Annals of Operations Research manuscript No.
(will be inserted by the editor)

On Exact Solution Approaches for Bilevel Quadratic 0–1 Knapsack Problem

Gabriel Lopez Zenarosa ·
Oleg A. Prokopyev ·
Eduardo L. Pasiliao

Received: date / Accepted: date

Abstract We consider the bilevel quadratic knapsack problem (BQKP) that model settings where a leader appropriates a budget for a follower, who solves a quadratic 0–1 knapsack problem. BQKP generalizes the bilevel knapsack problem introduced by Dempe and Richter (2000), where the follower solves a linear 0–1 knapsack problem. We present an exact-solution approach for BQKP based on extensions of dynamic programs for QKP bounds and the branch-and-backtrack algorithm. We compare our approach against a two-phase method computed using an optimization solver in both phases: it first computes the follower's value function for all feasible leader's decisions, and then solves a single-level, value-function reformulation of BQKP as a mixed-integer quadratically constrained program (MIQCP). Our computational experiments show that our approach for solving BQKP outperforms the two-phase method computed by a commercial state-of-the-art optimization software package.

Keywords bilevel programming · bilevel knapsack problem · quadratic knapsack problem · dynamic programming

Mathematics Subject Classification (2010) 90C20 · 90C27 · 90C39 · 90C57

Contract grant sponsor: U.S. Air Force Research Laboratory (AFRL) Mathematical Modeling and Optimization Institute
Contract grant sponsor: U.S. Air Force Office of Scientic Research (AFOSR)
Contract grant sponsor: U.S. Air Force Summer Faculty Fellowship and by AFRL/RW
Contract grant sponsor: National Science Foundation, NSF CMMI 1634835

G.L. Zenarosa
Systems Engineering and Engineering Management, UNC Charlotte, 9201 University City Blvd., Charlotte, NC 28223, USA
ORCID: 0000-0003-1096-3559

O.A. Prokopyev ✉
Department of Industrial Engineering, University of Pittsburgh, 3700 O'Hara Street, Pittsburgh, PA 15261, USA
Tel.:+1-412-624-9833
E-mail: droleg@pitt.edu
ORCID: 0000-0003-2888-8630

E.L. Pasiliao
AFRL Munitions Directorate, Building 13, Eglin AFB, FL 32542, USA
ORCID: 0000-0002-6403-6649

1 Introduction

Bilevel programming is a two-level optimization framework—introduced by Bracken and McGill (1973, 1974, 1978)—for modeling sequential decision-making by two independent (collaborative or adversarial) participants: the *leader* and the *follower*. The leader initially solves the first-level optimization problem, and the follower subsequently solves the second-level optimization problem, where both problems involve the leader's and follower's decision variables (i.e., in the objective functions and constraints). Thus, solving bilevel programs entails the leader making a (feasible) decision to induce a rational reactive decision from the follower (i.e., solving the second-level problem given the leader's decision) that results in an optimal first-level objective function value. An *optimistic* or *pessimistic* bilevel program is modeled whenever the follower is assumed to choose a reaction that is collaborative or adversarial, respectively, with respect to the leader's objective function.

Bilevel programming has applications in a number of diverse areas, including network design, revenue management, and economics, among others. Solving even the simplest of bilevel programs, where both first- and second-level programs are linear, is strongly *NP*-hard (Ben-Ayed and Blair 1990; Jeroslow 1985; Hansen et al. 1992). Numerical approaches for solving bilevel programs, include extreme-point, branch-and-bound, descent, penalty-function, and trust-region methods (cf. Candler and Townsley 1982; Bard and Falk 1982; Vicente et al. 1994; Aiyoshi and Shimizu 1981; Colson et al. 2005, respectively). Some classes of bilevel programs—in particular, those that do not involve integrality restrictions at the lower level—admit single-level linear mixed-integer programming reformulations, which can be solved by optimization software packages, such as CPLEX (IBM Corp. 2017); see, examples in Audet et al. (1997); Zare et al. (2017); Stozhkov et al. (2017). We refer the reader to the survey by Colson et al. (2007) for a more-thorough discussion.

Various researchers considered bilevel programs with special structures to develop exact-solution approaches, which they demonstrated to perform sufficiently well in practice. In particular, Brotcorne et al. (2009, 2013) developed dynamic programs (DPs) to solve the bilevel knapsack problem (BKP; Dempe and Richter 2000). Beheshti et al. (2015) developed a branch-and-backtrack approach (Özaltın et al. 2010) to solve BKP problems, where the leader's objective function is nonlinear (e.g., quadratic and fractional), while the follower's problem is a linear 0–1 knapsack problem as in the aforementioned studies of BKP. Beheshti et al. (2016) developed a combinatorial branch-and-bound algorithm to solve bilevel assignment problems.

In this note, we consider the bilevel quadratic knapsack problem (BQKP), where the second-level problem is a quadratic 0–1 knapsack problem (QKP). Thus, BQKP is a generalization of the BKP as in the latter the follower solves a linear 0–1 knapsack problem (cf. Dempe and Richter 2000). We note that since BKP is *NP*-hard (Brotcorne et al. 2009), then BQKP, as its generalization, is also *NP*-hard.

QKPs have a variety of applications in transportation networks (e.g., Rhys 1970) and telecommunications (e.g., Witzgall 1975), among others (cf. Kellerer et al. 2004; Pisinger 2007). Thus, an example BQKP is finding the optimal budget to enable the construction of a subset of potential facilities: a principal (i.e., leader) appropriates a budget to an agent (i.e., follower) to construct a network of facilities (e.g., airports). It is assumed that the agent obtains revenues from each facility and the paired connections between them (the latter assumption is responsible for appearance of the nonlinear terms in the follower's objective), while the principal obtains a return on investment from each facility.

We use results from the bilevel optimization, BKP and QKP literature to provide an exact-solution approach to BQKP. Specifically, we develop an exact-solution approach for

BQKP based on DP and the branch-and-backtrack algorithm. We compare our approach against a two-phase method computed using a commercial state-of-the-art solver in both phases: it first computes the follower's value function for all feasible leader's decisions, and then solves a single-level, value-function reformulation of BQKP as a mixed-integer quadratically constrained program (MIQCP). Our computational experiments show that our approach for solving BQKP outperforms the two-phase method computed by the commercial state-of-the-art solver.

The remainder of this note is organized as follows. Section 2 presents the BQKP formulation and its single-level, value-function reformulation. Section 3 describes our exact-solution approach to BQKP. Section 4 shows the results of our computational experiments. Finally, Section 5 concludes the note.

2 Mathematical Model and Value-function Reformulation

We consider a set of $n \in \mathbb{Z}^1_+$ items where each item $i \in \{1,\dots,n\}$ has an associated weight $a_i \in \mathbb{Z}^1_+$ and two revenues: the leader's revenue, $d_i \in \mathbb{R}^1_+$, and the follower's revenues: $q_{ii} \in \mathbb{R}^1_+$ and $q_{ij} \in \mathbb{R}^1_+$, where the latter is the joint revenue with item $j \in \{1,\dots,n\}, j \neq i$. The follower must solve a QKP to maximize its own objective subject to the capacity set by the leader, $y \in \mathbb{R}^1_+$. This yields the following BQKP:

$$
\begin{aligned}
(\mathrm{BQKP}) \qquad & \max\ ty + \mathbf{d}^\top \mathbf{x}(y) \\
\text{s.t.}\quad & \underline{y} \le y \le \overline{y} \\
& \mathbf{x}(y) \in \mathscr{R}(y),
\end{aligned}
$$

where $t \in \mathbb{R}^1$ and $\mathbf{d} \in \mathbb{R}^n$ are the leader's objective function coefficients, $\underline{y} \in \mathbb{R}^1_+$ and $\overline{y} \in \mathbb{R}^1_+$ are lower and upper bounds on the leader's decision variable $y \in \mathbb{R}^1_+$, respectively, and,

$$
\mathscr{R}(y) = \operatorname*{arg\,max}_{\mathbf{x}\in\{0,1\}^n} \{\mathbf{x}^\top Q \mathbf{x} : \mathbf{a}^\top \mathbf{x} \le y\}
$$

is the follower's *rational reaction set*.

For 0–1 variables, we have $x_i^2 = x_i$. Thus, QKP reduces to the 0–1 knapsack problem whenever Q is a diagonal matrix. Consequently, in this case, BQKP also reduces to BKP from Dempe and Richter (2000).

In the remainder of this note, we make the following assumptions:

(A1) Optimistic case;
(A2) Nonnegative integer parameters:

$$
\underline{y} \in \mathbb{Z}^1_+,\ \overline{y} \in \mathbb{Z}^1_+,\ \mathbf{d} \in \mathbb{Z}^n_+,\ Q \in \mathbb{Z}^{n\times n}_+ \text{ and } \mathbf{a} \in \mathbb{Z}^n_+;
$$

(A3) Symmetric follower's revenue matrix: $q_{ij} = q_{ji}$, and;
(A4) Feasible and non-trivial knapsack capacities:

$$
\max\nolimits_{i\in\{1,\dots,n\}} a_i < \underline{y} \ \text{ and } \ \overline{y} < \sum_{i=1}^{n} a_i.
$$

Considering only the optimistic case in Assumption **(A1)** is common in the hierarchical optimization literature (e.g., Audet et al. 1997; Migdalas et al. 1998). Nonnegativity and integrality restrictions in Assumption **(A2)** are typical in the literature for knapsack problems and BKPs (e.g., Brotcorne et al. 2009; Martello and Toth 1990). Using symmetric revenue matrices in Assumption **(A3)** along with feasible and non-trivial knapsack capacities in Assumption **(A4)** can be assumed without loss of generality (cf. Pisinger 2007).

For any $\beta \in \mathbb{R}^1_+$, we define the leader's value function for the follower's decisions as:

$$\lambda^n(\beta) \equiv \max_{\mathbf{x}\in\{0,1\}^n} \left\{ \sum_{i=1}^n d_i x_i : \sum_{i=1}^n \left(q_{ii}x_i + 2\sum_{j=i+1}^n q_{ij}x_ix_j \right) \geq \phi^n(\beta), \sum_{i=1}^n a_i x_i \leq \beta \right\},$$

where $\phi^k(\beta)$, for any $k \in \{1,\dots,n\}$, is the follower's value function for the QKP considering only the first k items defined as:

$$\phi^k(\beta) \equiv \max_{\mathbf{x}\in\{0,1\}^k} \left\{ \sum_{i=1}^k \left(q_{ii}x_i + 2\sum_{j=i+1}^k q_{ij}x_ix_j \right) : \sum_{i=1}^k a_i x_i \leq \beta \right\}.$$

The following proposition follows immediately from Theorem 3 of Dempe and Richter (2000) for BKP since $\phi^n(\beta)$ is piecewise constant, right-continuous, and monotonically nondecreasing in $\beta \in [\underline{y}, \overline{y}]$.

Proposition 1 *If $t \leq 0$, then BQKP has a solution. If $t > 0$, then either the optimal solution to BQKP is $(\overline{y}, \overline{\mathbf{x}})$ for some $\overline{\mathbf{x}} \in \mathscr{R}(\overline{y})$ or BQKP has no solution.*

Thus, we make the following additional assumption:

(A5) The coefficient to the leader's decision variable is nonpositive: $t \leq 0$.

Consequently, the following proposition follows immediately from Proposition 3 of Brotcorne et al. (2009).

Proposition 2 *If $t \leq 0$, then BQKP have integral solutions $(y^*, x^*) \in \mathbb{Z}_+ \times \{0,1\}^n$.*

Henceforth, without loss of generality, we consider only integral knapsack capacities $\beta \in \mathbb{Z}_+$, and hence DP algorithms can be used.

Value functions have been reported in the literature to afford the reformulation of BKPs and special-structured two-stage stochastic programs that enabled the development of efficient solution approaches. Brotcorne et al. (2013) used value functions to reformulate BKP as single-level mixed-integer program solvable by commercial software. Beheshti et al. (2015) used a value-function reformulation of BKPs with quadratic and fractional leader objective functions to develop their exact-solution approach. Özaltın et al. (2010) used value functions to reformulate and solve BKP as a two-stage stochastic program. Kong et al. (2006); Trapp et al. (2013); Trapp and Prokopyev (2015) and Özaltın et al. (2012) used value-function reformulations to solve two-stage linear and quadratic integer programs with stochastic right-hand sides, respectively, using superadditive-dual approaches.

BQKP admits a single-level, value-function based reformulation given by:

$$\begin{aligned} \text{(VF–BQKP)} \qquad & \max\ ty + \mathbf{d}^\top \mathbf{x} \\ & \text{s.t. } \underline{y} \leq y \leq \overline{y} \\ & \quad \mathbf{x}^\top Q \mathbf{x} \geq \phi^n(y) \\ & \quad \mathbf{a}^\top \mathbf{x} \leq y \\ & \quad \mathbf{x} \in \{0,1\}^n \\ & \quad y \in \mathbb{Z}_+, \end{aligned}$$

and, under Assumption **(A1)**, the following proposition follows immediately.

Proposition 3 *A solution $(y^*, x^*) \in \mathbb{Z}_+ \times \{0,1\}^n$ is optimal to VF–BQKP if and only if it is optimal to BQKP.*

Similar to Brotcorne et al. (2013), VF–BQKP can be solved by standard software packages in the following two-phase method. Specifically, in the first phase we precompute (i.e., independently) all the value functions $\phi^n(\beta)$, for all $\beta \in \{\underline{y}, \ldots, \overline{y}\}$, which are 0–1 quadratic programs (QPs) solvable by commercial state-of-the-art software packages after some simple manipulations (cf. Billionnet and Elloumi 2007). In the second phase, using these QPs as right-hand sides (RHSs), VF–BQKP can be encoded by using auxiliary indicator variables (cf. Brotcorne et al. 2013), as an MIQCP, also computable by commercial solvers:

$$\mathbf{x}^\top Q\mathbf{x} \geq \phi^n(y) \qquad \Leftrightarrow \qquad \begin{cases} \mathbf{x}^\top Q\mathbf{x} \geq \sum_{\beta=\underline{y}}^{\overline{y}} \phi^n(\beta) \cdot z_\beta \\ \qquad y = \sum_{\beta=\underline{y}}^{\overline{y}} \beta \cdot z_\beta \\ \sum_{\beta=\underline{y}}^{\overline{y}} z_\beta = 1 \\ \quad z_\beta \in \{0,1\}, \qquad \forall \beta \in \{\underline{y}, \ldots, \overline{y}\} \end{cases}$$

In the next section, we present an alternative exact-solution approach, which is the main contribution of this note. Our approach extends the branch-and-backtrack algorithm of Özaltın et al. (2010). Specifically, it prunes the branch-and-bound tree using the solutions to the QKP lower- and upper-bounding value functions, which can be computed using DP. The increased computational burden for solving BQKP over BKP results from having only the bounds to the QKPs—that is, not having their exact solutions.

3 Algorithms

Our exact-solution approach for solving BQKPs is comprised of three computing tasks, which we discuss in the following subsections. Subsection 3.1 describes the computation of the QKP lower bounds $\underline{\phi}^k(\beta)$ in Algorithm 1, where $\underline{\phi}^k(\beta) \leq \phi^k(\beta)$, for all $k \in \{1, \ldots, n\}$ and $\beta \in \{0, \ldots, \overline{y}\}$. Subsection 3.2 describes the computation of the QKP upper bounds $\overline{\phi}^k(\beta)$ in Algorithms 2, 3 and 4, where $\overline{\phi}^k(\beta) \geq \phi^k(\beta)$, for all $k \in \{1, \ldots, n\}$ and $\beta \in \{0, \ldots, \overline{y}\}$. Subsection 3.3 establishes the theoretical results afforded by the QKP lower and upper bounds and, subsequently, describes the procedure using those results for solving BQKP in Algorithm 5. Algorithms 1–3 are DP-based approaches and Algorithm 5 is a variant of the branch-and-backtrack algorithm of Özaltın et al. (2010).

For ease of exposition, we repeat the explicit DP pseudo-code in Algorithms 1–3; in our implementation, however, we define a common DP procedure reusable by those algorithms. In each of the DPs, γ_0 and γ_1 represent the profits for leaving and taking the k^{th} item, respectively (i.e., line 6 in Algorithm 1 and line 8 in Algorithms 2–3).

3.1 Computing the QKP Lower Bounds

Fomeni and Letchford (2014) reported a DP heuristic for computing a lower bound to QKPs based on the classical DP approach for solving the linear 0–1 knapsack problem. They use a greedy approach that considers the QKP with the first $k \in \{1,\ldots,n-1\}$ items and budgets $\beta \in \{0,\ldots,\bar{y}\}$ to inform solving the QKP with the first $k+1$ items via DP. In their first algorithm (cf. Fomeni and Letchford 2014, Algorithm 2), they compute an item-index set of the greedily selected items among the first $k \in \{1,\ldots,n\}$ items, for all $\beta \in \{0,\ldots,\bar{y}\}$.

Our extension of their first algorithm is given by Algorithm 1, where the key difference is that we use the leader objective function coefficients $\mathbf{d} \in \mathbb{Z}_+^n$ to break item-selection ties in the optimistic sense (cf. Assumption **A1**) in lines 13–18. We note that this tie-breaking step is afforded by lines 9, 11 and 13, where we store in $\delta^k(\beta)$ the leader's k-item objective function value corresponding to the follower's item-selection decisions, for all $\beta \in \{0,\ldots,\bar{y}\}$. Tie-breaking steps based on the leader's objective function are commonly used in algorithms for bilevel problems (e.g., Brotcorne et al. 2009; Özaltın et al. 2010; Beheshti et al. 2015).

The item-index set $\underline{\chi}^k(\beta)$ contains the greedily selected items among the first $k \in \{1,\ldots,n\}$ items, for all capacities $\beta \in \{0,\ldots,\bar{y}\}$. We note that $\underline{\chi}^k(\beta)$ induces a feasible (lower-bounding) solution:

$$\underline{\mathbf{x}} = [\underline{x}_i \in \{0,1\} : \underline{x}_i = 1 \Leftrightarrow i \in \underline{\chi}^k(\beta)] \in \{0,1\}^k$$

to the k-item, β-capacitated QKP corresponding to $\phi^k(\beta)$.

Algorithm 1 DP to compute the lower-bounding value functions $\underline{\phi}^\bullet(\cdot)$ and item sets $\underline{\chi}^\bullet(\cdot)$

```
 1: procedure LB($\mathbf{d}$, $Q$, $\mathbf{a}$)
 2:     Initialize, for every $\beta \in \{0,\ldots,\bar{y}\}$,
        $\left(\underline{\phi}^1(\beta), \delta^1(\beta), \underline{\chi}^1(\beta)\right) \leftarrow \begin{cases} (0,0,\emptyset), & \text{if } \beta \in \{0,\ldots,a_1-1\} \\ (q_{11}, d_1, \{1\}), & \text{otherwise} \end{cases}$
 3:     for $k \leftarrow 2,\ldots,n$ do
 4:         for $\beta \leftarrow 0,\ldots,\bar{y}$ do
 5:             if $\beta \geq a_k$ then
 6:                 Let $\gamma_0 \leftarrow \underline{\phi}^{k-1}(\beta)$ and $\gamma_1 \leftarrow \underline{\phi}^{k-1}(\beta - a_k) + q_{kk} + 2\sum_{i \in \underline{\chi}^{k-1}(\beta - a_k)} q_{ik}$
 7:                 Update $\underline{\phi}^k(\beta) \leftarrow \max\ \{\gamma_0, \gamma_1\}$
 8:                 if $\gamma_0 > \gamma_1$ then
 9:                     Update $\delta^k(\beta) \leftarrow \delta^{k-1}(\beta)$ and $\underline{\chi}^k(\beta) \leftarrow \underline{\chi}^{k-1}(\beta)$
10:                 else if $\gamma_0 < \gamma_1$ then
11:                     Update $\delta^k(\beta) \leftarrow \delta^{k-1}(\beta - a_k) + d_k$ and $\underline{\chi}^k(\beta) \leftarrow \underline{\chi}^{k-1}(\beta - a_k) \cup \{k\}$
12:                 else
13:                     Update $\delta^k(\beta) \leftarrow \max\ \{\delta^{k-1}(\beta), \delta^{k-1}(\beta - a_k) + d_k\}$    // Optimistic case
14:                     if $\delta^k(\beta) = \delta^{k-1}(\beta)$ then
15:                         Update $\underline{\chi}^k(\beta) \leftarrow \underline{\chi}^{k-1}(\beta)$
16:                     else
17:                         Update $\underline{\chi}^k(\beta) \leftarrow \underline{\chi}^{k-1}(\beta - a_k) \cup \{k\}$
18:                     end if
19:                 end if
20:             else
21:                 Update $\underline{\phi}^k(\beta) \leftarrow \underline{\phi}^{k-1}(\beta)$, $\delta^k(\beta) \leftarrow \delta^{k-1}(\beta)$, and $\underline{\chi}^k(\beta) \leftarrow \underline{\chi}^{k-1}(\beta)$
22:             end if
23:         end for
24:     end for
25:     return $(\underline{\phi}^\bullet(\cdot), \underline{\chi}^\bullet(\cdot))$
26: end procedure
```

3.2 Computing the QKP Upper Bounds

Let $Q^{[k]}$ denote the kth-order leading principal submatrix of Q. We note the following equalities on the objective function for the k-item, β-capacitated QKP corresponding to $\phi^k(\beta)$, for any $k \in \{1,\dots,n\}$ and $\beta \in \{0,\dots,\bar{y}\}$:

$$\mathbf{x}^\top Q^{[k]}\mathbf{x} = \sum_{i=1}^{k}\sum_{j=1}^{k} q_{ij}x_ix_j = \sum_{i=1}^{k}\left(q_{ii} + \sum_{\substack{j=1\\ j\neq i}}^{k} q_{ij}x_j \right) x_i.$$

We also note that $q_{ii} + \sum_{j=1,j\neq i}^{k} q_{ij}x_j$ is bounded above by the k-item, β-capacitated QKP *upper plane:*

$$\pi_i^k(\beta) \equiv q_{ii} + \max\left\{ \sum_{\substack{j=1\\ j\neq i}}^{k} q_{ij}x_j : \sum_{\substack{j=1\\ j\neq i}}^{k} a_jx_j \leq \beta - a_i; x_j \in \{0,1\}, \forall j \in \{1,\dots,k\}\setminus\{i\} \right\},$$

for all $i \in \{1,\dots,k\}$. Consequently, for all $k \in \{1,\dots,n\}$ and $\beta \in \{0,\dots,\bar{y}\}$, upper bounds to $\phi^k(\beta)$ are obtained by:

$$\overline{\phi}^k(\beta) \equiv \max_{\mathbf{x}\in\{0,1\}^k}\left\{ \sum_{i=1}^{k} \pi_i^k(\beta)\cdot x_i : \sum_{i=1}^{k} a_ix_i \leq \beta \right\},$$

since each item $i \in \{1,\dots,k\}$ selected in solving $\overline{\phi}^k(\beta)$ assumes the selection of additional items in $\{1,\dots,n\}\setminus\{i\}$ corresponding to $\pi_i^k(\beta)$, the union of which may be infeasible to $\overline{\phi}^k(\beta)$. We observe that $\overline{\phi}^k(\beta)$, for all $k \in \{1,\dots,n\}$, and $\beta \in \{0,\dots,\bar{y}\}$ are knapsack problems, which can be solved using classical DP.

Upper planes were first introduced by Gallo et al. (1980) and later refined by Caprara et al. (1999) that afforded effective approaches for computing upper bounds to QKP with n items with fixed capacity $c \in \mathbb{Z}_+$. A comprehensive overview of these and other bounds are surveyed by Pisinger (2007). Using our notation, the upper planes of the fourth kind (i.e., using 0–1 variables and knapsack constraints) of Caprara et al. (1999) is expressed as $\pi_i^n(c)$, for all $i \in \{1,\dots,n\}$.

The quality of the upper bounds induced by upper planes, however, highly depends on the order of the items. Rodrigues et al. (2012) described a procedure for enumerating a subset of item-index permutations to generate valid inequalities using upper planes[1] (i.e., induced by the item-order) to approximate the convex hull of the feasible region to a reformulation of QKP (i.e., linearization using a single variable and an exponential number of constraints). We refer to this procedure by Rodrigues et al. (2012) as *cutting upper planes* for brevity. Additionally, Lagrangian-based approaches were reported to strengthen QKP upper bounds, including those of Caprara et al. (1999) and Billionnet et al. (1999), both of which are discussed by Pisinger et al. (2007).

Straightforward computations of upper planes $\pi_i^k(\beta)$ and upper bounds $\overline{\phi}^k(\beta)$, for all $i \in \{1,\dots,k\}$, $k \in \{1,\dots,n\}$, and $\beta \in \{0,\dots,\bar{y}\}$ are afforded by DP Algorithms 2 then 3, respectively, which are explained below. Furthermore, cutting upper planes for strengthening the QKP upper bounds can be computed by iterating through Algorithms 2 and 3 for

[1] Rodrigues et al. (2012) presented valid inequalities using three types of upper planes: a variant of the first kind of upper planes of Gallo et al. (1980) and those of the third and fourth kind of Caprara et al. (1999).

different item-orders as induced by the last solution returned by Algorithm 3 (cf. Rodrigues et al. 2012, Algorithm 1), which we express as Algorithm 4.

Recall, however, that QKP upper bounds are used for solving BQKP. Thus, there exists a tradeoff in computation times between strengthening QKP upper bounds and solving BQKP itself. We investigate the computational tradeoff in solving the QKP upper bounds using straightforward upper-planes approach versus the cutting upper-planes approach in Section 4. We defer Lagrangian-based enhancements (e.g., Billionnet and Soutif 2004; Chen et al. 2018) for future studies.

Algorithm 2 DP to compute the upper planes $\pi^\bullet(\cdot)$ and item sets $\xi^\bullet(\cdot)$

```
1:  procedure UPPERPLANES(d, Q, a)
2:      for $i \leftarrow 1,\dots,n$ do
3:          Let $\hat{d} \leftarrow [d_1,\dots,d_{i-1},d_{i+1},\dots,d_n]^\top$, $\hat{q} \leftarrow [q_{i,1},\dots,q_{i,i-1},q_{i,i+1},\dots,q_{i,n}]^\top$, and
              $\hat{a} \leftarrow [a_1,\dots,a_{i-1},a_{i+1},\dots,a_n]^\top$
4:          Initialize, for every $\beta \in \{0,\dots,\bar{y}-a_i\}$,
              $\left(\hat{\phi}^1(\beta),\hat{\delta}^1(\beta),\hat{\chi}^1(\beta)\right) \leftarrow \begin{cases}(0,0,\emptyset), & \text{if } \beta \in \{0,\dots,\hat{a}_1-1\}\\ (\hat{q}_1,\hat{d}_1,\{1\}), & \text{if } i>1 \text{ and } \beta \ge \hat{a}_1\\ (\hat{q}_1,\hat{d}_1,\{2\}), & \text{otherwise}\end{cases}$
5:          for $j \leftarrow 2,\dots,n-1$ do
6:              for $\beta \leftarrow 0,\dots,\bar{y}-a_i$ do
7:                  if $\beta \ge \hat{a}_j$ then
8:                      Let $\gamma_0 \leftarrow \hat{\phi}^{j-1}(\beta)$ and $\gamma_1 \leftarrow \hat{\phi}^{j-1}(\beta-\hat{a}_j)+\hat{q}_j$
9:                      Update $\hat{\phi}^j(\beta) \leftarrow \max\ \{\gamma_0,\gamma_1\}$
10:                     Let $\hat{j} \leftarrow \begin{cases} j, & \text{if } j<i\\ j+1, & \text{otherwise}\end{cases}$
11:                     if $\gamma_0 > \gamma_1$ then
12:                         Update $\hat{\delta}^j(\beta) \leftarrow \hat{\delta}^{j-1}(\beta)$ and $\hat{\chi}^j(\beta) \leftarrow \hat{\chi}^{j-1}(\beta)$
13:                     else if $\gamma_0 < \gamma_1$ then
14:                         Update $\hat{\delta}^j(\beta) \leftarrow \hat{\delta}^{j-1}(\beta-\hat{a}_j)+\hat{d}_j$ and $\hat{\chi}^j(\beta) \leftarrow \hat{\chi}^{j-1}(\beta-\hat{a}_j)\cup\{\hat{j}\}$
15:                     else
16:                         Update $\hat{\delta}^j(\beta) \leftarrow \max\ \{\hat{\delta}^{j-1}(\beta),\hat{\delta}^{j-1}(\beta-\hat{a}_j)+\hat{d}_j\}$    // Optimistic case
17:                         if $\hat{\delta}^j(\beta) = \hat{\delta}^{j-1}(\beta)$ then
18:                             Update $\hat{\chi}^j(\beta) \leftarrow \hat{\chi}^{j-1}(\beta)$
19:                         else
20:                             Update $\hat{\chi}^j(\beta) \leftarrow \hat{\chi}^{j-1}(\beta-\hat{a}_j)\cup\{\hat{j}\}$
21:                         end if
22:                     end if
23:                 else
24:                     Update $\hat{\phi}^j(\beta) \leftarrow \hat{\phi}^{j-1}(\beta)$
25:                 end if
26:             end for
27:         end for
28:         Initialize, for every $k \leftarrow i,\dots,n$ and $\beta \leftarrow 0,\dots,\bar{y}$,
              $\left(\pi_i^k(\beta),\xi_i^k(\beta)\right) \leftarrow \begin{cases}(q_{11},\{1\}), & \text{if } k=1\\ (q_{ii},\{i\}), & \text{if } k>1 \text{ and } \beta \in \{0,\dots,a_i\}\\ (q_{ii}+\hat{\phi}^{k-1}(\beta-a_i),\{i\}\cup\hat{\chi}^{k-1}(\beta-a_i)), & \text{otherwise}\end{cases}$
29:     end for
30:     return $(\pi^\bullet(\cdot),\xi^\bullet(\cdot))$
31: end procedure
```

Algorithm 2 extends the classical DP for computing the upper planes of Caprara et al. (1999) to compute all upper planes $\pi_i^k(\beta)$, for all QKPs with the first $k \in \{1,\dots,n\}$ items, capacities $\beta \in \{0,\dots,\bar{y}\}$, and $i \in \{1,\dots,k\}$. Additionally, Algorithm 2 computes item-index sets $\xi_i^k(\beta)$ that correspond to the solutions to $\pi_i^k(\beta)$. We observe that lines 4–27 excluding line 10 describe a DP for an $(n-1)$-item knapsack problem that excludes the i^{th} item, for each $i \in \{1,\dots,n\}$ as iterated in line 2. Lines 13–18 breaks solution ties using the leader's objective function coefficients similar to Algorithm 1. Line 3 prepares the parameters to a knapsack DP without item i, and line 10 adjusts the item index to include it in $\xi_i^k(\beta)$ in lines 14 and 20. Line 28 stores the upper planes and corresponding item-index set computed in lines 4–27.

Algorithm 3 DP to compute the upper-bounding value functions $\overline{\phi}^\bullet(\cdot)$ and item sets $\overline{\chi}^\bullet(\cdot)$

```
1: procedure UB($\mathbf{d}$, $Q$, $\mathbf{a}$)
2:     $(\pi,\xi) = \text{UpperPlanes}(\mathbf{d}, Q, \mathbf{a})$
3:     for $\ell \leftarrow n,\dots,1$ step $-1$ do
4:         Initialize, for every $\beta \in \{0,\dots,\bar{y}\}$,
           $\left(\overline{\phi}^1(\beta), \delta^1(\beta), \overline{\chi}^1(\beta)\right) \leftarrow \begin{cases}(0,0,\emptyset), & \text{if } \beta \in \{0,\dots,a_1-1\}\\ (\pi_1^\ell(\beta), d_1, \xi_1^\ell(\beta)), & \text{otherwise}\end{cases}$
5:         for $k \leftarrow 2,\dots,\ell$ do
6:             for $\beta \leftarrow 0,\dots,\bar{y}$ do
7:                 if $\beta \geq a_k$ then
8:                     Let $\gamma_0 \leftarrow \overline{\phi}^{k-1}(\beta)$ and $\gamma_1 \leftarrow \overline{\phi}^{k-1}(\beta - a_k) + \pi_k^\ell(\beta)$
9:                     Update $\overline{\phi}^k(\beta) \leftarrow \max\,\{\gamma_0, \gamma_1\}$
10:                    if $\gamma_0 > \gamma_1$ then
11:                        Update $\delta^k(\beta) \leftarrow \delta^{k-1}(\beta)$ and $\overline{\chi}^k(\beta) \leftarrow \overline{\chi}^{k-1}(\beta)$
12:                    else if $\gamma_0 < \gamma_1$ then
13:                        Update $\delta^k(\beta) \leftarrow \delta^{k-1}(\beta - a_k) + d_k$ and $\overline{\chi}^k(\beta) \leftarrow \overline{\chi}^{k-1}(\beta - a_k) \cup \xi_k^\ell(\beta)$
14:                    else
15:                        Update $\delta^k(\beta) \leftarrow \max\,\{\delta^{k-1}(\beta), \delta^{k-1}(\beta - a_k) + d_k\}$    // Optimistic case
16:                        if $\delta^k(\beta) = \delta^{k-1}(\beta)$ then
17:                            Update $\overline{\chi}^k(\beta) \leftarrow \overline{\chi}^{k-1}(\beta)$
18:                        else
19:                            Update $\overline{\chi}^k(\beta) \leftarrow \overline{\chi}^{k-1}(\beta - a_k) \cup \xi_k^\ell(\beta)$
20:                        end if
21:                    end if
22:                else
23:                    Update $\overline{\phi}^k(\beta) \leftarrow \overline{\phi}^{k-1}(\beta)$, $\delta^k(\beta) \leftarrow \delta^{k-1}(\beta)$, and $\overline{\chi}^k(\beta) \leftarrow \overline{\chi}^{k-1}(\beta)$
24:                end if
25:            end for
26:        end for
27:    end for
28:    return $(\overline{\phi}^\bullet(\cdot), \overline{\chi}^\bullet(\cdot))$
29: end procedure
```

Algorithm 3 describes our DP for computing $\overline{\phi}^k(\beta)$, for all $k \in \{1,\dots,n\}$ and $\beta \in \{0,\dots,\bar{y}\}$, along with item-index sets $\overline{\chi}^k(\beta)$ of the greedily selected items among the first k items and their accompanying items greedily selected for the corresponding upper planes (i.e., the union of $\xi_i^k(\beta)$ over all selected items $i \in \{1,\dots,k\}$) computed by Algorithm 2. We observe that lines 4–26 describes an ℓ-item knapsack DP, for each $\ell \in \{1,\dots,n\}$, as back-iterated in line 3. Lines 15–20 breaks solution ties using the leader's objective function coefficients.

Algorithm 4 describes the iterative strengthening of QKP upper bounds through item-reordering (cf. Rodrigues et al. 2012, Algorithm 1). Line 6, in particular, reorders the items such that the first components of parameters $\hat{\mathbf{d}}$, $\hat{Q}$, and $\hat{\mathbf{a}}$ correspond to the selected items

Algorithm 4 Strengthening the upper-bounding value functions $\overline{\phi}^{\bullet}(\cdot)$ and item sets $\overline{\chi}^{\bullet}(\cdot)$

```
1: procedure UB$^+$(d, $Q$, a)
2:     Let $\hat{\mathbf{d}} \leftarrow \mathbf{d}$, $\hat{Q} \leftarrow Q$, and $\hat{\mathbf{a}} \leftarrow \mathbf{a}$
3:     $(\overline{\phi}, \overline{\chi}) = \mathrm{UB}(\hat{\mathbf{d}}, \hat{Q}, \hat{\mathbf{a}})$
4:     Let $\varphi^0 \leftarrow \infty$, $\varphi^1 \leftarrow \overline{\phi}^n(\bar{y})$
5:     while $\varphi^0 > \varphi^1$ do
6:         Reorder $(\hat{\mathbf{d}}, \hat{Q}, \hat{\mathbf{a}})$ such that the first $|\overline{\chi}^n(\bar{y})|$ components correspond to items in $\overline{\chi}^n(\bar{y})$
7:         $(\overline{\phi}, \overline{\chi}) = \mathrm{UB}(\hat{\mathbf{d}}, \hat{Q}, \hat{\mathbf{a}})$
8:         Update $\varphi^0 \leftarrow \varphi^1$ and $\varphi^1 \leftarrow \overline{\phi}^n(\bar{y})$
9:     end while
10:    return $(\overline{\phi}^{\bullet}(\cdot), \overline{\chi}^{\bullet}(\cdot))$ in terms of the original item-order
11: end procedure
```

under the previous reordering. The iterations terminate whenever the upper bounds do not improve (cf. Line 3). The procedure returns the value functions $\overline{\phi}^k(\beta)$, for all $k \in \{1, \dots, n\}$ and $\beta \in \{0, \dots, \bar{y}\}$, along with item-index sets $\overline{\chi}^k(\beta)$ in the original item-order.

3.3 Solving BQKP

The following proposition summarizes the results established in the previous two subsections, and its corollary provides the basis for pruning our branch-and-backtrack solution search tree.

Proposition 4 *For every $k \in \{1, \dots, n\}$ and $\beta \in \{\underline{y}, \dots, \bar{y}\}$,*

$$\underline{\phi}^k(\beta) \le \phi^k(\beta) \le \overline{\phi}^k(\beta).$$

The proof of this result is rather straightforward from Fomeni and Letchford (2014); Caprara et al. (1999); Gallo et al. (1980); Pisinger et al. (2007).

Corollary 1 *Let $\underline{\mathbf{x}} = [\underline{x}_i \in \{0,1\} : \underline{x}_i = 1 \Leftrightarrow i \in \underline{\chi}^k(\beta)] \in \{0,1\}^k$, which solves $\underline{\phi}^k(\beta)$, and let $\overline{\mathbf{x}} = [\bar{x}_i \in \{0,1\} : \bar{x}_i = 1 \Leftrightarrow i \in \overline{\chi}^n(\beta)] \in \{0,1\}^n$, which indicates a set of items including those that solve $\overline{\phi}^n(\beta)$. Then,*

$$\underline{\mathbf{x}}^\top Q^{[k]} \underline{\mathbf{x}} = \underline{\phi}^k(\beta) \le \phi^k(\beta) \le \overline{\phi}^k(\beta) \le \overline{\phi}^n(\beta) \le \overline{\mathbf{x}}^\top Q \overline{\mathbf{x}}.$$

Corollary 1 provides the basis for the bounds used in our branch-bound-and-backtrack algorithm (BBBA), which is described in Algorithm 5. Specifically, for any backtracked partial solution $\hat{\mathbf{x}} \in \{0,1\}^{n-k}$, the current problem node's zero-branch for x_k can be fathomed whenever $(\underline{\mathbf{x}}, 1, \hat{\mathbf{x}})$, where $\underline{\mathbf{x}} \in \{0,1\}^{k-1}$ solves $\underline{\phi}^{k-1}(\beta)$, induces a larger follower's objective function value than $(\bar{x}_1, \dots, \bar{x}_{k-1}, 0, \bar{x}_{k+1}, \dots, \bar{x}_n)$, thus indicating that item k must be selected (lines 10–11). To align with this result, we eliminated the outer loop of Algorithm 3 and fixed $\ell = n$ in our computational experiments. We will explore using the upper-bounding solutions induced by $\overline{\chi}^k(\beta)$ for $\phi^k(\beta)$, for all $k \in \{1, \dots, n-1\}$ and $\beta \in \{\underline{y}, \dots, \bar{y}\}$ in future studies.

BBBA differs from the branch-and-backtrack algorithm of Özaltın et al. (2010) in that we use the solutions to the lower- and upper-bounding value functions in lines 9 and 10 to

Algorithm 5 Branch-bound-and-backtrack algorithm (BBBA) for computing $\lambda^n(\beta)$

```
 1: procedure BBBA($\mathbf{d}$, $Q$, $\mathbf{a}$, $\beta$, $\underline{\chi}^\bullet(\cdot)$, $\overline{\chi}^\bullet(\cdot)$, $\psi^\bullet(\cdot)$)
 2:     Initialize $\nu \leftarrow 0$
 3:     Create node $\mathscr{P}^\nu$ with $k^\nu = n$, $\beta^\nu = \beta$, $\eta^\nu = 0$, and $\mathbf{x}^\nu = \mathbf{0}$
 4:     Initialize node list $\mathscr{M} \leftarrow \{\mathscr{P}^\nu\}$, lower bound $L \leftarrow 0$, and
            bounds $\underline{\phi}^0(\alpha) \leftarrow 0$ and $\overline{\phi}^0(\alpha) \leftarrow 0$, $\forall \alpha \in \{0,\dots,\beta\}$
 5:     while $\mathscr{M} \neq \emptyset$ do
 6:         Update $\mathscr{M} \leftarrow \mathscr{M} \setminus \{\mathscr{P}^\mu\}$
 7:         while $k^\mu \geq 1$ and $\eta^\mu + \psi^{k^\mu}(\beta^\mu) > L$ do
 8:             if $\beta^\mu \geq a_{k^\mu}$ then
 9:                 Let $\underline{\mathbf{x}} \leftarrow [\underline{x}_i = 1 \Leftrightarrow i \in \underline{\chi}^{k^\mu}(\beta^\mu) \vee x_i^\mu = 1]$ and $\overline{\mathbf{x}} \leftarrow [\overline{x}_i = 1 \Leftrightarrow i \in \overline{\chi}^n(\beta) \setminus \{k^\mu\}]$
10:                 if $\underline{x}_{k^\mu} = 1$ and $\left(\underline{\mathbf{x}}^\top Q \underline{\mathbf{x}} > \overline{\mathbf{x}}^\top Q \overline{\mathbf{x}}\right)$ then
11:                     Update $\eta^\mu \leftarrow \eta^\mu + d_{k^\mu}$, $x_{k^\mu}^\mu \leftarrow 1$, $\beta^\mu \leftarrow \beta^\mu - a_{k^\mu}$, and $L \leftarrow \max\ \{L, \eta^\mu\}$
12:                 else
                        // Branch on not selecting $k^\mu$
13:                     Update $\nu \leftarrow \nu + 1$
14:                     Create node $\mathscr{P}^\nu$ with $k^\nu = k^\mu - 1$, $\beta^\nu = \beta^\mu$, $\eta^\nu = \eta^\mu$, and $\mathbf{x}^\nu = \mathbf{x}^\mu$
15:                     Update $\mathscr{M} \leftarrow \mathscr{M} \cup \{\mathscr{P}^\nu\}$
                        // Proceed with selecting $k^\mu$
16:                     Update $\eta^\mu \leftarrow \eta^\mu + d_{k^\mu}$, $x_{k^\mu}^\mu \leftarrow 1$, $\beta^\mu \leftarrow \beta^\mu - a_{k^\mu}$, and $L \leftarrow \max\ \{L, \eta^\mu\}$
17:                 end if
18:             end if
19:             $k^\mu \leftarrow k^\mu - 1$
20:         end while
21:     end while
22:     return $L$
23: end procedure
```

fathom nodes, which incur additional computational burden. In order to reduce the computation, BBBA also uses the following upper-bounding value function for the leader's revenues (Özaltın et al. 2010):

$$\psi^k(\beta) \equiv \max_{\mathbf{x} \in \{0,1\}^k} \left\{ \sum_{i=1}^{k} d_i x_i : \sum_{i=1}^{k} a_i x_i \leq \beta \right\}, \qquad \forall k \in \{1,\dots,n\}, \forall \beta \in \{0,\dots,\overline{y}\},$$

in line 7. BBBA computes the leader's revenues for the optimal follower's QKP decisions corresponding to capacity decision $\beta \in \{\underline{y},\dots,\overline{y}\}$. Hence, the leader's optimal decision is given by:

$$y^* \in \operatorname*{arg\,max}_{y \in \{\underline{y},\dots,\overline{y}\}} \{ty + \lambda^n(y)\}.$$

We evaluate the performance of our exact-solution approach for solving BQKP compared with the two-phase approach for solving VF–BQKP in the next section.

4 Computational Experiments

4.1 Test Instances and Setup

We designed our experiments based on public datasets created by Beheshti et al. (2015) available online at `http://www.pitt.edu/~droleg/files/NBKP.htm`. We repurposed the relevant subset of their randomly generated parameters for their quadratic test instances, and we derived our test instances as follows:

- For our leader's objective function (i.e., $ty+\mathbf{d}^\top\mathbf{x}$), we fixed $t=-1$ and reused the linear component of their leader's objective function (cf. Beheshti et al. 2015, Equation 11);
- For our follower's objective function (i.e., $\mathbf{x}^\top Q\mathbf{x}$), we added their follower's linear objective function coefficients to the diagonal of their leader's (zero-diagonal[2]) Q-matrix; we report the Q-matrix off-diagonal densities Δ in alignment with their random-instance generation, where $\Delta = 0\%$ represent BKPs;
- For our follower's knapsack constraints (i.e., $\mathbf{a}^\top\mathbf{x} \le y$), we reused the left-hand side coefficients in their follower's knapsack constraints, and;
- For our leader's constraints on y (i.e., $\underline{y} \le y \le \overline{y}$), we used $\underline{y} = \max_{i\in\{1,\dots,n\}} a_i + 1$ and $\overline{y} = \sum_{i=1}^{n} a_i - 1$ in accordance with Assumption **(A4)**.

Details on the random-parameter generation are described further by Beheshti et al. (2015). We note the generated Q matrices induce test instances with negative semi-definite QKPs; however, this assumption can be made for the two-phase approach for solving VF–BQKP without loss of generality (cf. Billionnet and Elloumi 2007).

We implemented Algorithms 1–5 for solving BQKP in MATLAB 2017a. We also used MATLAB 2017a to implement our two-phase approach of solving VF–BQKP (i.e., solving all the QPs for all knapsack capacities $\beta \in \{\underline{y},\dots,\overline{y}\}$ and the MIQCP reformulation of BQKP as described in Section 2) using two state-of-the-art commercial solvers: Gurobi 7.5.1 (Gurobi Optimization, Inc. 2017) and BARON 17.8.9[3] (Sahinidis 2017). We conducted our experiments on a Dell PowerEdge R930 machine with Intel® Xeon® CPU E7-8890 v3 @ 250GHz (4 NUMA nodes, 144 logical cores) and 1TB RAM running Windows Server 2012. We report 95% confidence intervals of computation times from solving up to 20 test instances.

4.2 Results and Discussion

Between the two commercial solvers, we found Gurobi 7.5.1 to significantly perform better on initial experiments in our computing environment. Table 1 summarizes the performance of the commercial solvers in executing the two-phase approach for solving VF–BQKP—that is, the total time for solving all QPs for all knapsack capacities, as well as the MIQCP reformulation of BQKP—for the simplest test instances: $n = 25$ knapsack items, number of RHSs $b \equiv \overline{y} - \underline{y} + 1$ average of 11,052.6, and uncorrelated parameters. Thus, subsequent findings report on comparisons of our solution method against Gurobi 7.5.1 only.

Table 1 Performance of State-of-the-art Commercial Solvers on the Simplest Test Instances

			Off-diagonal	Computation Time (s)	
	Average	Correlated	Density	95% Confidence Interval	
n	b	Q and $\mathbf{a}$	of Q (Δ)	Gurobi 7.5.1	BARON 17.8.9
25	11052.6	Uncorrelated	0%	[222.8, 223.8]	[5507.8, 5533.2]
25	11052.6	Uncorrelated	10%	[340.4, 342.4]	[5555.4, 5578.1]
25	11052.6	Uncorrelated	50%	[862.7, 874.2]	[6623.9, 6647.5]
25	11052.6	Uncorrelated	100%	[1138.0, 1152.9]	[9281.6, 9283.3]

[2] Beheshti et al. (2015) factored the diagonal entries of their leader's Q-matrix into the linear component of their leader's objective function.

[3] BARON 17.8.9 uses CPLEX 12.7.1 as the default linear and mixed-integer programming solver.

Table 2 summarizes the performance of our solution method BBBA and an enhanced variant BBBA$^+$ in solving BQKPs, as well as Gurobi 7.5.1 solving the corresponding MIQCPs and all their 0–1 QPs. BBBA and BBBA$^+$ implement the straightforward and cutting upper-planes approaches, respectively. (We recall the straightforward upper-planes approach executes Algorithms 2 and 3 once, while the cutting upper-planes approach executes the same algorithms iteratively on refined orderings of the knapsack items (cf. Algorithm 4.))

We note that, except in six groups of test instances (i.e., the BKPs with $\Delta = 0\%$, for all $n > 25$), BBBA and BBBA$^+$ significantly outperform Gurobi 7.5.1. We further note that BBBA significantly outperforms BBBA$^+$ for test instances with $n > 25$, which indicates

Table 2 Performance of BBBA, BBBA$^+$, and Gurobi 7.5.1 for Multiple Test Instances

			Off-diagonal	Computation Time (s)		
	Average	Correlated	Density	95% Confidence Interval		
n	b	Q and **a**	of Q (Δ)	BBBA	BBBA$^+$	Gurobi 7.5.1
25	11052.6	Uncorrelated	0%	[110.1, 110.4]	[84.0, 84.3]	[222.8, 223.8]
25	11052.6	Uncorrelated	10%	[116.2, 116.7]	[92.8, 93.2]	[340.4, 342.4]
25	11052.6	Uncorrelated	50%	[122.3, 122.8]	[98.6, 99.0]	[862.7, 874.2]
25	11052.6	Uncorrelated	100%	[122.6, 123.1]	[98.7, 99.1]	[1138.0, 1152.9]
25	11052.6	Correlated	0%	[126.7, 127.1]	[98.2, 98.6]	[546.1, 549.6]
25	11052.6	Correlated	10%	[128.0, 128.5]	[103.3, 103.7]	[1066.1, 1077.3]
25	11052.6	Correlated	50%	[125.4, 125.9]	[104.3, 104.7]	[2467.9, 2491.4]
25	11052.6	Correlated	100%	[122.6, 123.1]	[104.5, 104.9]	[3568.7, 3616.9]
25	11148.0	Highly correlated	0%	[116.2, 116.6]	[100.7, 101.1]	[804.8, 808.9]
25	11148.0	Highly correlated	10%	[117.3, 117.7]	[103.3, 103.7]	[1856.4, 1872.4]
25	11148.0	Highly correlated	50%	[115.3, 115.6]	[102.4, 102.8]	[2904.0, 2924.5]
25	11148.0	Highly correlated	100%	[112.3, 112.7]	[102.6, 103.0]	[4430.7, 4481.8]
50	22937.4	Uncorrelated	0%	[592.8, 593.6]	[682.3, 683.3]	[441.6, 442.9]
50	22937.4	Uncorrelated	10%	[609.3, 610.1]	[719.2, 720.8]	[1017.0, 1023.8]
50	22937.4	Uncorrelated	50%	[613.0, 613.9]	[736.8, 738.3]	[6388.2, 6547.0]
50	22937.4	Uncorrelated	100%	[599.2, 600.1]	[739.4, 741.6]	> 14400[a]
50	22937.4	Correlated	0%	[628.9, 629.7]	[725.5, 726.7]	[375.7, 378.1]
50	22937.4	Correlated	10%	[632.1, 632.9]	[727.2, 728.5]	[1805.1, 1828.7]
50	22937.4	Correlated	50%	[622.9, 623.8]	[733.7, 735.3]	> 14400[a]
50	22937.4	Correlated	100%	[604.4, 605.3]	[715.2, 716.2]	> 14400[b]
50	23794.8	Highly correlated	0%	[663.3, 665.2]	[765.2, 767.3]	[1043.5, 1047.1]
50	23794.8	Highly correlated	10%	[659.6, 661.1]	[765.9, 767.9]	[4189.4, 4263.7]
50	23794.8	Highly correlated	50%	[652.3, 654.1]	[772.7, 775.2]	[50283.0, 52522.1]
50	23794.8	Highly correlated	100%	[638.8, 640.6]	[769.9, 771.9]	> 14400[a]
75	34875.6	Uncorrelated	0%	[1724.8, 1726.6]	[2065.1, 2067.5]	[865.9, 868.5]
75	34875.6	Uncorrelated	10%	[1758.9, 1761.0]	[2149.7, 2153.9]	[2246.4, 2258.4]
75	34875.6	Uncorrelated	50%	[1739.5, 1742.2]	[2276.3, 2284.0]	[7314.9, 7371.6]
75	34875.6	Uncorrelated	100%	[1742.7, 1745.1]	[2604.2, 2614.8]	> 14400[b]
75	34875.6	Correlated	0%	[1817.2, 1819.7]	[2540.5, 2546.6]	[732.1, 734.0]
75	34875.6	Correlated	10%	[1832.8, 1835.3]	[2530.3, 2536.9]	[3135.3, 3147.4]
75	34875.6	Correlated	50%	[1833.3, 1835.7]	[2679.8, 2690.7]	> 14400[b]
75	34875.6	Correlated	100%	[1840.9, 1843.6]	[2475.5, 2481.5]	> 14400[b]
75	36186.8	Highly correlated	0%	[1888.0, 1891.6]	[2582.8, 2588.9]	[674.0, 676.6]
75	36186.8	Highly correlated	10%	[1979.2, 1985.3]	[2572.6, 2579.4]	[3269.3, 3299.9]
75	36186.8	Highly correlated	50%	[2074.7, 2081.4]	[2659.3, 2671.0]	> 14400[b]
75	36186.8	Highly correlated	100%	[2109.4, 2119.6]	[2587.3, 2595.6]	> 14400[b]

[a] The 95% confidence interval exceeds 4 hours. Most test instances could not be solved within 4 hours; those solvable within 4 hours took longer to solve with Gurobi 7.5.1 than BBBA and BBBA$^+$.

[b] The 95% confidence interval exceeds 4 hours. All test instances could not be solved within 4 hours.

that the additional computation for enhancing the QKP upper bounds scales poorly to the BQKP size. The latter observations hold for both sparse and dense Q matrices.

Table 3 summarizes the performance of BBBA and BBBA$^+$ on large test instances. We observe from the tables that BBBA and BBBA$^+$ computation times appear to increase only in the number of knapsack items (n) and number of RHSs (b): in practice, they appear to run in pseudo-polynomial time. In contrast, Gurobi 7.5.1 computation times appear to have partially ordered, exponentially fast increases with respect to multiple factors. We confirm our observations using regression analyses in the next section.

Table 3 Performance of BBBA and BBBA$^+$ on Large Test Instances

			Off-diagonal	Computation Time (s) 95% Confidence Interval	
	Average	Correlated	Density		
n	b	Q and **a**	of Q (Δ)	BBBA	BBBA$^+$
100	47336.6	Uncorrelated	0%	[6300.4, 6359.5]	[6793.7, 6813.0]
100	47336.6	Uncorrelated	10%	[6461.7, 6490.9]	[6856.1, 6884.3]
100	47336.6	Uncorrelated	50%	[7092.8, 7151.7]	[8851.2, 8927.8]
100	47336.6	Uncorrelated	100%	[6569.0, 6610.6]	[8730.7, 8802.9]
100	47336.6	Correlated	0%	[6523.6, 6543.0]	[9509.7, 9548.3]
100	47336.6	Correlated	10%	[6112.1, 6130.3]	[9770.1, 9814.5]
100	47336.6	Correlated	50%	[6107.4, 6134.3]	[9305.5, 9350.2]
100	47336.6	Correlated	100%	[6000.7, 6027.3]	[9160.9, 9187.8]
100	47939.8	Highly correlated	0%	[6819.2, 6858.5]	[7557.1, 7593.4]
100	47939.8	Highly correlated	10%	[6719.6, 6748.8]	[7189.2, 7211.3]
100	47939.8	Highly correlated	50%	[6120.6, 6134.5]	[7504.1, 7523.4]
100	47939.8	Highly correlated	100%	[6312.4, 6341.1]	[8168.2, 8190.0]

4.3 Further Analyses

We used R 3.4.1 (R Core Team 2017) to conduct our regression analyses of computation time against the features of all individual test instances, which are summarized in Tables 2 and 3. Our starting regression models included the key regressors: number of knapsack items n and number of RHSs b, which determine the number of iterations and branches in Algorithms 1–5 and solving the MIQCP reformulation of BQKP. Through multiple iterations of residual analyses, variable transformations, and forward and backward selection of regressors with and without interaction effects, we found the following regression models to best fit our data:

$$\log(\text{BBBA time}) = -9.13519 + 1.92354 \cdot \log n + 0.81674 \cdot \log b \tag{1a}$$

$$\text{BBBA time} = 0.0001078046 \cdot n^{1.92354} \cdot b^{0.81674}, \tag{1b}$$

along with:

$$\log(\text{BBBA}^+\text{ time}) = -11.02172 + 2.14211 \cdot \log n + 0.92874 \cdot \log b \tag{2a}$$

$$\text{BBBA}^+\text{ time} = 0.00001634285 \cdot n^{2.14211} \cdot b^{0.92874}, \tag{2b}$$

and:

$$\log(\text{Gurobi 7.5.1 time}) = \begin{cases} \text{baseline} \equiv 0 & \\ \qquad + 0.013629 \cdot n & \\ \qquad + 0.567823 \cdot \log b & \\ \qquad + 0.004625 \cdot \Delta & \\ \qquad + 0.000528 \cdot n \cdot \Delta, & \text{if Uncorrelated } Q \text{ and } \mathbf{a} \\ \text{baseline} + 1.480660 & \\ \qquad - 0.023628 \cdot n & \\ \qquad - 0.031607 \cdot \Delta & \\ \qquad + 0.001241 \cdot n \cdot \Delta, & \text{if Correlated } Q \text{ and } \mathbf{a} \\ \text{baseline} + 2.242742 & \\ \qquad - 0.028339 \cdot n & \\ \qquad - 0.026478 \cdot \Delta & \\ \qquad + 0.000855 \cdot n \cdot \Delta, & \text{if Highly Correlated } Q \text{ and } \mathbf{a}. \end{cases} \tag{3}$$

Regression Models (1), (2), and (3) have R^2 values of 0.9794, 0.9889 and 0.9924, respectively, and sufficiently satisfy the linearity assumptions. All coefficients are significant (p-value < 0.01) except for Δ in the Gurobi 7.5.1 model for Uncorrelated Q and $\mathbf{a}$ (p-value $= 0.05121$); we retain Δ because of significant interaction effects. We note that, for all regressed values, the positive coefficients of Model (3) increases faster than the negative coefficients. Thus, we find that, for our test instances, solving BQKP using BBBA executes in pseudo-polynomial time $O(n^2 \cdot b)$, while solving its MIQCP reformulation remains intractable. Finally, we note that BKP can be solved in $O(n \cdot b)$ by DP-based approaches (cf. Brotcorne et al. 2009).

5 Conclusion

In this note we considered BQKP, which models two-level problems where the leader appropriates a budget for the follower who solves an instance of 0–1 QKP. We first presented the single-level, value-function reformulation of BQKP as an MIQCP, which can be solved by state-of-the-art commercial solvers. We then described our exact-solution approach, BBBA, which uses DPs to find lower- and upper-bounding solutions to QKP, which, in turn, are used in a branch-and-backtrack procedure. We showed through computational experiments that BBBA outperforms commercial software packages solving the MIQCP reformulation.

Our approach exploits the special structure of BQKP, particularly QKP in the second level. QKPs are well studied in the literature (cf. Pisinger 2007) for which efficient heuristics and exact-solution algorithms exist (e.g., Caprara et al. 1999; Fomeni and Letchford 2014; Gallo et al. 1980; Rodrigues et al. 2012). We exploited the simplest variants of the heuristics—computable by DPs—to establish QKP bounds used in pruning our branch-and-backtrack solution search tree in BBBA. Possible enhancements to BBBA include using more-efficient heuristics for QKP (Fomeni and Letchford 2014) and/or applying Lagrangian relaxations (Billionnet et al. 1999; Billionnet and Soutif 2004; Caprara et al. 1999).

BQKP in this paper differs from linear-quadratic and quadratic-quadratic bilevel programs considered in the literature (LQBPs and QQBPs, respectively) in that LQBPs and QQBPs involve only continuous decision variables in their second-level convex

QPs, for which approaches to the single-level reformulation afforded by the Karush-Kuhn-Tucker (KKT) optimality conditions have been proposed (e.g., Adasme and Lisser 2016; Júdice and Faustino 1994; Qin et al. 2017; Vicente et al. 1994; Wang et al. 2008). The single-level reformulation of BQKP in this paper, on the other hand, is afforded by value functions; however, the resulting MIQCP is computationally challenging for commercial state-of-the-art solvers as demonstrated in our experiments, which motivated the development of the proposed exact solution methods.

Furthermore, BQKP in this paper differs from the nonlinear bilevel knapsack problems considered by Beheshti et al. (2015) as the latter assumes nonlinear leader's objective function (i.e., quadratic or fractional), but a linear 0–1 knapsack problem at the follower's level. Therefore, the current study and the study by Beheshti et al. (2015) complement each other as they consider nonlinearities at different levels of the decision-making hierarchy. Note that the models considered in both of the papers can be viewed as natural generalizations of BKP by Brotcorne et al. (2009).

Consequently, in future work one can focus on methods that exploit the ideas from these studies to accommodate nonlinearities at both levels. In particular, within the BBBA framework one needs to combine the branching mechanism from the current paper with the approaches that compute the global lower bound for the leader's objective function from Beheshti et al. (2015). Note that the branching mechanism in this paper depends on the outputs of Algorithms 1–4, while the global lower bound computation in Beheshti et al. (2015) is specifically customized to handle quadratic and fractional 0–1 types of the leader's objective function.

We make some final remarks on the novel contributions of this note. First, we introduced BQKP as a bilevel program where the second-level optimization problem is integer and nonlinear; this expands the bilevel programming literature, where models with continuous and/or linear second-level programs are the norm. Second, we presented a value-function reformulation of BQKP as an MIQCP, which, with recent advancements in the commercial state-of-the-art solvers, have become computable, albeit still intractable for large test instances. Third, we developed an exact-solution approach that performs well in practice.

Acknowledgements The authors are grateful to Dr. Behdad Beheshti and three anonymous reviewers for their constructive comments on earlier versions of this note. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL/RW or the U.S. Government.

References

Adasme P., Lisser A. (2016). A computational study for bilevel quadratic programs using semidefinite relaxations. European Journal of Operational Research, 254(1), 9–18.

Aiyoshi E., Shimizu K. (1981). Hierarchical decentralized systems and its new solution by a barrier method. IEEE Transactions on Systems, Man, and Cybernetics, 11(6), 444–449.

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

Bard J. F., Falk J. E. (1982). An explicit solution to the multi-level programming problem. Computers & Operations Research, 9(1), 77–100.

Beheshti B., Özaltın O. Y., Zare M. H., Prokopyev O. A. (2015). Exact solution approach for a class of nonlinear bilevel knapsack problems. Journal of Global Optimization, 61(2), 291–310.

Beheshti B., Prokopyev O. A., Pasiliao E. L. (2016). Exact solution approaches for bilevel assignment problems. Computational Optimization and Applications, 64(1), 215–242.

Ben-Ayed O., Blair C. E. (1990). Computational difficulties of bilevel linear programming. Operations Research, 38(3), 556–560.

Billionnet A., Elloumi S. (2007). Using a mixed integer quadratic programming solver for the unconstrained quadratic 0–1 problem. Mathematical Programming, 109(1), 55–68.

Billionnet A., Soutif E. (2004). An exact method based on lagrangian decomposition for the 01 quadratic knapsack problem. European Journal of Operational Research, 157(3), 565 – 575.

Billionnet A., Faye A., Soutif É. (1999). A new upper bound for the 0–1 quadratic knapsack problem. European Journal of Operational Research, 112(3), 664 – 672.

Bracken J., McGill J. T. (1973). Mathematical programs with optimization problems in the constraints. Operations Research, 21(1), 37–44.

Bracken J., McGill J. T. (1974). Defense applications of mathematical programs with optimization problems in the constraints. Operations Research, 22(5), 1086–1096.

Bracken J., McGill J. T. (1978). Production and marketing decisions with multiple objectives in a competitive environment. Journal of Optimization Theory and Applications, 24(3), 449–458.

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

Brotcorne L., Hanafi S., Mansi R. (2013). One-level reformulation of the bilevel knapsack problem using dynamic programming. Discrete Optimization, 10(1), 1–10.

Candler W., Townsley R. (1982). A linear two-level programming problem. Computers & Operations Research, 9(1), 59–76.

Caprara A., Pisinger D., Toth P. (1999). Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing, 11(2), 125–137.

Chen W.-A., Zhu Z., Kong N. (2018). A lagrangian decomposition approach to computing feasible solutions for quadratic binary programs. Optimization Letters, 12(1), 155–169.

Colson B., Marcotte P., Savard G. (2005). A trust-region method for nonlinear bilevel programming: Algorithm and computational experience. Computational Optimization and Applications, 30(3), 211–227.

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

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

Fomeni F. D., Letchford A. N. (2014). A dynamic programming heuristic for the quadratic knapsack problem. INFORMS Journal on Computing, 26(1), 173–182.

Gallo G., Hammer P. L., Simeone B. (1980). Quadratic Knapsack Problems, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 132–149.

Gurobi Optimization, Inc. (2017). Gurobi optimizer reference manual. Available at `http://www.gurobi.com`. Accessed 24 July 2017.

Hansen P., Jaumard B., Savard G. (1992). New branch-and-bound rules for linear bilevel programming. SIAM Journal on Scientific and Statistical Computing, 13(5), 1194–1217.

IBM Corp. (2017). IBM ILOG CPLEX optimization studio, *reference manual*. Available at `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/pdf/usrcplex.pdf`. Accessed 24 July 2017.

Jeroslow R. G. (1985). The polynomial hierarchy and a simple model for competitive analysis. Mathematical Programming, 32(2), 146–164.

Júdice J., Faustino A. (1994). The linear-quadratic bilevel programming problem. INFOR: Information Systems and Operational Research, 32(2), 87–98.

Kellerer H., Pferschy U., Pisinger D. (2004). Knapsack Problems. Springer, Berlin.

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

Martello S., Toth P. (1990). Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Inc., New York, NY, USA.

MATLAB (2017a). version 9.2.0.556344 (r2017a). Available at `https://www.mathworks.com/products/matlab.html`. Accessed 5 June 2017.

Migdalas A., Pardalos P. M., Värbrand P. (1998). Multilevel Optimization: Algorithms and Applications, vol 20. Springer.

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

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

Pisinger D. (2007). The quadratic knapsack problem—a survey. Discrete Applied Mathematics, 155(5), 623–648.

Pisinger W. D., Rasmussen A. B., Sandvik R. (2007). Solution of large quadratic knapsack problems through aggressive reduction. INFORMS Journal on Computing, 19(2), 280–290.

Qin S., Le X., Wang J. (2017). A neurodynamic optimization approach to bilevel quadratic programming. IEEE Transactions on Neural Networks and Learning Systems, doi:10.1109/TNNLS.2016.2595489.

R Core Team (2017). R: A language and environment for statistical computing. Available at `https://www.R-project.org`. Accessed 22 August 2017.

Rhys J. M. W. (1970). A selection problem of shared fixed costs and network flows. Management Science, 17(3), 200–207.

Rodrigues C. D., Quadri D., Michelon P., Gueye S. (2012). 0-1 quadratic knapsack problems: An exact approach based on a t-linearization. SIAM Journal on Optimization, 22(4), 1449–1468.

Sahinidis N. V. (2017). BARON 17.8.9: Global optimization of mixed-integer nonlinear programs, *user's manual*. Available at `http://www.minlp.com/downloads/docs/baron manual.pdf`. Accessed 15 September 2017.

Stozhkov V., Boginski V., Prokopyev O. A., Pasiliao E. L. (2017). A simple greedy heuristic for linear assignment interdiction. Annals of Operations Research, 249(1), 39–53.

Trapp A. C., Prokopyev O. A. (2015). A note on constraint aggregation and value functions for two-stage stochastic integer programs. Discrete Optimization, 15, 37–45.

Trapp A. C., Prokopyev O. A., Schaefer A. J. (2013). On a level-set characterization of the value function of an integer program and its application to stochastic programming. Operations Research, 61(2), 498–511.

Vicente L., Savard G., Júdice J. (1994). Descent approaches for quadratic bilevel programming. Journal of Optimization Theory and Applications, 81(2), 379–399.

Wang G., Wan Z., Wang X., Lv Y. (2008). Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. Computers and Mathematics with Applications, 56(10), 2550–2555.

Witzgall C. (1975). Mathematical methods of site selection for electronic message systems (EMS). Tech. Rep. 76, NASA STI/Recon.

Zare M. H., Borrero J. S., Zeng B., Prokopyev O. A. (2017). A note on linearized reformulations for a class of bilevel linear integer problems. Annals of Operations Research, DOI 10.1007/s10479-017-2694-x.