimization problem (4.2) is a classical fixed charge problem [64]. A bilevel extension of the fixed charge problem in a military setting is considered in [2], where the leader is a defender and the follower is an attacker. The authors propose two heuristic methods to solve the resulting bilevel programming model.

Another interesting observation is that our problem becomes "easy" to solve when the government's budget is sufficiently large. More specifically, if $b \geq \sum_{k\in\mathcal{A}} p_k \sum_{j\in\mathcal{B}} a_{jk} r_j$, then setting $x_k = w_k = \sum_{j\in\mathcal{B}} a_{jk} r_j$ for all $k \in \mathcal{A}$, $y_j = r_j$ and $z_j = 1$ for all $j \in \mathcal{B}$ results in an optimal solution to (4.1)-(4.2). This observation is opposite to the results in Chapter 2, where we show that the bilevel knapsack problem remains NP-hard even if the right-hand side of the knapsack constraint is "extremely" large.

We would also like to note that although the antigens' prices are assumed to be fixed, the average antigens' price in the optimal solution can be less than fixed price p_k, $k \in \mathcal{A}$. Specifically, we have:

Remark 1. In an optimal solution of problem (4.1)-(4.2), it is possible that the amount of antigens produced is strictly larger than the amount of antigens sold, i.e., $w_k < \sum_{j\in\mathcal{B}} a_{jk} y_j$. In other words, the manufacturer may produce some doses of a bundle despite the fact that the government does not pay for all antigens in that bundle, i.e., producing a bundle can remain profitable even if the manufacturer is paid only for a subset of antigens of that bundle rather than for all of its antigens.

In the remainder of the chapter we make the following assumptions:

A1: Optimistic case of the bilevel problem is considered.

A2: For all $k \in \mathcal{A}$, $p_k > 0$. Also, for all $j \in \mathcal{B}$, $c_j > 0$ and $s_j > 0$.

A3: Every bundle is economically feasible, i.e., for all $j \in \mathcal{B}$, there exists $y_j \in \mathbb{Z}^1_+$ such that $y_j \leq r_j$ and $\sum_{k\in\mathcal{A}} p_k a_{jk} y_j - s_j - c_j y_j > 0$.

Assumption **A1** is typical in the hierarchical optimization literature and is also present in Chapters 2 and 3. Assumptions **A2** and **A3** are technical. In fact, unless **A3** holds,

the manufacturer does not produce the bundle in his/her optimal production plan. It also implies that the unit price of each bundle is greater than its unit cost, i.e., $\sum_{k\in\mathcal{A}} p_k a_{jk} > c_j$ for all $j \in \mathcal{B}$.

4.2 CENTRAL PLANNER FORMULATION

In this section we formulate the decision problem of a central planner, where the autonomy of the other decision-maker is violated. We consider this problem from both the government's and the manufacturer's perspective. These models provide us with benchmarks for evaluating optimal solutions of the bilevel model. Subsequently, we compare the results of these formulations in Section 4.4.

4.2.1 Leader's perspective

The central planner formulation from the leader's perspective ignores the manufacturer's profitability. However, it assumes that the government pays for each antigen produced.

$$
\begin{aligned}
[\text{CPL}] \quad \min_{\psi,\mathbf{y}} \quad & \psi && && (4.3a)\\
\text{s.t.} \quad & \psi \geq d_k - x_k && \forall k \in \mathcal{A}, && (4.3b)\\
& \sum_{k\in\mathcal{A}} p_k x_k \leq b, && && (4.3c)\\
& x_k \in \mathbb{Z}^1_+ && \forall j \in \mathcal{B}. && (4.3d)
\end{aligned}
$$

Observe that in problem (4.3) the government pays for every antigen produced, which is in contrast with the bilevel model as mentioned in Remark 1. Thus, the optimal objective value of problem (4.3) does not necessarily provide an upper bound for the objective function in (4.1a).

4.2.2 Follower's perspective

To formulate the problem from the follower's perspective, we add the leader's budget constraint (4.1d) to the follower problem (4.2). We have:

$$
\begin{aligned}
[\text{CPF}] \quad \max_{\mathbf{w},\mathbf{y},\mathbf{z}} \quad & \sum_{k\in\mathcal{A}} p_k w_k - \sum_{j\in\mathcal{B}} s_j z_j - \sum_{j\in\mathcal{B}} c_j y_j && && \text{(4.4a)}\\
\text{s.t. } & w_k \le \sum_{j\in\mathcal{B}} a_{jk} y_j && \forall k \in \mathcal{A}, && \text{(4.4b)}\\
& y_j \le r_j z_j && \forall j \in \mathcal{B}, && \text{(4.4c)}\\
& \sum_{k\in\mathcal{A}} p_k w_k \le b, && && \text{(4.4d)}\\
& w_k, y_j \in \mathbb{Z}^1_+, \quad z_j \in \{0,1\} && \forall k \in \mathcal{A}, \forall j \in \mathcal{B}. && \text{(4.4e)}
\end{aligned}
$$

As mentioned earlier regarding the lower-level optimization problem (4.2), problem (4.4) is also a classical fixed charge problem with resource and capacity constraints.

4.3 EXACT SOLUTION APPROACH

We propose a tailored branch-and-bound (B&B) algorithm along with a threshold algorithm to solve problem (4.1)-(4.2). Our B&B algorithm verifies whether there exists a bilevel feasible solution for a given value of the leader's objective. For a given feasible leader's solution $\mathbf{x}$, we define the *follower's rational reaction set* as:

$$
\mathcal{R}(\mathbf{x}) = \underset{\mathbf{w},\mathbf{y},\mathbf{z}}{\operatorname{argmax}} \Big\{ (4.2a) : (4.2b) - (4.2e) \Big\}.
$$

4.3.1 Branch-and-Bound Algorithm

We consider a single-level auxiliary problem obtained by adding (4.1b) and (4.1c) to the follower's problem in (4.2). We also substitute x_k by w_k for all $k \in \mathcal{A}$ and introduce set $\mathcal{W} \subseteq \mathbb{Z}^1_+$ such that $w \in \mathcal{W}$. The resulting problem for given $\bar{\psi} \in \mathbb{Z}^1_+$ and $\mathcal{W} \subseteq \mathbb{Z}^1_+$ is as follows:

$$
\begin{aligned}
[\mathcal{AP}(\bar{\psi}, \mathcal{W})] \quad \max_{\mathbf{w},\mathbf{y},\mathbf{z}} \quad & \sum_{k\in\mathcal{A}} p_k w_k - \sum_{j\in\mathcal{B}} s_j z_j - \sum_{j\in\mathcal{B}} c_j y_j && && \text{(4.5a)}\\
\text{s.t. } & w_k \le \sum_{j\in\mathcal{B}} a_{jk} y_j && \forall k \in \mathcal{A}, && \text{(4.5b)}\\
& y_j \le r_j z_j && \forall j \in \mathcal{B}, && \text{(4.5c)}\\
& d_k - \sum_{j\in\mathcal{B}} a_{jk} y_j \le \bar{\psi} && \forall k \in \mathcal{A}, && \text{(4.5d)}\\
& \sum_{k\in\mathcal{A}} p_k w_k \le b, && && \text{(4.5e)}\\
& \mathbf{w} \in \mathcal{W}, && && \text{(4.5f)}\\
& z_j \in \{0,1\}, \quad y_j \in \mathbb{Z}^1_+ && \forall j \in \mathcal{B}. && \text{(4.5g)}
\end{aligned}
$$

Within our branch-and-bound algorithm each node of the search tree is associated with $\mathcal{AP}(\bar{\psi}, \mathcal{W})$ for a given set $\mathcal{W}$. Specifically, the root node of the B&B tree corresponds to $\mathcal{AP}(\bar{\psi}, \mathcal{W}^0)$, where $\mathcal{W}^0 = \mathbb{Z}^{|k|}_+$. The next result describes how an optimal solution to $\mathcal{AP}(\bar{\psi}, \mathcal{W})$ can be exploited to check whether there exists a feasible solution to the original bilevel problem with the objective function value of $\bar{\psi}$.

Proposition 9. *Let $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution to $\mathcal{AP}(\bar{\psi}, \mathcal{W})$. If $\mathcal{R}(\mathbf{w}^*) \neq \emptyset$, then the the following statements hold:*

1. *If $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*) \in \mathcal{R}(\mathbf{w}^*)$, then $(\mathbf{w}^*, \mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*)$ is a feasible solution to problem (4.1)-(4.2) with objective value $\bar{\psi}$.*
2. *If $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*) \notin \mathcal{R}(\mathbf{w}^*)$, then there exists $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}) \in \mathcal{R}(\mathbf{w}^*)$ and there is no $\mathbf{w} \ge \tilde{\mathbf{w}}$, $\mathbf{w} \in \mathcal{W}$ such that for some $(\mathbf{y}, \mathbf{z})$, $(\mathbf{w}, \mathbf{y}, \mathbf{z}) \in \mathcal{R}(\mathbf{w})$ and $(\mathbf{w}, \mathbf{y}, \mathbf{z})$ is feasible for $\mathcal{AP}(\bar{\psi}, \mathcal{W})$.*
3. *If $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*) \notin \mathcal{R}(\mathbf{w}^*)$, then there exists $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}) \in \mathcal{R}(\mathbf{w}^*)$ such that $\psi > \bar{\psi}$ for any $\mathbf{x} \ge \tilde{\mathbf{w}}$ and $\mathbf{x} \in \mathcal{W}$.*

Proof. First, we prove the correctness of the second statement. Let $\mathcal{F}(\mathbf{x})$ be the optimal objective function value of problem (4.2) for a given leader's decision $\mathbf{x}$. Note that $\mathcal{F}(\mathbf{x})$ is nondecreasing over $\mathbf{x}$. Let $\mathbf{x} = \mathbf{w}^*$ and $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}) \in \mathcal{R}(\mathbf{w}^*)$. First, observe that $\tilde{\mathbf{w}} \leq \mathbf{w}^*$ and $\mathcal{F}(\tilde{\mathbf{w}}) = \mathcal{F}(\mathbf{w}^*)$ due to (4.2b). Next, let g^* be the optimal objective function value of $\mathcal{AP}(\bar{\psi}, \mathcal{W})$ for some given $\bar{\psi}$ and $\mathcal{W}$. Note that g^* constitutes a lower bound on $\mathcal{F}(\mathbf{w})$ for all $\mathbf{w} \in \mathcal{W}$ and since $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*) \notin \mathcal{R}(\mathbf{w}^*)$, this lower bound is strict, i.e., $g^* < \mathcal{F}(\mathbf{w}^*) = \mathcal{F}(\tilde{\mathbf{w}})$. Note that this strict inequality holds for all $\mathbf{w} \geq \tilde{\mathbf{w}}$. Hence, the second statement holds.

The first and the third statements of the proposition follow from the fact that feasible solution $(\mathbf{x}, \mathbf{w}, \mathbf{y}, \mathbf{z})$ to problem (4.1)-(4.2) with $\mathbf{x} \in \mathcal{W}$ and objective value $\bar{\psi}$ is also feasible to $\mathcal{AP}(\bar{\psi}, \mathcal{W})$. □

Given vector $\tilde{\mathbf{w}} = (\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{|\mathcal{A}|})^T$, set $\mathcal{W}$ can be partitioned with respect to $\tilde{\mathbf{w}}$ as the union of mutually disjoint subsets $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_{|\mathcal{A}|}$, where

$$\begin{aligned}
&\mathcal{W}_1 = \mathcal{W} \cap \{w_1 \leq \tilde{w}_1 - 1\};\\
&\mathcal{W}_2 = \mathcal{W} \cap \{w_1 \geq \tilde{w}_1, w_2 \leq \tilde{w}_2 - 1\};\\
&\mathcal{W}_3 = \mathcal{W} \cap \{w_1 \geq \tilde{w}_1, w_2 \geq \tilde{w}_2, w_3 \leq \tilde{w}_3 - 1\};\\
&\quad\vdots\\
&\mathcal{W}_{|\mathcal{A}|-1} = \mathcal{W} \cap \{w_1 \geq \tilde{w}_1, w_2 \geq \tilde{w}_2, \ldots, w_{k-1} \geq \tilde{w}_{\mathcal{A}-1}, w_{\mathcal{A}} \leq \tilde{w}_{\mathcal{A}} - 1\};\\
&\mathcal{W}_{|\mathcal{A}|} = \mathcal{W} \cap \{w_1 \geq \tilde{w}_1, w_2 \geq \tilde{w}_2, \ldots, w_{\mathcal{A}-1} \geq \tilde{w}_{\mathcal{A}-1}, w_k \geq \tilde{w}_{\mathcal{A}}\}.
\end{aligned} \tag{4.6}$$

Then, we have

$$\mathcal{W} = \cup_{i=1}^{|\mathcal{A}|} \mathcal{W}_i. \tag{4.7}$$

Given that $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*)$ is an optimal solution to $\mathcal{AP}(\bar{\psi}, \mathcal{W})$ such that $(\mathbf{w}^*, \mathbf{y}^*, \mathbf{z}^*) \notin \mathcal{R}(\mathbf{w}^*)$, let $(\tilde{\mathbf{w}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}) \in \mathcal{R}(\mathbf{w}^*)$. Then $\mathcal{W}$ can be partitioned with respect to $\tilde{\mathbf{w}}$ by (4.6) and according to Proposition 9, $\mathcal{W}_{|\mathcal{A}|}$ can be immediately pruned.

For any node $\mathcal{N}^\eta$ of the branch-and-bound tree, let $(\mathbf{w}^\eta, \mathbf{y}^\eta, \mathbf{z}^\eta)$ denote an optimal solution of the corresponding problem $\mathcal{AP}(\bar{\psi}, \mathcal{W}^\eta)$, where $\bar{\psi} \in \mathbb{Z}_+^1$ and $\mathcal{W}^\eta \subseteq \mathbb{Z}_+^{|\mathcal{A}|}$. Specifically, the root node $\mathcal{N}^0$ is associated with $\mathcal{W}^0 = \mathbb{Z}_+^{|\mathcal{A}|}$. Let $\mathcal{Q}$ be a set of unprocessed nodes (i.e., nodes that have not been neither pruned nor branched on) of the B&B tree.

Algorithm 1. *Checking whether $\mathcal{AP}(\bar{\psi}, \mathbb{Z}_+^{|\mathcal{A}|})$ is bilevel feasible for a given value of $\bar{\psi}$.*

Step 0: (Initialization) Create a node $\mathcal{N}^0$, let $\mathcal{W}^0 = \mathbb{Z}_+^{|\mathcal{A}|}$ and $\mathcal{AP}(\bar{\psi}, \mathcal{W}^0)$. Initialize list $\mathcal{Q} \leftarrow \{\mathcal{N}^0\}$.

Step 1: (Node selection) If $\mathcal{Q} = \emptyset$, terminate with the result FALSE; otherwise, select and delete node $\mathcal{N}^\eta$ from $\mathcal{Q}$.

Step 2: (Solving) Solve $\mathcal{AP}(\bar{\psi}, \mathcal{W}^\eta)$, for node $\mathcal{N}^\eta$. Denote by $(\mathbf{w}^\eta, \mathbf{y}^\eta, \mathbf{z}^\eta)$ its optimal solution.

Step 3: (Follower Optimality)

(3a) If $(\mathbf{w}^\eta, \mathbf{y}^\eta, \mathbf{z}^\eta) \in \mathcal{R}(\mathbf{w}^\eta)$, then terminate with the result TRUE.

(3b) If $(\mathbf{w}^\eta, \mathbf{y}^\eta, \mathbf{z}^\eta) \notin \mathcal{R}(\mathbf{w}^\eta)$, then find $(\tilde{\mathbf{w}}^\eta, \tilde{\mathbf{y}}^\eta, \tilde{\mathbf{z}}^\eta) \in \mathcal{R}(\mathbf{w}^\eta)$ and go to Step 4.

Step 4: (Branching) From $\mathcal{W}^\eta$ construct $\mathcal{W}_1^\eta, \ldots, \mathcal{W}_{|\mathcal{A}|-1}^\eta$ with respect to $(\tilde{\mathbf{w}}^\eta, \tilde{\mathbf{y}}^\eta, \tilde{\mathbf{z}}^\eta)$ using (4.6). Add the corresponding new nodes $\mathcal{N}_1^\eta, \ldots, \mathcal{N}_{|\mathcal{A}|-1}^\eta$ to $\mathcal{Q}$ and go to Step 1.

In Step 0, the algorithm initializes the root node by $\mathcal{W}^0 = \mathbb{Z}_+^{|\mathcal{A}|}$. In Step 1, node $\mathcal{N}^\eta$ is chosen from $\mathcal{Q}$ according to some node selection rule. Step 2 solves $\mathcal{AP}(\bar{\psi}, \mathcal{W}^\eta)$. Step 3a verifies whether the optimal solution of $\mathcal{AP}(\bar{\psi}, \mathcal{W}^\eta)$ is also optimal for the lower-level problem. Specifically, if it is optimal, then the algorithm stops and returns true. Otherwise, in Step 4 the algorithm branches by creating new nodes of the search tree by using an optimal solution to the follower's problem.

Note that $\mathcal{AP}(\psi, \mathcal{W})$ holds the threshold property on ψ. In other words, if $\mathcal{AP}(\bar{\psi}, \mathcal{W})$ is bilevel feasible for given $\bar{\psi}$, then it is also bilevel feasible for any $\psi \geq \bar{\psi}$. Likewise, if $\mathcal{AP}(\bar{\psi}, \mathcal{W})$ is bilevel infeasible for given $\bar{\psi}$, then it is also bilevel infeasible for any $\psi \leq \bar{\psi}$. Thus, a binary search algorithm [23] can be applied to find the minimum ψ for which $\mathcal{AP}(\psi, \mathcal{W})$ is bilevel feasible.

4.4 COMPUTATIONAL EXPERIMENTS

In this section, we report the computational results illustrating potential practical value of our bilevel model. We compare the leader's and follower's objective function values of

the bilevel programming formulation in (4.1)-(4.2) against its central planner counterparts discussed in Section 4.2, i.e., the leader's and follower's perspective problems given by CPL and CPF, respectively.

The data example for this section consists of a hypothetical vaccine market with one manufacturer. The government's antigen requirements consist of Diphtheria, Tetanus and Pertussis (DTaP), Hepatits B (HBV), and Polio (IPV). Table 16 displays the government's demands and prices for each antigen. Antigen prices are adapted from [61].

Table 17 displays the content, cost and supply features of 7 different bundles. Note that more complex bundles, i.e., bundles including more antigens, have larger fixed costs and higher profit margins. We consider three different manufacturing scenarios in which the manufacturer offers specific set of bundles:

- M3: Bundles 1, 2 and 3;
- M6: Bundles 1, 2, 3, 4, 5 and 6;
- M7: Bundles 1, 2, 3, 4, 5, 6 and 7.

We implement our algorithms in C++ using CPLEX 12.4 callable library. We run the computational experiments on a Windows 7 PC with 3.6GHz CPU and 32GB of RAM.

Table 16: Description of antigens: prices and demands.

Antigens	Price per dose	Demand
	$USD	Million doses
DTaP	24	300
HBV	19	400
IPV	22	200

Table 18 reports the solution times for scenarios M3, M6 and M7 for a range of the government's budget. Not surprisingly, the computational difficulty of solving our test instances increases when the manufacturer has the capability of producing a larger set of bundles. Moreover, the solution time often increases as the available budget increases, because the

Table 17: Description of bundles: costs and supply features.

Bundle	Vaccine	Unit cost	Fixed cost	Supply
		$USD	Million $USD	Million doses
1	DTaP	14	150	400
2	HBV	9	120	400
3	IPV	12	140	400
4	DTaP-HBV	20	200	300
5	DTaP-IPV	22	220	300
6	HBV-IPV	18	190	300
7	DTaP-HBV-IPV	25	300	200

search space expands when the right-hand side of (4.1c) is larger. However, as mentioned in Section 4.1, the problem becomes "easy" for sufficiently large budgets, for example, see the solution times of M3 and M6 when the budget is $20,000$.

Figures 3, 4 and 5 compare the leader's and the follower's objective function values obtained by the bilevel model against those obtained by the central planner models for M3, M6 and M7 scenarios, respectively. We solve these models for different budgets from 1000 to $20,000$ with 1000 increments and compute objective function values (4.1a) and (4.2a) for the obtained optimal solutions. Note that CPL does not consider the production plan. Hence, we change the inequality sign in constraint (4.2b) to equality and solve problem (4.2) to find the follower's objective function value for the given government's order obtained by (4.3). Note that unlike the bilevel model, the manufacturer requires to satisfy the government's order in CPL.

One observation is that the leader's objective function value in the bilevel formulation is closer to that in CPL than CPF. A similar observation is valid for the follower's objective in the bilevel model where its value is closer to the follower's objective function value in CPF. In

other words, the leader can achieve a reasonable shortage by only controlling the order quantities while respecting the manufacturer's autonomy. This observation becomes even more interesting as the follower does not loose a significant portion of profit from its ideal situation.

Another observation is that more complex bundles, i.e., bundles with more antigens, benefit both the government and the manufacturer. Observe that the manufacturer's profit (for the bilevel and the central planner models) increases as the bundle set expands. Similarly, as the bundle set increases, the government shortage often decreases. This is due to the fact that in our test instances the more complex bundles are economically more attractive for the manufacturer.

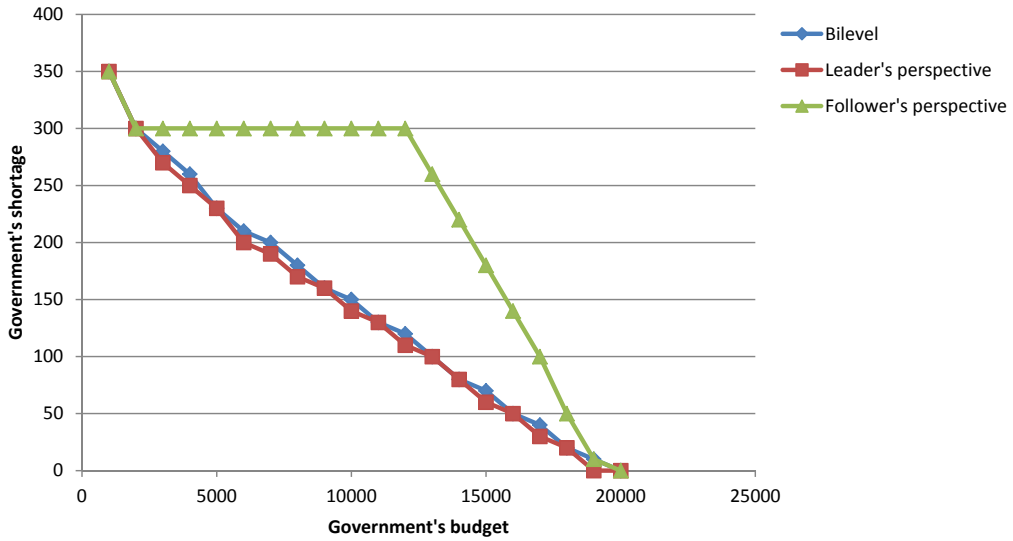
4.5 CONCLUDING REMARKS

This chapter presents a bilevel formulation for a problem in vaccine supply chain. We formulate the problem as a bilevel mixed-integer program and describe an exact solution approach. The performed computational examples demonstrate some interesting insights with possible practical applications of our bilevel model.

An implicit assumption in the proposed bilevel model is that the prices of antigens are fixed. It is interesting to consider extensions, where the leader can also decide on the prices of antigens. However, this generalization can be challenging as the single-level relaxation of the bilevel model becomes nonlinear.

Table 18: Solution time for different government's budgets (in seconds).

Budget	M3	M6	M7
1000	$<$ 1s	$<$ 1s	$<$ 1s
2000	$<$ 1s	$<$ 1s	$<$ 1s
3000	$<$ 1s	2	2
4000	1	8	9
5000	3	18	19
6000	4	22	21
7000	21	151	144
8000	19	138	132
9000	36	263	227
10000	39	291	275
11000	95	799	733
12000	127	959	962
13000	104	1376	863
14000	202	3867	5234
15000	183	4296	4295
16000	115	3492	3271
17000	350	16558	22507
18000	139	8078	7627
19000	142	12191	16263
20000	$<$ 1s	$<$ 1s	6604



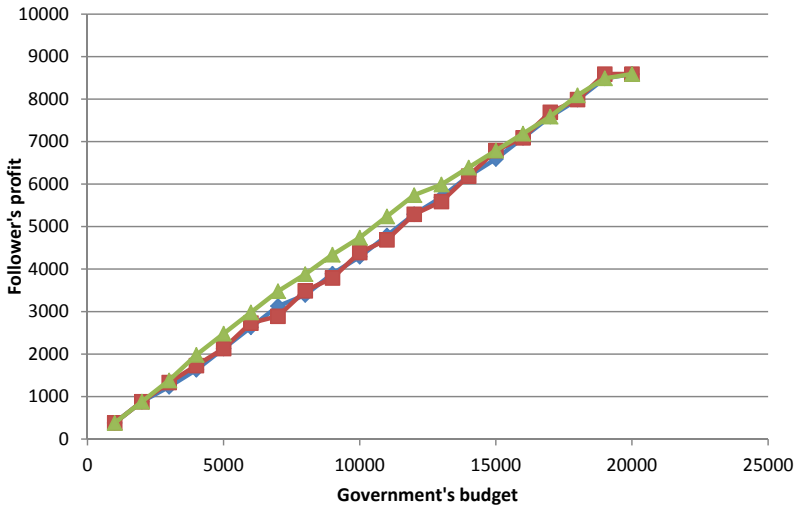


Figure 3: The leader's and the follower's objective function values for different government's budget in Scenario M3.

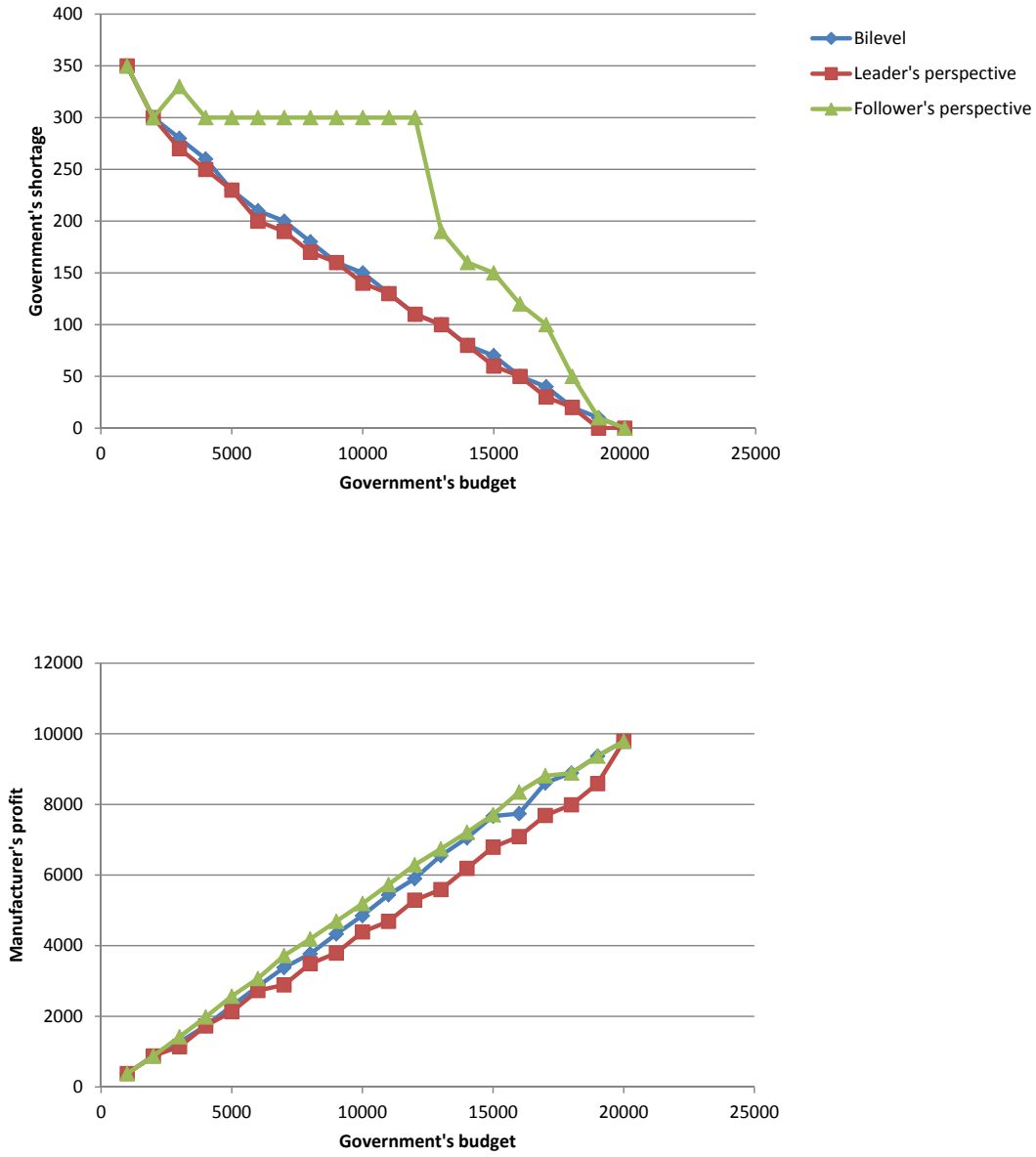


Figure 4: The leader's and the follower's objective function values for different government's budget in Scenario M6.

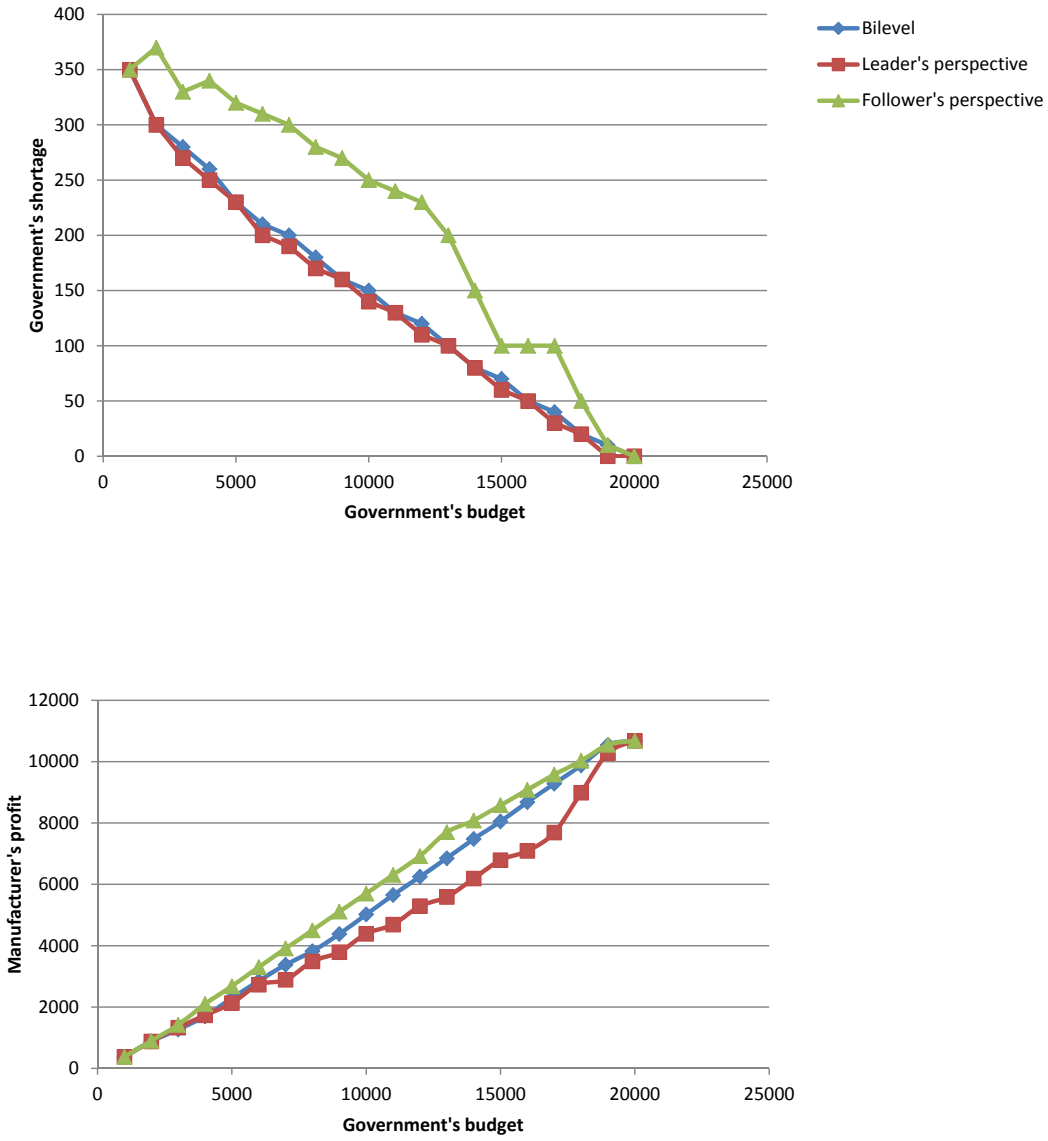


Figure 5: The leader's and the follower's objective function values for different government's budget in Scenario M7.

5.0 CONCLUSIONS

In this dissertation we study three bilevel programs that involve knapsack and assignment constraints in their lower-level problems. We develop novel exact solution approaches that exploit structural properties of the underlying optimization problems. The superiority of the proposed approaches is demonstrated through extensive computational experiments.

Chapter 2 considers a general class of nonlinear bilevel knapsack problems, where the follower's decisions have a nonlinear effect on the leader's objective, and the follower's available capacity is an arbitrary function of the leader's capacity allocation decision. We propose an exact solution approach, which runs efficiently as long as good upper and lower bounds can be generated in the generic branch-and-backtrack algorithm. The performed computational experiments show that our approach can handle large-scale instances efficiently.

Chapter 3 describes exact solution methods for a bilevel extension of the classical assignment problems. Our main approach is based on a branch-and-bound framework and is tailored to exploit structural properties of the assignment constraints and specific classes of the considered objective functions. Our extensive computational experiments demonstrate that our approach is particularly efficient in handling instances with some type of correlation (possibly "weak") between the leader's and the follower's objective functions. While the considered problem is known to be NP-hard in general, we also describe its several restricted subclasses that can be solved in polynomial time.

Chapter 4 presents a bilevel formulation for the vaccine procurement policy problem. We formulate the problem as a bilevel mixed-integer program and describe an exact solution approach to solve the problem. We also present a computational example to illustrate that our model can be especially appropriate to capture the hierarchical structure of the decision-making processes prevalent in practice.

APPENDIX A

THE NBKP TEST SET

The NBKP test set includes 1800 randomly generated instances of the nonlinear bilevel knapsack problem which is available online [8]. The problem formulation and the random instance generation scheme are discussed in Chapter 2.

For each instance there are two data files: the leader's problem file and the follower's problem file. The format of these files is custom and one should have his own reader. Tables 19, 20 and 21 describe the details of this custom format.

Table 19: Format of the leader's problem data file

m	[number of variables]
$\underline{b}$	[lower bound on the leader's budget]
$\bar{b}$	[upper bound on the leader's budget]
$\mathbf{t}_{1\times m}$	[leader's profit]
$\mathbf{w}_{1\times m}$	[leader's weight]

Table 20: Format of the follower's problem data file - quadratic

n	[number of variables]		
o	[*null*]		
$\bar{b} - \underline{b} + 1$	[number of distinct right-hand sides]		
o	[*null*]		
$\mathbf{c}_{n\times 1}$	[follower's profit]	$\mathbf{a}_{n\times 1}$	[follower's weight]
$\mathbf{d}_{1\times n}$	[coefficients of leader's linear objective]		
$Q_{n\times n}$	[leader's quadratic matrix]		
$\mathbf{h}_{1\times(\bar{b}-\underline{b}+1)}$	[values of the right-hand side function]		

Table 21: Format of the follower's problem data file - fractional

n	[number of variables]		
o	[*null*]		
$\bar{b} - \underline{b} + 1$	[number of distinct right-hand sides]		
$\mathbf{c}_{n\times 1}$	[follower's profit]	$\mathbf{a}_{n\times 1}$	[follower's weight]
$\mathbf{p}_{1\times(n+1)}$	[numerator coefficients of the leader's objective]		
$\mathbf{q}_{1\times(n+1)}$	[denominator coefficients of the leader's objective]		
$\mathbf{h}_{1\times(\bar{b}-\underline{b}+1)}$	[values of the right-hand side function]		

APPENDIX B

THE BAP TEST SET

The BAP test set includes 1080 randomly generated instances of the bilevel assignment problem which is available online [7]. The problem formulation and the random instance generation scheme are discussed in Chapter 3.

The format of the file is custom and one should have his own reader. Table 22 describes the details of this custom format.

Table 22: Format of data file

n	[number of variables]
$D = (d_{ij})_{n \times n}$	[leader's cost matrix]
$C = (c_{ij})_{n \times n}$	[follower's cost matrix]

An entry of "-1" in the follower's cost matrix indicates that the corresponding edge doesn't belong to the follower's edge set, i.e., the edge is controlled by the leader.

Test file name

`File Name : BAP_num_coeffMax_corr_pct_id.txt`

`Example: BAP_25_100_1_50_2.txt`

- `num`: number of variables, i.e., n.
- `coeffMax`: maximum value of the leader's and the follower's coefficients.

- `corr`: correlation between the coefficients of the leader and the follower (1: uncorrelated, 2: correlated, 3: highly correlated).
- `pct`: follower's matrix density, i.e., $m_f \approx \texttt{pct} \times n^2$ where m_f is the number of edges that belong to the follower's edge set.
- `id`: test file id.

The leader's problem test file name

`File Name : Leader_num_profitMax_weightMax_numRhs_id.txt`

`Example: Leader_25_1000_100_500_2.txt`

- `num`: number of variables, i.e., m.
- `profitMax`: maximum value of the leader's profit coefficients.
- `weightMax`: maximum value of the leader's weight coefficients.
- `numRhs`: number of distinct right-hand sides, i.e., $\bar{b} - \underline{b} + 1$.
- `id`: test file id.

The follower's problem test file name - quadratic

`File Name : QTEST_num_corr_leaderMax_numRhs_pct_typeRhs_id.txt`

`Example: QTEST_25_3_100_500_100_0_2.txt`

- `num`: number of variables, i.e., n.
- `corr`: correlation between the weights and the follower's profits (1: uncorrelated, 2: correlated, 3: highly correlated).
- `leaderMax`: maximum value of the leader's quadratic coefficients.
- `numRhs`: number of distinct right-hand sides, i.e., $\bar{b} - \underline{b} + 1$.
- `pct`: density of the quadratic matrix.
- `typeRhs`: type of the right-hand side function (0: class C, 1: class S).
- `id`: test file id.

The follower's problem test file name - fractional

`File Name : FTEST_num_corr_leaderMax_numRhs_typeRhs_id.txt`

`Example: FTEST_25_3_100_500_0_2.txt`

- `num`: number of variables, i.e., n.
- `corr`: correlation between the weights and the follower's profits (1: uncorrelated, 2: correlated, 3: highly correlated).
- `leaderMax`: maximum value of the denominator coefficients.
- `numRhs`: number of distinct right-hand sides, i.e., $\bar{b} - \underline{b} + 1$.
- `typeRhs`: type of the right-hand side function (0: class C, 1: class S).
- `id`: test file id.

BIBLIOGRAPHY

[1] R. Aboudi and K. Jørnsten. Resource constrained assignment problems. *Discrete Applied Mathematics*, 26(2):175–191, 1990.

[2] D. Aksen and N. Aras. A bilevel fixed charge location model for facilities under imminent attack. *Computers & Operations Research*, 39(7):1364–1381, 2012.

[3] C. Audet, P. Hansen, B. Jaumard, and G. Savard. Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications*, 93(2):273–300, 1997.

[4] M. L. Balinski and R. E. Gomory. A primal method for the assignment and transportation problems. *Management Science*, 10(3):578–593, 1964.

[5] J. F. Bard, J. Plummer, and J. C. Sourie. A bilevel programming approach to determining tax credits for biofuel production. *European Journal of Operational Research*, 120(1):30–46, 2000.

[6] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. Nonlinear programming: theory and algorithms. 1993. *John Wiley&Sons, New York*.

[7] B. Beheshti. Test instances for bilevel assignment problem. Available at http://www.pitt.edu/~droleg/files/BAP.htm Accessed on June 16, 2014.

[8] B. Beheshti. Test instances for nonlinear bilevel knapsack problem. Available at http://www.pitt.edu/~droleg/files/NBKP.html Accessed on March 31, 2014.

[9] B. Beheshti, O. Y. Özaltın, M. H. Zare, and O. A. Prokopyev. Exact solution approach for a class of nonlinear bilevel knapsack problems. *Journal of Global Optimization*, accepted for publication, 2014.

[10] C. Blair. Sensitivity analysis for knapsack problems: A negative result. *Discrete Applied Mathematics*, 81(1):133–139, 1998.

[11] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1):155–225, 2002.

[12] L. Brotcorn, S. Hanafi, and R. Mansi. One-level reformulation of the bilevel knapsack problem using dynamic programming. *Discrete Optimization*, 10(1):1–1710, 2013.

[13] L. Brotcorne, S. Hanafi, and R. Mansi. A dynamic programming algorithm for the bilevel knapsack problem. *Operations Research Letters*, 37(3):215–218, 2009.

[14] G. Brown, M. Carlyle, J. Salmerón, and K. Wood. Defending critical infrastructure. *Interfaces*, 36(6):530–544, 2006.

[15] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. Society for Industrial Mathematics, 2009.

[16] A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger. A complexity and approximability study of the bilevel knapsack problem. In *Integer Programming and Combinatorial Optimization*, pages 98–109. Springer, 2013.

[17] R. G. Cassidy, M. J. L. Kirby, and W. M. Raike. Efficient distribution of resources through three levels of government. *Management Science*, 17(8):B–462 – B–473, 1971.

[18] J.-P. Côté and G. Savard. A bilevel modeling approach to pricing and fare optimization in the airline industry. *Journal of Revenue and Pricing Management*, 2(1):23–26, 2003.

[19] C. R. Chegireddy and H. W. Hamacher. Algorithms for finding k-best perfect matchings. *Discrete Applied Mathematics*, 18(2):155–165, 1987.

[20] S.-W. Chiou. Bilevel programming for the continuous transport network design problem. *Transportation Research Part B*, 39(4):361–17383, 2005.

[21] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[22] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

[24] S. Dempe. *Foundations of bilevel programming*. Springer, 2002.

[25] S. Dempe and K. Richter. Bilevel programming with knapsack constraints. *Central European Journal of Operations Research*, 8(2):93–107, 2000.

[26] X. Deng. Complexity issues in bilevel linear programming. In A. Migdalas, P. M. Pardalos, and P. Varbrand, editors, *Multilevel Optimization: Algorithms and Applications*, pages 149–164. Kluwer Academic Publishers, 1998.

[27] K. Fukuda and T. Matsui. Finding all minimum-cost perfect matchings in bipartite graphs. *Networks*, 22(5):461–468, 1992.

[28] G. Gallo, P. L. Hammer, and B. Simeone. Quadratic knapsack problems. In M.W. Padberg, editor, *Combinatorial Optimization*, volume 12 of *Mathematical Programming Studies*, pages 132–149. Springer Berlin Heidelberg, 1980.

[29] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[30] R. S. Garfinkel. An improved algorithm for the bottleneck assignment problem. *Operations Research*, 19(7):1747–1751, 1971.

[31] E. Gassner and B. Klinz. The computational complexity of bilevel assignment problems. *4OR: A Quarterly Journal of Operations Research*, 7(4):379–394, 2009.

[32] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[33] F. Gzara. A cutting plane approach for bilevel hazardous material transport network design. *Operations Research Letters*, 41(1):40–1746, 2013.

[34] P. Hansen, M. V. Poggi de Aragão, and C. C. Ribeiro. Hyperbolic 0–1 programming and query optimization in information retrieval. *Mathematical Programming*, 52(1):255–263, 1991.

[35] IBM ILOG. Cplex optimizer user's manual. www.ibm.com/software/commerce/optimization/cplex-optimizer/, 2014.

[36] Gurobi Optimization Inc. Gurobi optimizer reference manual. http://www.gurobi.com, 2014.

[37] S. Keçici, N. Aras, and V. Verter. Incorporating the threat of terrorist attacks in the design of public service facility networks. *Optimization Letters*, 6(6):1101–1121, 2012.

[38] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer Verlag, 2004.

[39] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10(7):1755–1758, 2009.

[40] N. Kong, A. J. Schaefer, and B. Hunsaker. Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach. *Mathematical Programming*, 108(2):275–296, 2006.

[41] P. M. D. Lieshout and A. Volgenant. A branch-and-bound algorithm for the singly constrained assignment problem. *European Journal of Operational Research*, 176(1):151–164, 2007.

[42] Y. H. Liu and T. H. Spencer. Solving a bilevel linear program when the inner decision maker controls few variables. *European Journal of Operational Research*, 81(3):644–651, 1995.

[43] Z.-Q. Luo. *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[44] R. Mansi, C. Alves, J. M. Valério de Carvalho, and S. Hanafi. An exact algorithm for bilevel 0-1 knapsack problems. *Mathematical Problems in Engineering*, 2012. http://www.hindawi.com/journals/mpe/2012/504713/abs/.

[45] R. Mansi, S. Hanafi, and L. Brotcorne. Integer programming formulation of the bilevel knapsack problem. *Mathematical Modelling of Natural Phenomena*, 5(7):116–121, 2010.

[46] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley and Sons, 1990.

[47] A. Migdalas. Bilevel programming in traffic planning: models, methods and challenge. *Journal of Global Optimization*, 7(4):381–405, 1995.

[48] A. Migdalas, P. M. Pardalos, and P. Värbrand. *Multilevel optimization: algorithms and applications*. Kluwer Academic Publishers, 1998.

[49] C. Morefield. Application of 0-1 integer programming to multitarget tracking problems. *Automatic Control, IEEE Transactions on*, 22(3):302–312, 1977.

[50] K. G. Murty. Letter to the editor–an algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16(3):682–687, 1968.

[51] World Health Organization. Immunization, vaccines and biologicals. http://www.who.int/immunization/programmes_systems/procurement/en/. Accessed on: October 28, 2014.

[52] O. Y. Özaltın. *Optimal design of the annual influenza vaccine*. PhD thesis, University of Pittsburgh, 2011.

[53] O. Y. Özaltın, O. A. Prokopyev, and A. J. Schaefer. The bilevel knapsack problem with stochastic right-hand sides. *Operations Research Letters*, 38(4):328–333, 2010.

[54] O. Y. Özaltın, O. A. Prokopyev, and A. J. Schaefer. Two-stage quadratic integer programs with stochastic right-hand sides. *Mathematical Programming*, 133(1–2):121–158, 2012.

[55] P. M. Pardalos and S. Jha. Complexity of uniqueness and local search in quadratic 0–1 programming. *Operations Research Letters*, 11(2):119–123, 1992.

[56] P. M. Pardalos and L. Pitsoulis. *Nonlinear assignment problems: algorithms and applications*, volume 7. Springer, 2000.

[57] P. M. Pardalos and G. Schnitger. Checking local optimality in constrained quadratic programming is NP-hard. *Operations Research Letters*, 7(1):33–35, 1988.

[58] M. Patriksson and R. T. Rockafellar. A mathematical model and descent algorithm for bilevel traffic management. *Transportation Science*, 36(3):271–17291, 2002.

[59] C. R. Pedersen, L. Relund Nielsen, and K. A. Andersen. An algorithm for ranking assignments using reoptimization. *Computers & Operations Research*, 35(11):3714–3726, 2008.

[60] D. Pisinger. The quadratic knapsack problem–a survey. *Discrete Applied Mathematics*, 155(5):623–648, 2007.

[61] R. A. Proano, S. H. Jacobson, and W. Zhang. Making combination vaccines more accessible to low-income countries: The antigen bundle pricing problem. *Omega*, 40(1):53–64, 2012.

[62] M. J. Robbins and S. H. Jacobson. Pediatric vaccine procurement policy: The monopsonist's problem. *Omega*, 39(6):589–597, 2011.

[63] S. Shen, J. C. Smith, and R. Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 9(3):172–17188, 2012.

[64] W. L. Winston and J. B. Goldberg. *Operations Research: Applications And Algorithms*. Thomson Brooks/Cole, 2004.

[65] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.