The bilevel knapsack problem with stochastic right-hand sides

Osman Y. Özaltın, Oleg A. Prokopyev*, Andrew J. Schaefer

Department of Industrial Engineering, University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA 15261, USA

A R T I C L E   I N F O

Article history:
Received 3 March 2010
Accepted 9 April 2010
Available online 20 April 2010

Keywords:
Bilevel programming
Stochastic integer programming
Value functions

A B S T R A C T

We introduce the bilevel knapsack problem with stochastic right-hand sides, and provide necessary and sufficient conditions for the existence of an optimal solution. When the leader’s decisions can take only integer values, we present an equivalent two-stage stochastic programming reformulation with binary recourse. We develop a branch-and-cut algorithm for solving this reformulation, and a branch-and-backtrack algorithm for solving the scenario subproblems. Computational experiments indicate that our approach can solve large instances in a reasonable amount of time.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Bilevel programs [3,5,16] model the hierarchical relationship between two autonomous, and possibly conflicting, decision makers: the leader and the follower. This hierarchical relationship results from the fact that the follower’s problem is affected by the decision of the leader. Moreover, the follower’s decision in return affects the leader’s problem.

The bilevel knapsack problem was first considered by Dempe and Richter [6]. In this problem, the follower solves a 0–1 knapsack problem subject to the capacity set by the leader. The leader earns a profit from the items selected by the follower, and both decision makers seek to maximize their own profits. Dempe and Richter [6] formulated this problem as a mixed-integer bilevel program, and proposed a branch-and-bound algorithm. Recently, Brotcorne et al. [2] considered the same problem, and developed a dynamic programming algorithm that outperformed Dempe and Richter’s [6] branch–and-bound algorithm.

We introduce the bilevel knapsack problem with stochastic right-hand sides (BKPS). BKPS is a stochastic extension of the bilevel knapsack problem where the leader’s decision has an uncertain effect on the follower’s knapsack capacity. We model this uncertainty using a finite set of scenarios. We discuss necessary and sufficient conditions for the existence of an optimal solution. When the leader’s decisions can take only integer values, we give an equivalent two-stage stochastic programming reformulation. We develop a branch-and-cut algorithm for solving this reformulation, and a branch-and-backtrack algorithm for solving the scenario subproblems.

Brotcorne et al. [2] identified an application of the bilevel knapsack problem in revenue management, where a company (i.e. the leader) determines the number of units to sell by itself, and handing the remainder over to an intermediary (i.e. the follower). In this context, BKPS arises when there is uncertainty in the number of units transferred to the intermediary. For example, in the distribution of perishable goods [7], some items may be spoiled during the shipment process.

2. Formulation and properties of BKPS

Consider a set of n items where each item j ∈ {1, . . . , n} has an associated weight qj ∈ Z+ and two revenues: the follower’s revenue cj ∈ R1, and the leader’s revenue dj ∈ R1. The follower must solve a knapsack problem to maximize its own objective subject to a capacity h(ω, y) that depends on the leader’s choice of y as well as a discretely distributed random variable ω ∈ Ω. This yields the following stochastic bilevel program:

[BKPS] maximize f(y, X) = ty + Eω [dT x(ω, y)]
subject to h ≤ y ≤ b, y ∈ R1,
x(ω, y) ∈ R(h(ω, y)) ∀ω ∈ Ω,

where R(h(ω, y)) = argmax |{cT x : aT x ≤ h(ω, y), x ∈ [0, 1]n}|
and the follower’s rational reaction set. For y ∈ [h, b], X is an |Ω| × n binary matrix whose rows represent the subset of items selected by the follower under the scenario ω ∈ Ω, i.e. x(ω, y). We assume that h(ω, y) : Ω → R1 is a nondecreasing function of y, and that h(ω, b) is finite for all ω ∈ Ω.

There are two classes of bilevel programs [12]. In the optimistic case, whenever the rational reaction set is not a singleton, the
follower implements a so-called strong solution that maximizes the objective of the leader. In the pessimistic case, the leader assumes that whenever the follower is facing ties, she selects a so-called weak solution that minimizes the leader’s objective. The optimistic case might arise in a collaborative environment, and the pessimistic case in an adversarial environment.

For $1 \leq k \leq n$ and any $\beta \in \mathbb{R}_1$, define the follower’s value function as:
\[
\psi_k(\beta) = \max_{x \in \{0,1\}^k} \left\{ \sum_{j=1}^{k} a_j x_j : \sum_{j=1}^{k} a_j x_j \leq \beta \right\}.
\]

**Lemma 1** ([15]), $\psi_k(\beta)$ is piecewise constant and monotonically nondecreasing over $\beta$. Moreover, it can have discontinuities only at integer values of $\beta$.

**Lemma 1** holds since we assume that $a \in \mathbb{Z}_n$, Dempe and Richter [6] proved that an optimal solution to the deterministic bilevel knapsack problem exists if $t < 0$. Moreover, they showed that if $t > 0$ then either the optimal solution occurs at $y = \bar{y}$, or there is no optimal solution. This result of Dempe and Richter [6] extends to the BKPS, as we show in **Proposition 1**.

**Proposition 1.** BKPS has an optimal solution if and only if $t \leq 0$ or $f(\beta,X) + t \leq f(\bar{y},X)$ for all $\beta \in \{\bar{b}, \bar{b}\} \cap \mathbb{Z}_1$.

**Proof.** From **Lemma 1**, it follows that $f(y,X) = ty + E_\omega [d^T \mathbf{x}(\omega, y)]$ is a piecewise linear, right-continuous function of $y$ where each linear portion has slope $t$. Moreover, the endpoints of each continuous part occur at integer values of $y$.

"⇒" Suppose BKPS has an optimal solution when $t > 0$ and $f(\beta,X) + t > f(\bar{y},X)$ for some $\beta \in \{\bar{b}, \bar{b}\} \cap \mathbb{Z}_1$. The supremum of $f(y,X)$ over each linear portion occurs at its right endpoint as $t > 0$. Therefore, the optimal solution can only be attained at $\bar{y}$ as $f(y,X)$ is right-continuous. Let $\beta^*$ be the largest value in $\{\bar{b}, \bar{b}\} \cap \mathbb{Z}_1$ that satisfies $f(\beta^*,X) + t > f(\bar{y},X)$. If $\beta^* + 1 = \bar{b}$, then BKPS has no optimal solution since the value $f(\beta^*,X) + t$, which is strictly greater than $f(\bar{y},X)$, is not attained. Otherwise, the definition of $\beta^*$ implies that $f(\beta^* + 1,X) + t \leq f(\bar{y},X) < f(\beta^*,X) + t \Rightarrow f(\beta^* + 1,X) < f(\beta^*,X)$. Then
\[
t(\beta^* + 1) + E_\omega [d^T \mathbf{x}(\omega, \beta^*)] = f(\beta^*,X) + t > f(\beta^*,X)
\]

From (2), $f(y,X)$ is discontinuous at $\beta^* + 1$ and the value $f(\beta^*,X) + t$, which is strictly greater than $f(\bar{y},X)$, is not attained. As a result, $(\bar{b}, \bar{b})$ is not the optimal solution, which implies BKPS has no optimal solution.

"⇐" When $t \leq 0$, the supremum of $f(y,X)$ over each linear portion occurs at its left endpoint. Therefore, an optimal solution is always attained as $f(y,X)$ is right-continuous. If $t > 0$ and $f(\beta,X) + t \leq f(\bar{y},X)$ for all $\beta \in \{\bar{b}, \bar{b}\} \cap \mathbb{Z}_1$, then $(\bar{b}, \bar{b})$ is the optimal solution since each linear portion of $f(y,X)$ has slope $t$. □

**Remark 1.** When $t \geq 0$, BKPS may not have an optimal solution even if $f(\beta,X) < f(\bar{y},X)$ for all $\beta \in \{\bar{b}, \bar{b}\} \cap \mathbb{Z}_1$, as illustrated in Fig. 1.

Let $p_\omega$ denote the probability of scenario $\omega$, where $\sum_{\omega \in \Omega} p_\omega = 1$. For all $\omega \in \Omega$, define the scenario subproblem BKPS(\omega) as:

\[
\begin{align*}
\text{[BKPS(\omega)]} \text{ maximize } & f_\omega(y,X) = ty + d^T \mathbf{x}(\omega, y) \\
\text{subject to } & \bar{b} \leq y \leq \bar{b}, \quad y \in \mathbb{R}_1, \\
& \mathbf{x}(\omega, y) \in \mathcal{R}(h(\omega, y))
\end{align*}
\]

**Proposition 2** provides a sufficient condition for the existence of an optimal solution to BKPS in terms of the individual scenario subproblems BKPS(\omega).

In the rest of this paper, we focus on a discrete version of BKPS, where the leader’s capacity decisions can take only integer values, i.e. $y \in \mathbb{Z}_1^k$ in problem (1), and $h(\omega, y) : \Omega \times \mathbb{Z}_1^k \rightarrow \mathbb{Z}_1^k$. Following the convention of Broctorne et al. [2], we term this problem as BKPSd. Note that an optimal solution always exists for BKPSd. Moreover, if BKPS has an optimal solution, it can be obtained by solving BKPSd, since $f(y,X)$ can have discontinuities only at integer values of $y$. These two observations are also valid for the deterministic bilevel knapsack problem [2].

3. Model reformulation of BKPSd

Bilevel programs are generally non-convex and discontinuous. Hansen et al. [9] proved that the linear bilevel programming problem is strongly NP-hard. Moreover, Vicente et al. [19] showed that checking local optimality in the linear bilevel programming problem is also NP-hard. Many combinatorial optimization problems can be reduced to bilevel programs, including mixed-binary programming, generalized linear complementarity, bilinear disjoint programming, and the traveling salesman problem [13].

We formulate BKPSd as a two-stage stochastic program with binary first- and second-stage problems using the follower’s value function $\psi_k(\beta)$. Ahmed et al. [1] and Kong et al. [10] gave other
examples of value function reformulation in stochastic integer programming. For \( i = b, \ldots, b \), define the binary first-stage variable \( z_i = 1 \) if \( y = i \), and \( z_i = 0 \) otherwise, and let \( t_i = t \times i \), \( \forall i \). Then the optimistic case of BKP\(s_d \) can be formulated as:

\[
\begin{align*}
\text{maximize } & f(z, X) = \sum_{i=\bar{b}}^{b} t_i z_i + \mathbb{E}_\omega \left[ d^T x_\omega(z, z) \right] \\
\text{subject to } & \sum_{i=\bar{b}}^{b} z_i = 1, \\
& z_i \in \{0, 1\}, \; i = \bar{b}, \ldots, b,
\end{align*}
\]

where,

\[
\begin{align*}
x_\omega(z, z) & \in \text{argmax} \sum_{j=1}^{n} d_j x_j \\
& \text{subject to } \sum_{j=1}^{n} c_j x_j \geq \psi_n \left( h\left( \omega, \sum_{i=\bar{b}}^{b} t_i z_i \right) \right), \\
& \sum_{j=1}^{n} a_j x_j \leq h\left( \omega, \sum_{i=\bar{b}}^{b} t_i z_i \right),
\end{align*}
\]

The first-stage objective (4a) maximizes the expected return of the leader, and constraint (4b) chooses an integer capacity between \( \bar{b} \) and \( b \). The second-stage objective (5a) maximizes the leader’s benefit gained from the items selected by the follower. Constraints (5b) and (5c) ensure that the follower maximizes its own objective function subject to the capacity constraint. Without loss of generality, we focus on the optimistic case in the remainder of this paper, as changing the argmax to argmin in (5a) captures the pessimistic case.

4. Solution approach

Imposing integrality restrictions on second-stage variables typically increases the difficulty of stochastic programs, as in general the expected recourse function becomes nonconvex and discontinuous [18]. Algorithms developed in the literature for solving stochastic programs with integer recourse utilize cutting planes and/or branch-and-bound techniques in combination with decomposition methods that exploit the block separability of the underlying problem structure [8,17].

Let \( u = \max_{\omega, h(z, \bar{b})} \{ h(\omega, \bar{b}) \} \). Note that \( u < \infty \) as assumed that \( h(\omega, \bar{b}) \) is finite \( \forall \omega \in \Omega \). For all \( b \leq u, b \in \mathbb{Z}_+ \), denote the value function of subproblem (5) by:

\[
\lambda_n(\beta) = \max_{x \in \mathbb{Z}_+^n} \left\{ \sum_{j=1}^{n} d_j x_j : \sum_{j=1}^{n} c_j x_j \geq \psi_n(\beta), \sum_{j=1}^{n} a_j x_j \leq \beta \right\}.
\]

Moreover, for \( k = 1, \ldots, n \), and for all \( \beta \leq u, b \in \mathbb{Z}_+ \), define an upper-bounding value function:

\[
\phi_k(\beta) = \max_{x \in \mathbb{Z}_+^n} \left\{ \sum_{j=1}^{k} d_j x_j : \sum_{j=1}^{k} a_j x_j \leq \beta \right\},
\]

which is equal to the leader’s profit when the follower selects from the first \( k \) items so as to maximize the leader’s profit irrespective of the follower’s own objective. Hence, \( \phi_k(\beta) \) constitutes an upper bound on \( \lambda_n(\beta) \), \( \forall \beta \leq u, b \in \mathbb{Z}_+ \). In the first phase of our solution approach, we compute the follower’s value function \( \psi_n(\beta) \), and the upper-bounding value function \( \phi_k(\beta) \) for all \( k = 1, \ldots, n \) and for all integers \( \beta \leq u \) using a dynamic programming recursion.

We propose a branch-and-backtrack algorithm to evaluate \( \lambda_n(h(\omega, \sum_{i=\bar{b}}^{b} t_i z_i)) \) by solving subproblem (5) over \( \omega \in \Omega \) given a first-stage solution \( z \). We compare the performances of this algorithm and an off-the-shelf IP solver in Section 6.

Algorithm 1. Branch-and-backtrack algorithm (BBA) to evaluate \( \lambda_n(\beta) \).

Step 0: (Initialization) Create a node \( \mathcal{P}^0 \) with \( k_0 = n, \beta_0 = \beta \), and \( y_0 = 0 \). Initialize list \( M \) \leftarrow \{ \mathcal{P}^0 \}. Initialize lower bound \( L = 0 \) and \( \psi_0(\beta) = 0 \) for \( 0 \leq \beta \leq \beta_0 \).

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

Step 2: (Backtracking and pruning) While \( k_m \geq 1 \)

(2a) If \( \psi_m + \phi_{k_m}(\beta_m) \leq L \), go to Step 1.

(2b) If \( \psi_m > \beta_m \), set \( k_m = k_m - 1 \), go to Step 2.

(2c) If \( \psi_m(\beta_m) < \psi_{m-1}(\beta_m - a_m) + c_m \), set \( \eta_m = \psi_m + a_m, \beta_m = \beta_m - a_m, \) and \( k_m = k_m - 1. \) If \( \eta_m > L \), update \( L = \eta_m \), go to Step 2.

Step 3: (Branching) Create two nodes \( \mathcal{P}^{m_1} \) and \( \mathcal{P}^{m_2} \). Set \( k_{m_1} = k_m - 1, \beta_{m_1} = \beta_m - a_m, \) and \( \eta_{m_1} = \eta_m + a_m \), set \( k_{m_2} = k_m - 1, \beta_{m_2} = \beta_m, \) and \( \eta_{m_2} = \eta_m \). Update \( M \leftarrow M \cup \{ \mathcal{P}^{m_1}, \mathcal{P}^{m_2} \} \). Go to Step 1.

If \( M \) is the set of unprocessed nodes, \( m \) is a node index, and \( L \) is the current lower bound on \( \lambda_n(\beta) \). For each node \( \mathcal{P}^m \in M, k_m \) is the number of items that still need to be backtracked, \( \beta_m \) is the amount of remaining capacity, and \( \eta_m \) is the leader’s current benefit from the items selected so far.

Step 2a checks if the current node is promising based on the upper bound obtained from \( \phi_{k_m}(\beta_m) \). If so, Step 2b checks whether selecting item \( k_m \) is feasible based on the remaining capacity \( \beta_m \). If it is feasible, Step 2c compares the benefits gained by selecting and not selecting item \( k_m \). If selecting is more profitable, item \( k_m \) is selected, the remaining capacity as well as the current value of the lower bound on the leader’s optimal objective value, i.e., \( L \), is updated, and backtracking is resumed by considering item \( k_m - 1 \). If not selecting is more profitable, the algorithm continues backtracking by considering item \( k_m - 1 \). If they are equally profitable, Step 3 branches to create two new nodes.

In the backtracking step, the follower selects or rejects the items to maximize its own profit. If selecting or rejecting an item has the same benefit for the follower, then the algorithm should decide in favor of the leader in the optimistic case. This is accomplished by branching and considering both options.

Proposition 3. For all \( \beta \leq u, b \in \mathbb{Z}_+ \), the branch-and-backtrack algorithm returns \( \lambda_n(\beta) \) after a finite number of iterations.

Our first approach, exhaustive search, solves problem (4) by computing \( t_i + \mathbb{E}_\omega \left[ \lambda_n(h(\omega, i)) \right] \) for each \( i = b, \ldots, b \) using either BBA or an IP solver; and picks the value that returns the highest benefit for the leader.

Exhaustive search requires solving subproblem (5) \( \Omega \times (b - \bar{b}) \) times, which can be burdensome if \( \bar{b} \gg b \). Therefore, we also propose a branch-and-cut algorithm to solve problem (4). This algorithm is based on the integer L-shaped method of Laporte and Louveaux [11]. We assume that giving a feasible leader’s decision \( z \), the value of \( \mathbb{E}_\omega \left[ d^T x_\omega(z, z) \right] \) can be computed. Moreover, we assume a finite upper bound \( U \) on \( \mathbb{E}_\omega \left[ d^T x_\omega(z^*) \right] \), where \( z^* \) is the optimal decision of the leader. Section 5 shows one way to derive such a bound.
Our branch-and-cut algorithm considers a current problem at each node of the search tree. The initial current problem is defined as a relaxation of problem (4), which is obtained by replacing \(\mathbb{E}_w[d^f(x(\omega, z))]\) by an upper bound \(\theta\), and removing the integrality restrictions. The current problem is modified by introducing integrality conditions through the branching process, and by generating optimality cuts on \(\theta\) at integer solutions.

Proposition 4 states that when \(t > 0\) and the objective coefficients of the leader and the follower coincide, the leader gives at least as many resources to the follower. However, if \(t \leq 0\), then it might be the case that \(y^* < y^\prime\), as illustrated in Example 1.

Example 1. Consider the following instance of BKPS with two scenarios:

\[
\max f(y, X) = -2y + \sum_{i=1}^2 \left[ 0.5x_1(\omega_i, y) + 6x_2(\omega_i, y) \right] \quad (9a)
\]

subject to \(1 \leq y \leq 2\), \(x(\omega_i, y) \in \mathbb{R}(h(\omega_i, y))\) for \(i = 1, 2\), \(h(\omega_1, y) = 3\); \(y \in \{1, 2\}\), \(h(\omega_2, y) = 4\); and \(y \in \{1, 2\}\). Moreover, for \(i = 1, 2\),

\[
R(h(\omega_i, y)) = \text{argmax} \ 6x_1(\omega_i, y) + 2x_3(\omega_i, y) \quad (10a)
\]

subject to \(5x_1(\omega_i, y) + 3x_2(\omega_i, y) + 2x_3(\omega_i, y) \leq h(\omega_i, y), \quad (10b)
\]

When we formulate and optimize problem (8) over this example, the optimal solution is \(y^* = 1\). However, the optimal solution of the bilevel problem is \(y^\prime = 2\).

Proposition 5. We have that \(f^* \leq f^\prime\).

Proof.

\[
f^* = ty^* + \mathbb{E}_o [\lambda_n(h(\omega, y^*))] \leq ty^* + \mathbb{E}_o [\phi_n(h(\omega, y^*))] \\
\leq ty^* + \mathbb{E}_o [\phi_n(h(\omega, y^*))] = f^\prime,
\]

where the first inequality follows since \(\phi_n(\beta)\) is an upper bound on \(\lambda_n(\beta)\), and the last inequality follows from the optimality of \(y^\prime\) to problem (8).

6. Computational results

6.1. Preliminaries

We generate 16 test instance classes with sizes given in Table 1. Input data for the leader, i.e. \(d\) and \(t\), are generated according to a uniform distribution over the intervals \([1, 1000]\) and \([-1000, -1]\), respectively. We set \(b = 1\) for all test instances. Input data for the follower’s knapsack problem, i.e. \(c\) and \(a\) vectors, are randomly generated with varying degrees of correlation between the weights and profits of items (i.e. uncorrelated, correlated, and highly correlated) using a method proposed by Martello et al. [14] for the 0–1 knapsack problems. Given \(b\) and \(b\), to generate a \([b \times b]\) scenario matrix for \(h(\omega, y)\) that is nondecreasing in \(y\), we run
the generator for all $y = b, \ldots, \bar{b}$, and sort generated capacities in nondecreasing order under each scenario $\omega = \Omega$.

We generate 5 instances from each instance class and report the average time required to solve each of them. When reporting our computational results, we refer to uncorrelated instances as ICx-u, correlated instances as ICx-c, and highly correlated instances as ICx-h. In our computational experiments, we consider 4 different algorithmic combinations:

1. Exhaustive search using BBA to solve subproblems (5).
2. Exhaustive search using CPLEX 11.0 [4] to solve subproblems (5).

In our implementation of the branch-and-backtrack algorithm (Algorithm 1), in Step 1, we select the node $m$ that has the largest $k_m$ value, breaking the ties arbitrarily. For the branch-and-cut algorithm (Algorithm 2), in Step 2, we solve the current problem after recalculating the upper bound $U$ of Proposition 6 based on the local bounds of $z_\omega$ variables. All computational experiments are conducted on an Intel Xeon PC with 3 GHz CPU and 3 GB of RAM.

6.2. Results and discussion

We report the solution times (in seconds) for the uncorrelated, correlated, and highly correlated instances in Tables 2–4, respectively. In the first phase of our solution approach, the time required for calculating $\psi_k(\beta)$ and $\phi_k(\beta)$ for all $k = 1, \ldots, n$, and for all integers $\beta \leq u$ using a dynamic programming recursion is less than a second for all second instances, and included in the reported total solution times. As seen in Tables 2 and 3, BBA is much faster than CPLEX 11.0 for all uncorrelated and correlated instances. For highly correlated instances in Table 4, BBA is still faster than CPLEX 11.0 up to 80 variables. However, when the number of variables exceeds 80, CPLEX 11.0 outperforms BBA. This result is due to the “curse of dimensionality”; as the variable space gets too large for backtracking and BBA fathoms few nodes when the weights and profits are highly correlated.

Tables 2–4 show that branch-and-cut is at least as fast as exhaustive search for all instances. Moreover, it can provide up to 35% speedup over exhaustive search, e.g. the IC16-h instance. We observe that the benefit of using branch-and-cut over exhaustive search is more significant for larger test instances. For example, the average speedup due to using branch-and-cut is 19% for the instances IC1-h through IC7-h, compared to 27% for the instances IC8-h through IC16-h.

A potential drawback of our solution approach is the explicit storage of value functions $\psi_k(\beta)$ and $\phi_k(\beta)$ in computer memory for all $k = 1, \ldots, n$ and for all integers $\beta \leq u$. This might be a limitation for instances that have large values of $(n \times u)$. One approach is to calculate the value function for a given right-hand side whenever it is needed. However, this would increase the solution time of our algorithms since it does not exploit the information gained from former value function calculations.

Finally, the performance of our branch-and-cut algorithm heavily depends on the quality of the upper bound $U$ imposed on the expected recourse function. Hence, an upper bound that is stronger than the one provided in Proposition 8 could drastically improve the performance of our branch-and-cut algorithm.

### Table 1
Size of test instance classes.

|   | n   | b   | $|\Omega|$ |
|---|-----|-----|--------|
| IC1 | 20  | 50  | 100    |
| IC2 | 20  | 50  | 200    |
| IC3 | 20  | 100 | 100    |
| IC4 | 20  | 100 | 200    |
| IC5 | 50  | 50  | 100    |
| IC6 | 50  | 50  | 200    |
| IC7 | 50  | 100 | 100    |
| IC8 | 50  | 100 | 200    |

### Table 2
Solution times of exhaustive search and branch-and-cut for uncorrelated instances (in seconds).

<table>
<thead>
<tr>
<th></th>
<th>Exhaustive search</th>
<th>Branch-and-cut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
<td>CPLEX</td>
</tr>
<tr>
<td>IC1-u</td>
<td>&lt;1</td>
<td>23.6</td>
</tr>
<tr>
<td>IC2-u</td>
<td>&lt;1</td>
<td>48.4</td>
</tr>
<tr>
<td>IC3-u</td>
<td>&lt;1</td>
<td>43.6</td>
</tr>
<tr>
<td>IC4-u</td>
<td>&lt;1</td>
<td>87.8</td>
</tr>
<tr>
<td>IC5-u</td>
<td>&lt;1</td>
<td>34.4</td>
</tr>
<tr>
<td>IC6-u</td>
<td>&lt;1</td>
<td>60.8</td>
</tr>
<tr>
<td>IC7-u</td>
<td>&lt;1</td>
<td>61.6</td>
</tr>
<tr>
<td>IC8-u</td>
<td>&lt;1</td>
<td>134.4</td>
</tr>
<tr>
<td>IC9-u</td>
<td>&lt;1</td>
<td>44.0</td>
</tr>
<tr>
<td>IC10-u</td>
<td>&lt;1</td>
<td>87.0</td>
</tr>
<tr>
<td>IC11-u</td>
<td>&lt;1</td>
<td>84.4</td>
</tr>
<tr>
<td>IC12-u</td>
<td>&lt;1</td>
<td>181.8</td>
</tr>
<tr>
<td>IC13-u</td>
<td>&lt;1</td>
<td>54.6</td>
</tr>
<tr>
<td>IC14-u</td>
<td>&lt;1</td>
<td>104.8</td>
</tr>
<tr>
<td>IC15-u</td>
<td>&lt;1</td>
<td>99.8</td>
</tr>
<tr>
<td>IC16-u</td>
<td>&lt;1</td>
<td>219.2</td>
</tr>
</tbody>
</table>

### Table 3
Solution times of exhaustive search and branch-and-cut for correlated instances (in seconds).

<table>
<thead>
<tr>
<th></th>
<th>Exhaustive search</th>
<th>Branch-and-cut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
<td>CPLEX</td>
</tr>
<tr>
<td>IC1-c</td>
<td>&lt;1</td>
<td>29.4</td>
</tr>
<tr>
<td>IC2-c</td>
<td>&lt;1</td>
<td>64.4</td>
</tr>
<tr>
<td>IC3-c</td>
<td>&lt;1</td>
<td>63.2</td>
</tr>
<tr>
<td>IC4-c</td>
<td>&lt;1</td>
<td>123.8</td>
</tr>
<tr>
<td>IC5-c</td>
<td>&lt;1</td>
<td>67.2</td>
</tr>
<tr>
<td>IC6-c</td>
<td>&lt;1</td>
<td>152.4</td>
</tr>
<tr>
<td>IC7-c</td>
<td>&lt;1</td>
<td>145.8</td>
</tr>
<tr>
<td>IC8-c</td>
<td>&lt;1</td>
<td>299.2</td>
</tr>
<tr>
<td>IC9-c</td>
<td>&lt;1</td>
<td>145.0</td>
</tr>
<tr>
<td>IC10-c</td>
<td>&lt;1</td>
<td>267.0</td>
</tr>
<tr>
<td>IC11-c</td>
<td>&lt;1</td>
<td>256.4</td>
</tr>
<tr>
<td>IC12-c</td>
<td>&lt;1</td>
<td>508.0</td>
</tr>
<tr>
<td>IC13-c</td>
<td>&lt;1</td>
<td>193.4</td>
</tr>
<tr>
<td>IC14-c</td>
<td>&lt;1</td>
<td>333.2</td>
</tr>
<tr>
<td>IC15-c</td>
<td>&lt;1</td>
<td>304.4</td>
</tr>
<tr>
<td>IC16-c</td>
<td>&lt;1</td>
<td>606.6</td>
</tr>
</tbody>
</table>

### Table 4
Solution times of exhaustive search and branch-and-cut for highly correlated instances (in seconds).

<table>
<thead>
<tr>
<th></th>
<th>Exhaustive search</th>
<th>Branch-and-cut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BBA</td>
<td>CPLEX</td>
</tr>
<tr>
<td>IC1-h</td>
<td>&lt;1</td>
<td>67.2</td>
</tr>
<tr>
<td>IC2-h</td>
<td>&lt;1</td>
<td>122.6</td>
</tr>
<tr>
<td>IC3-h</td>
<td>&lt;1</td>
<td>88.6</td>
</tr>
<tr>
<td>IC4-h</td>
<td>&lt;1</td>
<td>186.6</td>
</tr>
<tr>
<td>IC5-h</td>
<td>8.4</td>
<td>198.4</td>
</tr>
<tr>
<td>IC6-h</td>
<td>6.4</td>
<td>358.2</td>
</tr>
<tr>
<td>IC7-h</td>
<td>19.8</td>
<td>364.8</td>
</tr>
<tr>
<td>IC8-h</td>
<td>32.2</td>
<td>711.0</td>
</tr>
<tr>
<td>IC9-h</td>
<td>126.4</td>
<td>225.8</td>
</tr>
<tr>
<td>IC10-h</td>
<td>134.2</td>
<td>451.8</td>
</tr>
<tr>
<td>IC11-h</td>
<td>164.8</td>
<td>424.8</td>
</tr>
<tr>
<td>IC12-h</td>
<td>364.6</td>
<td>976.4</td>
</tr>
<tr>
<td>IC13-h</td>
<td>1257.6</td>
<td>2342</td>
</tr>
<tr>
<td>IC14-h</td>
<td>903.6</td>
<td>526.8</td>
</tr>
<tr>
<td>IC15-h</td>
<td>1208.8</td>
<td>467.0</td>
</tr>
<tr>
<td>IC16-h</td>
<td>7575.4</td>
<td>922.2</td>
</tr>
</tbody>
</table>
Acknowledgements

The authors are partially supported by National Science Foundation Grants CMMI-0826141 and DMI-0620780, AFOSR Grant FA9550-08-1-0268, and a grant from the University of Pittsburgh Central Research Development Fund.

References