Improving the integer L-shaped method

Gustavo Angulo, Shabbir Ahmed, Santanu S. Dey
Georgia Institute of Technology
gangulo@gatech.edu, sahmed@isye.gatech.edu, santanu.dey@isye.gatech.edu

April 29, 2014

Abstract

We consider the integer L-shaped method for two-stage stochastic integer programs. To improve the performance of the algorithm, we present and combine two strategies. First, to avoid time-consuming exact evaluations of the second-stage cost function, we propose a simple modification that alternates between linear and mixed-integer subproblems. Then, to better approximate the shape of the second-stage cost function, we present a general framework to generate optimality cuts via a cut-generating linear program which considers information from all solutions found up to any given stage of the method. In order to address the impact of the proposed approaches, we report computational results on two classes of stochastic integer problems.

1 Introduction

In this work we consider mixed-integer programs of the form

$$
\begin{aligned}
\text{(IP)} \quad \min_{x,z,\theta} \quad & cx + dz + \theta \\
\text{s.t.} \quad & Ax + Cz \le b && (1)\\
& Q(x) - \theta \le 0 && (2)\\
& x \in \{0,1\}^n && (3)\\
& z \ge 0,\ z \in Z, && (4)
\end{aligned}
$$

where Z is a mixed-integer set and $Q(x)$ is a real-valued function taking a binary vector x as argument. We say that (x^*, z^*, θ^*) is a candidate solution if (x^*, z^*) satisfies (1), (3), and (4). If in addition (2) holds, then we say (x^*, z^*, θ^*) is a feasible (candidate) solution. Constraint (2) together with the presence of θ in the objective function ensures $\theta = Q(x)$ is satisfied by any optimal solution to (IP). A fundamental assumption is that given x, $Q(x)$ can be computed with a reasonable amount of effort.

In the context of two-stage stochastic integer programming, we usually have

$$Q(x) := \mathbb{E}_\xi \left[\min_y \{ qy : \ Wy = h - Tx, \ y \in Y \} \right],$$

which denotes the expected second-stage cost of x with respect to the random data $\xi = (q, W, T, h)$. We assume that Y imposes some integrality requirements on y. When ξ has a finite set of possible outcomes, we have $Q(x) = \sum_{\xi} p_{\xi} Q_{\xi}(x)$, where $Q_{\xi}(x)$ denotes the optimal second-stage value of the scenario associated to ξ, and p_{ξ} is the probability of occurrence of ξ. Thus, (IP) can be cast as a large-scale mixed integer program. When the burden of solving (IP) is mainly due to the presence of a large number of scenarios, schemes similar to Benders' decomposition [5] and the L-shaped method [17] can be effective. The idea is to relax (2) and consider θ as an underestimator of $Q(x)$, and successively add cuts in the (x, θ)-space to better approximate the shape of $Q(x)$. This is done until an optimal solution (x^*, z^*, θ^*) satisfying $\theta^* = Q(x^*)$ is found. When the second-stage problem is a linear program, $Q(x)$ is convex in x and thus can be approximated by subgradients using optimal dual solutions. In contrast, when the second-stage problem is a mixed-integer program, such a nice property does not hold, and moreover, $Q(x)$ can even be discontinuous. Thus, decomposition approaches for the linear case have to be modified to accommodate integer variables in the second stage. In [9], such a modification, the integer L-shaped method, is introduced. It is designed for two-stage stochastic integer problems having binary first-stage variables as it exploits the facial property of 0-1 sets. More generally, the integer L-shaped method can be applied to any mixed-integer problem having the form of (IP) as long as $Q(x)$ is computable from binary x. In particular, it also fits situations where $Q(x)$ can be evaluated with a closed-form analytical formula, but it does not have an amenable mixed-integer formulation. Applications of this method include vehicle routing [12], [7], probabilistic traveling salesman problems [10], location problems [11], and generalized assignment [2], among others.

Next we describe the integer L-shaped method. Let X be the projection of the feasible region of (IP) onto the x-space, and let $L \in \mathbb{R}$ be a lower bound on $Q(x)$ over X. Then (IP) can be equivalently formulated as

$$\begin{aligned}
\text{(MP)} \quad \min \quad & cx + dz + \theta \\
\text{s.t.} \quad & Ax + Cz \le b \\
& \Pi x - \mathbf{1}\theta \le \pi_0 \qquad (5)\\
& x \in \{0,1\}^n \\
& z \ge 0,\ z \in Z \\
& \theta \ge L,
\end{aligned}$$

where $\mathbf{1}$ denotes a vector of ones of appropriate size, as long as for each $x^* \in X$ constraints (5) include a cut of the form $\pi^k x - \theta \le \pi_0^k$ such that $\pi^k x - \pi_0^k \le Q(x)$ for all $x \in X$ and $\pi^k x^* - \pi_0^k = Q(x^*)$. In other words, the affine function $\pi^k x - \pi_0^k$ underestimates $Q(x)$ on X, and the estimate is tight at x^*. The optimality cuts of Laporte and Louveaux [9] define such a cut family and form the basis of the integer L-shaped method.

Given $x^* \in \{0,1\}^n$, let $S(x^*) := \{i : \ x_i^* = 1\}$. In [9], the (standard) integer optimality cut at x^* is defined as

$$\theta \ge (Q(x^*) - L)\left(\sum_{i \in S(x^*)} x_i - \sum_{i \notin S(x^*)} x_i - |S(x^*)|\right) + Q(x^*). \qquad (6)$$

Given the enumerative nature of (6), in practice, these cuts are complemented with other inequalities that, albeit may not be tight, help to improve the global lower bound on $Q(x)$. When $Q(x)$ is the expected second-stage value of x given by the value function of a mixed-integer program, the most obvious inequalities to add are the subgradient cuts given by the continuous relaxation $Q_{LP}(x)$ of $Q(x)$. They have the form

$$\theta \ge s(x - x^*) + Q_{LP}(x^*), \qquad (7)$$

where s is a subgradient of $Q_{LP}(x)$ at x^*.

A typical implementation of the integer L-shaped method with a current state-of-the-art solver works as follows. Having computed a lower bound L on $Q(x)$ and solved the continuous relaxation of (IP) with Benders' decomposition, we end up with a linear master problem that includes subgradient cuts of the form (7). Then we reinforce the binary requirements on x and any integrality restrictions on z, leading to a mixed-integer master problem of the form (MP), but where the system (5) is a relaxation of (IP), so that an optimal solution to the current problem may not be feasible to (IP). The idea now is to solve the mixed-integer master problem in a way such that all integer solutions are checked against feasibility with respect to (IP) before being accepted as an incumbent. For this, the solver proceeds in a similar fashion to branch-and-cut, that is, it generates a search tree by solving linear subproblems, branching, and adding cutting planes. The main difference is that when a candidate integer solution (x^*, z^*, θ^*) satisfying (1), (3) and (4) is found at a node of the search tree, an additional routine, the so-called optimality cut function, is called in order to assert feasibility and add optimality cuts. If the solution is infeasible to the true problem (IP), i.e., $\theta^* < Q(x^*)$, this function generates an optimality cut that is applied to all pending nodes in the master problem tree, ensuring that this solution is discarded. Then the solver continues exploring the tree with the guarantee that any discarded, and thus infeasible, solution will not appear again. If the solution is actually feasible to (IP), then it is accepted by the optimality cut function and the current incumbent is updated accordingly. This description of the (standard) integer L-shaped method is summarized in Algorithm 1 below.

Algorithm 1 Integer L-shaped method

```
Input: $A, C, b, c, d, Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}$
Output: Optimal solution $x^*$ to (IP) and optimal value
1: Compute a lower bound $L$ of $Q(x)$
2: Solve the LP relaxation of (IP) with Benders' decomposition
3: Declare $x$ variables as binary in master problem
4: Initialize the optimality cut function
5: Solve the integer master problem using the optimality cut function to assert feasibility of solutions
   and add optimality cuts
6: return $x^*$ and optimal value
```

In line 4 of Algorithm 1 we initialize any additional structures that may be needed by the optimality cut function before invoking the solver in line 5. In particular, as there may be several solutions sharing the same x subvector, we keep a list V of first-stage x for which $Q(x)$ has been computed to avoid duplicate evaluations. In a standard implementation, the optimality cut function has the form shown in Algorithm 2

Algorithm 2 Standard optimality cut function

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}, V$
Output: true if solution is feasible, false otherwise
1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
2:     return true
3: end if
4: Compute $Q_{LP}(x^*)$
5: Add the subgradient cut (7)
6: Compute $Q(x^*)$
7: $V \leftarrow V \cup \{x^*\}$
8: if $\theta^* < Q(x^*)$ then
9:     Add the integer optimality cut (6)
10:    return false
11: else
12:    return true
13: end if
```

The optimality cut function returns `true` if the candidate integer solution is indeed feasible to (IP).

Otherwise it returns `false` to reject the solution and apply the optimality cut. Note that the steps in lines 4 and 5 in Algorithm 2 are not needed for convergence of the method, but help to improve the global lower bound on $Q(x)$.

The optimality cut (6) relies on exact evaluations of $Q(x)$, which can be very time-consuming in the case where $Q(x)$ is given by a complicated mixed-integer program. Also, observe that (6) depends on x^* and $Q(x^*)$ only, i.e., it only depends on the point to be cut-off. In particular, it does not take into account the information provided by other solutions that we may have found while exploring the first-stage set. To improve the performance of the integer L-shaped method, we propose two approaches to deal with the above issues. First, in Section 2, we present a simple modification that alternates between exact and approximate evaluations of $Q(x)$. Then, in Section 3, we introduce of a new class of optimality cuts that includes information obtained from different solutions; in particular, evaluations and estimates of $Q(x)$ at different points. These new cuts are obtained through a cut-generating linear program which is constructed based on ideas from disjunctive programming and the forbidden-vertices problem [3]. Then, in Section 4, we outline an implementation that combines both modifications in a single method. Finally, in Section 5, we present computational results of the proposed variants on two classes of stochastic integer programs.

2 Alternating cuts

In this section we present a simple cut strategy to decrease the overall effort incurred in computing the function $Q(x)$.

Suppose that while solving (IP) with the integer L-shaped method, a candidate solution (x^*, z^*, θ^*) has been found along the search tree of (MP). Recall that we reject the solution if $\theta^* < Q(x^*)$. Since $Q_{LP}(x) \leq Q(x)$, a sufficient condition to reject (x^*, z^*, θ^*) is $\theta^* < Q_{LP}(x^*)$. Given that $Q_{LP}(x)$ is convex, we have that the subgradient cut (7) is a valid inequality that cuts off the pair (x^*, θ^*) in the (x, θ)-space. Therefore, instead of evaluating $Q(x^*)$ exactly, we first evaluate $Q_{LP}(x^*)$ and check whether $\theta^* < Q_{LP}(x^*)$. If so, we add the subgradient cut (7) to remove (x^*, θ^*). Otherwise, we compute $Q(x^*)$ and check whether $\theta^* < Q(x^*)$. If so, we add the integer optimality cut (6). Otherwise, we accept the solution. The key idea is to use $Q_{LP}(x)$ as a proxy for $Q(x)$ to check feasibility of a candidate solution, preventing unnecessary, and more costly, computations of $Q(x)$.

The modification just described is similar in spirit to sequential approximation schemes such as [16], [6], [8], and [15], where the second-stage cost function $Q(x)$ is approximated by linear programs which, starting with $Q_{LP}(x)$, are iteratively strengthened with additional cuts. Although these methods are shown to converge after a finite number of steps, the convergence can be very slow and in practice exact evaluations of $Q(x)$ may be required. In contrast, in the context of the integer L-shaped method, we propose to use $Q_{LP}(x)$ as the unique intermediate approximation for $Q(x)$, which is a simple yet useful modification whose implementation is rather straightforward and, to the best of our knowledge, has not been reported in the literature.

To implement the approach presented above, in addition to the set V of visited first-stage solutions x for which $Q(x)$ is known, we also keep a list V_{LP} of solutions for which the continuous relaxation $Q_{LP}(x)$ has been computed. The modified strategy, which we call alternating cuts, proceeds as shown in Algorithm 3.

Note that if $x^* \notin V_{LP}$ satisfies (7), then x^* is included into V_{LP} and thus the steps in lines 12–19 of Algorithm 3 are applied to check whether (x^*, z^*, θ^*) is a feasible solution or not. As we shall see in Section 5, this simple modification yields speedups of one order of magnitude on instances from the literature.

Algorithm 3 Optimality cut function with alternating cut strategy

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}$, $Q_{LP} : X \to \mathbb{R}$, $V$, $V_{LP}$
Output: true if solution is feasible, false otherwise
1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
2:     return true
3: end if
4: if $x^* \notin V_{LP}$ then
5:     Compute $Q_{LP}(x^*)$
6:     $V_{LP} \leftarrow V_{LP} \cup \{x^*\}$
7:     if $\theta^* < Q_{LP}(x^*)$ then
8:         Add the subgradient cut (7).
9:         return false
10:    end if
11: end if
    // Now we have $x^* \in V_{LP}$ and $\theta^* \geq Q_{LP}(x^*)$
12: Compute $Q(x^*)$.
13: $V \leftarrow V \cup \{x^*\}$
14: if $\theta^* < Q(x^*)$ then
15:     Add the integer optimality cut (6)
16:     return false
17: else
18:     return true
19: end if
```

3 New optimality cuts

In this section, we present a new class of integer optimality cuts that can be used as an alternative to the standard cut (6). After providing an overview of the approach, we show how to construct a cut-generating linear program to separate these new inequalities and then we discuss some implementation details. In this section we denote $\text{conv}(T)$ the convex hull of a set T of real vectors.

Let S be the projection of the feasible set of (MP) onto the (x, θ)-space, which corresponds to the epigraph of $Q(x)$ over X, i.e.,

$$S = \{(x, \theta) \in X \times \mathbb{R} : \ \theta \geq Q(x)\} .$$

Let $V \subseteq X$ be such that $Q(x)$ is known for all $x \in V$. We have

$$S \subseteq S(X, V) := \bigcup_{x \in V} \{(x, \theta) : \ \theta \geq Q(x)\} \cup (X \setminus V) \times \{\theta : \ \theta \geq L\}.$$

In some sense, $S(X, V)$ is the best approximation of S when only the values of $Q(x)$ for $x \in V$ are known and only the trivial lower bound L is available over $X \setminus V$. We consider the relaxation $S(V)$ of $S(X, V)$ given by

$$S(X, V) \subseteq S(V) := \bigcup_{x \in V} \{(x, \theta) : \ \theta \geq Q(x)\} \cup (\{0, 1\}^n \setminus V) \times \{\theta : \ \theta \geq L\}.$$

Figure 1 illustrates an example with $x \in \{0, 1\}^2$ and $V = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$. The bold dots represent the values of $Q(x)$ that are known depending on whether x belongs to V or not. Then $S(V)$ is given by the union of the vertical rays.

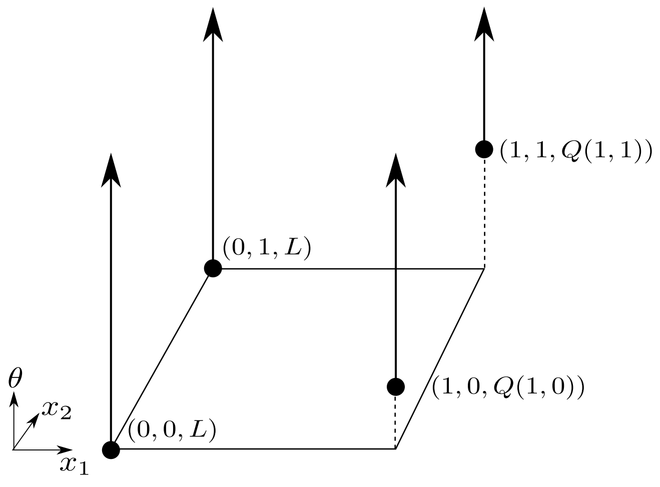


Figure 1: Illustration of $S(V)$.

Observe that $S(V) \subseteq S(U)$ for any $U \subseteq V$, and in particular, $S(V) \subseteq S(\{x\})$ for $x \in V$. Moreover, $S(V) = \bigcap_{x\in V} S(\{x\})$. Since (6) is a valid inequality for $\text{conv}(S(\{x\}))$, it is also valid for $\text{conv}(S(V))$. In fact, (6) is the only nontrivial cut needed to describe $\text{conv}(S(\{x\}))$. However, in general, $\text{conv}(S(V)) \subseteq \bigcap_{x\in V} \text{conv}(S(\{x\}))$ holds with strict containment, i.e., adding (6) for all $x \in V$ does not yield $\text{conv}(S(V))$. Our goal is to derive a compact extended formulation for $\text{conv}(S(V))$ and use it to generate optimality cuts for a point (x^*, θ^*) in the (x, θ)-space that take into account the values of $Q(x)$ for $x \in V$.

Continuing the example given by Figure 1 for $x \in \{0,1\}^2$ and $V = \left\{ \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right), \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix} \right) \right\}$, Figure 2 shows how additional information can improve our approximation of the convex hull of the epigraph of $Q(x)$. Figure 2a compares the standard optimality cut at $(1,1)$ with an improved cut which also takes $Q(1,0)$ into account. Figure 2b shows that adding standard optimality cuts at $(1,0)$ and $(1,1)$ does not yield $\text{conv}(S(V))$. In particular, it does not yield a tight formulation for $\{0,1\}^n \setminus V$ in the x-space.

Several steps of the construction of our cut-generating linear program rely on Lemma 1 below, which follows from disjunctive programming [4] applied in the context of linear extended formulations of polyhedra.

Lemma 1. *Let $P_1, \dots, P_k$ be nonempty polyhedra in $\mathbb{R}^n$ having the same recession cone. If $P_i = \{x \in \mathbb{R}^n | \exists y_i \in \mathbb{R}^{m_i} : E_i x + F_i y_i \geq h_i\}$, then $\text{conv}\left(\cup_{i=1}^k P_i\right) = \{x \in \mathbb{R}^n | \exists x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^{m_i}, \lambda \in \mathbb{R}^k : x = \sum_{i=1}^k x_i, E_i x_i + F_i y_i \geq \lambda_i h_i, \sum_{i=1}^k \lambda_i = 1, \lambda \geq 0\}$.*

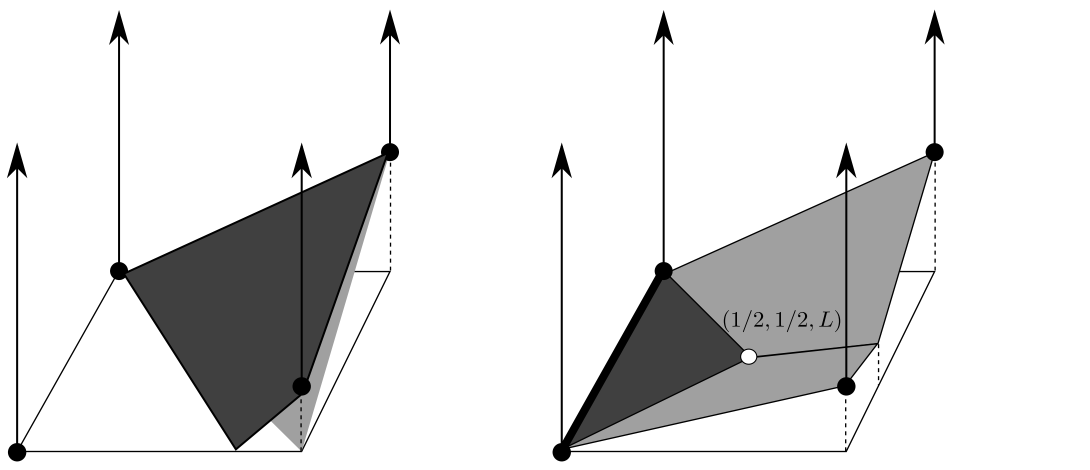
3.1 Construction of CGLP

Clearly, we have $\text{conv}(S(V)) = \text{conv}\big(P_Q(V) \cup P_L(V)\big)$, where

$$P_Q(V) := \text{conv}\left(\bigcup_{x\in V} \{(x,\theta) : \theta \geq Q(x)\} \right)$$

and

$$P_L(V) := \text{conv}(\{0,1\}^n \setminus V) \times \{\theta : \theta \geq L\}.$$



(a) The standard cut at $x = (1,1)$ (light gray) can be strengthened if $Q(1,0)$ is known (dark gray).

(b) The approximation of the remaining solutions (dark gray) given by standard cuts at $(1,0)$ and $(1,1)$ (light gray) can be made exact (bold segment).

Figure 2: Improving the description of $\text{conv}(S(V))$.

Thus to describe $\text{conv}(S(V))$ it suffices to provide compact extended formulations for $P_Q(V)$ and $P_L(V)$ and then apply disjunctive programming to their union as illustrated in Figure 3.

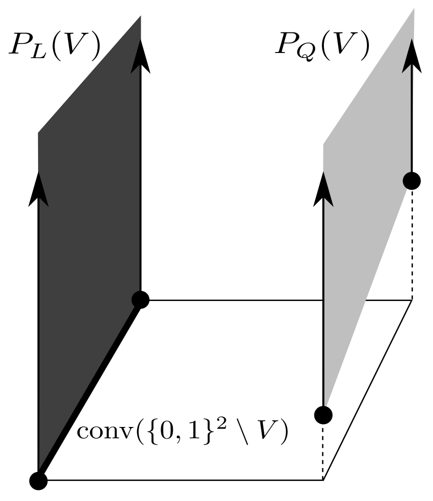


Figure 3: $\text{conv}(S(V)) = \text{conv}\big(P_Q(V) \cup P_L(V)\big)$.

$P_Q(V)$ can be described as follows. Let $V = \{x^1, \ldots, x^t\}$. Then $P_Q(V)$ is the set of vectors $(x^Q, \theta^Q) \in$

$\mathbb{R}^n \times \mathbb{R}$ for which there exists $\phi \in \mathbb{R}^t$ satisfying

$$\begin{aligned}
&-x^Q + \sum_{s=1}^{t} \phi_s x^s = 0\\
&-\theta^Q + \sum_{s=1}^{t} \phi_s Q(x^s) \leq 0\\
&\sum_{s=1}^{t} \phi_s = 1\\
&\phi \geq 0.
\end{aligned}$$

To describe $P_L(V)$, it is enough to describe $\operatorname{conv}(\{0,1\}^n \setminus V)$ and then take its Cartesian product with $\{\theta : \theta \geq L\}$. We build on results from the forbidden-vertices problem [3] to do this.

Let V^i be the projection of V onto the first i coordinates. Define $\hat{V}^1 := \{0,1\} \setminus V^1$, $\hat{V}^i := [V^{i-1} \times \{0,1\}] \setminus V^i \subseteq \{0,1\}^i$ for $i \geq 2$, and write $\hat{V}^i = \{v^i_1, \dots, v^i_{k_i}\}$. Finally, for all i, let $W^{ij} := \hat{V}^i \times \{0\}^{j-i} = \{w^{ij}_1, \dots, w^{ij}_{k_i}\} \subseteq \{0,1\}^j$ for all $j \geq i$ and define $W^i := W^{in} = \{w^i_1, \dots, w^i_{k_i}\} \subseteq \{0,1\}^n$.

From [3], for all $1 \leq j \leq n-1$ we have

$$\{0,1\}^{j+1} \setminus V^{j+1} = \Big[\Big(\{0,1\}^j \setminus V^j\Big) \times \{0,1\}\Big] \cup \hat{V}^{j+1}. \tag{8}$$

The idea behind (8) is that any vector in $\{0,1\}^{j+1} \setminus V^{j+1}$ is such that either its projection onto $\{0,1\}^j$ does not lie in V^j or is obtained by flipping the value of the last component of a vector in V^{j+1}.

Example 2. *Consider $n = 3$ and $V = \left\{ \begin{pmatrix} 0\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\1 \end{pmatrix}, \begin{pmatrix} 1\\0\\1 \end{pmatrix} \right\}$. For $j = 2$, we have $V^2 = \left\{ \begin{pmatrix} 0\\1 \end{pmatrix}, \begin{pmatrix} 1\\0 \end{pmatrix} \right\}$ and therefore $\{0,1\}^2 \setminus V^2 = \left\{ \begin{pmatrix} 0\\0 \end{pmatrix}, \begin{pmatrix} 1\\1 \end{pmatrix} \right\}$. Clearly, any vector in $\{0,1\}^3$ whose projection onto $\{0,1\}^2$ lies outside V^2 must belong to $\{0,1\}^3 \setminus V$. Hence $[\{0,1\}^2 \setminus V^2] \times \{0,1\} = \left\{ \begin{pmatrix} 0\\0\\0 \end{pmatrix}, \begin{pmatrix} 0\\0\\1 \end{pmatrix}, \begin{pmatrix} 1\\1\\0 \end{pmatrix}, \begin{pmatrix} 1\\1\\1 \end{pmatrix} \right\} \subseteq \{0,1\}^3 \setminus V$. On the other hand, we have $V \subseteq V^2 \times \{0,1\} = \left\{ \begin{pmatrix} 0\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\1 \end{pmatrix}, \begin{pmatrix} 1\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\1 \end{pmatrix} \right\}$, and thus $\hat{V}^3 = [V^2 \times \{0,1\}] \setminus V = \left\{ \begin{pmatrix} 1\\0\\0 \end{pmatrix} \right\} \subseteq \{0,1\}^3 \setminus V$. Then we can verify that (8) holds for $j = 2$, i.e., $\{0,1\}^3 \setminus V = \big[\big(\{0,1\}^2 \setminus V^2\big) \times \{0,1\}\big] \cup \hat{V}^3$.*

We use the recursion (8) to derive an explicit extended formulation for $\operatorname{conv}(\{0,1\}^n \setminus V)$ having $\mathcal{O}(n|V|)$ variables and constraints.

Proposition 3. *For all $2 \leq j \leq n$, $\operatorname{conv}\big(\{0,1\}^j \setminus V^j\big)$ is given by all $x \in \mathbb{R}^j$ for which there exist vectors y, λ,* *and μ satisfying*

$$\begin{aligned}
&-x + y + \sum_{i=1}^{j}\sum_{l=1}^{k_i} \mu_l^i w_l^{ij} = 0\\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0\\
&\sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \le i \le j-1\\
&\sum_{l=1}^{k_j} \mu_l^j + \lambda_{j-1} = 1\\
&y_1 = 0\\
&y_i - \lambda_{i-1} \le 0 \quad \forall 2 \le i \le j\\
&y \ge 0,\ \lambda \ge 0,\ \mu \ge 0.
\end{aligned}$$

Proof. We apply induction on $2 \le j \le n$. The base case reduces to proving that $\operatorname{conv}\big(\{0,1\}^2 \setminus V^2\big)$ is given by

$$\begin{aligned}
&-x + y + \sum_{i=1}^{2}\sum_{l=1}^{k_i} \mu_l^i w_l^{i2} = 0\\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0\\
&\sum_{l=1}^{k_2} \mu_l^2 + \lambda_1 = 1\\
&y_1 = 0\\
&y_2 - \lambda_1 \le 0\\
&y \ge 0,\ \lambda \ge 0,\ \mu \ge 0.
\end{aligned} \tag{9}$$

Indeed, from (8), we have

$$\{0,1\}^2 \setminus V^2 = \Big[\Big(\{0,1\}^1 \setminus V^1\Big) \times \{0,1\}\Big] \cup \hat{V}^2. \tag{10}$$

By definition, we have $W^{12} = \hat{V}^1 \times \{0\} = (\{0,1\} \setminus V^1) \times \{0\}$. Then observe that

$$\Big(\{0,1\}^1 \setminus V^1\Big) \times \{0,1\} = W^{12} + \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\},$$

and thus

$$\operatorname{conv}\Big(\Big(\{0,1\}^1 \setminus V^1\Big) \times \{0,1\}\Big) = \operatorname{conv}\Big(W^{12}\Big) + \Big\{ y \in \mathbb{R}^2 :\ y_1 = 0,\ 0 \le y_2 \le 1 \Big\}.$$

Writing $W^{12} = \{w_1^{12}, \dots, w_{k_1}^{12}\}$, it follows that $\operatorname{conv}\big((\{0,1\}^1 \setminus V^1) \times \{0,1\}\big)$ is given by $p \in \mathbb{R}^2$ such that

$$\begin{aligned}
&-p + y + \sum_{l=1}^{k_1} \mu_l^1 w_l^{12} = 0 \\
&\sum_{l=1}^{k_1} \mu_l^1 = 1 \\
&y_1 = 0 \\
&y_2 \leq 1 \\
&y \geq 0,\ \mu^1 \geq 0.
\end{aligned}$$

We also have by definition $\hat{V}^2 = W^{22} = \{w_1^{22}, \dots, w_{k_2}^{22}\}$, and thus $\operatorname{conv}(\hat{V}^2)$ is given by $q \in \mathbb{R}^2$ such that

$$\begin{aligned}
&-q + \sum_{l=1}^{k_2} \mu_l^2 w_l^{22} = 0 \\
&\sum_{l=1}^{k_2} \mu_l^2 = 1 \\
&\mu^2 \geq 0.
\end{aligned}$$

From (10), we apply Lemma 1 to the above polytopes: we introduce a multiplier $0 \leq \lambda_1 \leq 1$, we include the equation $x = p + q$, and we multiply the right-hand-side vectors of the first and second systems by λ_1 and $1 - \lambda_1$, respectively. After eliminating p and q, we immediately obtain the desired system (9) for $\operatorname{conv}(\{0,1\}^2 \setminus V^2)$.

Now, assuming that the claim holds for some $2 \leq j \leq n-1$, we will prove that it also holds for $j+1$. Since $\operatorname{conv}((\{0,1\}^j \setminus V^j) \times \{0,1\}) = \operatorname{conv}((\{0,1\}^j \setminus V^j)) \times [0,1]$, by the inductive hypothesis, we have that $\operatorname{conv}((\{0,1\}^j \setminus V^j) \times \{0,1\})$ is given by $p \in \mathbb{R}^{j+1}$ satisfying

$$\begin{aligned}
&-p + y + \sum_{i=1}^{j} \sum_{l=1}^{k_i} \mu_l^i w_l^{ij+1} = 0 \\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
&\sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \leq i \leq j-1 \\
&\sum_{l=1}^{k_j} \mu_l^j + \lambda_{j-1} = 1 \\
&y_1 = 0 \\
&y_i - \lambda_{i-1} \leq 0 \quad \forall 2 \leq i \leq j \\
&y_{j+1} \leq 1 \\
&y \geq 0,\ \lambda \geq 0,\ \mu \geq 0,
\end{aligned}$$

where we have appended a new variable $0 \leq y_{j+1} \leq 1$ and vectors w_l^{ij} have been extended to w_l^{ij+1} by appending another component with value 0.

We also have that $\operatorname{conv}(\hat{V}^{j+1})$ is given by $q \in \mathbb{R}^{j+1}$ satisfying

$$\begin{aligned}
&-q + \sum_{l=1}^{k_{j+1}} \mu_l^{j+1} w_l^{j+1 j+1} = 0 \\
&\sum_{l=1}^{k_{j+1}} \mu_l^{j+1} = 1 \\
&\mu^{j+1} \geq 0.
\end{aligned}$$

From (8), it is enough to apply Lemma 1 to the above polytopes to find an extended formulation for $\operatorname{conv}(\{0,1\}^{j+1} \setminus V^{j+1})$. Analogously to the base case, we introduce a multiplier $0 \leq \lambda_j \leq 1$, we include the equation $x = p + q$, and we multiply the right-hand-side vectors of the first and second systems by λ_j and $1 - \lambda_j$, respectively. After eliminating p and q, we immediately obtain the desired system for $\operatorname{conv}(\{0,1\}^{j+1} \setminus V^{j+1})$. □

From Proposition 3, we obtain that $\operatorname{conv}(\{0,1\}^n \setminus V)$ is given by the vectors $x^L \in \mathbb{R}^n$ such that

$$\begin{aligned}
&-x^L + y + \sum_{i=1}^{n} \sum_{l=1}^{k_i} \mu_l^i w_l^i = 0 \\
&\sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
&\sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \leq i \leq n-1 \\
&\sum_{l=1}^{k_n} \mu_l^n + \lambda_{n-1} = 1 \\
&y_1 = 0 \\
&y_i - \lambda_{i-1} \leq 0 \quad \forall 2 \leq i \leq n \\
&y \geq 0,\ \lambda \geq 0,\ \mu \geq 0.
\end{aligned}$$

Appending the variable θ^L and the constraint $\theta^L \geq L$ to the above system gives an extended formulation for $P_L(V)$. Note that excluding the nonnegativity restrictions, the constraint matrix has $3n$ rows and $3n + \mathcal{O}(n|V|)$ columns, i.e, only its width changes with V. In particular, updating the formulation can be done columnwise, which is a desirable property from the computational point of view.

Once again, we apply disjunctive programming, but this time to $P_L(V)$ and $P_Q(V)$ to derive an extended formulation for $\operatorname{conv}(S(V))$. Note that both $P_L(V)$ and $P_Q(V)$ have $\{(0,\theta) \in \mathbb{R}^n \times \mathbb{R} : \theta \geq 0\}$ as their recession cone and thus Lemma 1 applies. We introduce a multiplier $0 \leq \delta \leq 1$, we include the equations $x = x^L + x^Q$ and $\theta = \theta^L + \theta^Q$, and we multiply the right-hand-side vectors of the systems defining $P_L(V)$ and $P_Q(V)$ by δ and $1 - \delta$, respectively.

Recall that in the definition of $S(V)$ we have dropped the dependence on X. To recover part of that information, we can describe a polyhedron that lies between $\operatorname{conv}(S)$ and $\operatorname{conv}(S(V))$. For that, $P_L(V)$ can be coupled with any valid inequality for (MP). In particular, including variables $z \geq 0$ and the system $Ax^L + Cz \leq b$ tightens the formulation. Lower bounds of the form $\Pi x^L - \mathbf{1}\theta^L \leq \pi_0$ can be useful too to better approximate the shape of the epigraph S of $Q(x)$. Thus we may assume that both types of constraints are added to the formulation of $P_L(V)$, and that $\theta^L \geq L$ is absorbed in $\Pi x^L - \mathbf{1}\theta^L \leq \pi_0$.

Finally, we obtain that if (x^*, θ^*) does not belong to $\text{conv}(S(V))$, and thus not to $\text{conv}(S)$, then the following system is infeasible:

$$\begin{array}{cl}
(\alpha) & x^L + x^Q = x^* \\
(\beta) & \theta^L + \theta^Q = \theta^* \\
(\sigma) & -x^L + y + \sum_{i=1}^{n} \sum_{l=1}^{k_i} \mu_l^i w_l^i = 0 \\
(\rho_1) & \sum_{l=1}^{k_1} \mu_l^1 - \lambda_1 = 0 \\
(\rho_i) & \sum_{l=1}^{k_i} \mu_l^i + \lambda_{i-1} - \lambda_i = 0 \quad \forall 2 \leq i \leq n-1 \\
(\rho_n) & \sum_{l=1}^{k_n} \mu_l^n + \lambda_{n-1} - \delta = 0 \\
(\varphi_1) & y_1 = 0 \\
(\varphi_i) & y_i - \lambda_{i-1} \leq 0 \quad \forall 2 \leq i \leq n \\
(\psi) & \Pi x^L - \mathbf{1}\theta^L - \pi_0 \delta \leq 0 \\
(\nu) & Ax^L + Cz - b\delta \leq 0 \\
(\gamma) & -x^Q + \sum_{s=1}^{t} \phi_s x^s = 0 \\
(\tau) & -\theta^Q + \sum_{s=1}^{t} \phi_s Q(x_s) \leq 0 \\
(\eta) & \sum_{s=1}^{t} \phi_s + \delta = 1 \\
 & y \geq 0,\ \lambda \geq 0,\ \mu \geq 0 \\
 & \phi \geq 0 \\
 & \delta \geq 0.
\end{array}$$

By Farkas' Lemma, and after removing redundancies, we arrive at the alternative system

$$\begin{array}{rcll}
x^*\alpha + \theta^*\beta + \eta & < & 0 & \\
\alpha - \sigma + A^\top \nu + \Pi^\top \psi & = & 0 & \\
\beta - \mathbf{1}\psi & = & 0 & \\
-\rho_n + \eta - b\nu - \pi_0\psi & \geq & 0 & \\
C^\top \nu & \geq & 0 & \\
\sigma_i + \varphi_i & \geq & 0 & 2 \leq i \leq n \\
-\rho_i + \rho_{i+1} + \varphi_{i+1} & \geq & 0 & 1 \leq i \leq n-1 \\
w_l^i \sigma + \rho_i & \geq & 0 & 1 \leq n,\ 1 \leq l \leq k_i \\
x^s\alpha + Q(x^s)\beta + \eta & \geq & 0 & 1 \leq s \leq t \\
\beta \geq 0,\ \varphi \geq 0,\ \nu \geq 0,\ \psi \geq 0. & & &
\end{array}$$

Thus, any feasible solution to the above system yields an inequality $\alpha x + \beta\theta \geq -\eta$ that is valid for $\text{conv}(S)$, but is violated by (x^*, θ^*).

For finite termination of the integer L-shaped method, we need a tightness condition at the current solution, namely $\alpha x^* + \beta Q(x^*) = -\eta$. Including this condition yields $0 > x^*\alpha + \theta^*\beta + \eta = x^*\alpha + \beta Q(x^*) + \eta - \beta Q(x^*) + \theta^*\beta = \beta(\theta^* - Q(x^*))$. Since $\theta^* < Q(x^*)$, we conclude that $\beta > 0$ in any feasible tight solution. Therefore, we replace the condition $x^*\alpha + \theta^*\beta + \eta < 0$ with $x^*\alpha + Q(x^*)\beta + \eta = 0$ and the normalization $\beta = 1$. Note that the objective function of the resulting linear program is fixed to zero, and we only need to find a feasible solution, which always exists by definition of the system; in particular, (6) is feasible. We arrive at the following result.

Proposition 4. *Let $(x^*, \theta^*) \in \{0,1\}^n \times \mathbb{R}$ be such that $\theta^* < Q(x^*)$. Set $x^t := x^*$ and let $V = \{x^1, \ldots, x^t\}$ be the set of solutions for which $Q(x)$ is known. Consider the cut-generating linear program CGLP below*

$$\begin{array}{rcll}
\alpha - \sigma + A^\top \nu + \Pi^\top \psi & = & 0 & \\
\mathbf{1}\psi & = & 1 & \\
-\rho_n + \eta - b\nu - \pi_0\psi & \geq & 0 & \\
C^\top \nu & \geq & 0 & \\
\sigma_i + \varphi_i & \geq & 0 & 2 \leq i \leq n \\
-\rho_i + \rho_{i+1} + \varphi_{i+1} & \geq & 0 & 1 \leq i \leq n-1 \\
w_l^i \sigma + \rho_i & \geq & 0 & 1 \leq i \leq n,\ 1 \leq l \leq k_i \qquad (11) \\
x^s\alpha + \eta & \geq & -Q(x^s) & 1 \leq s < t \qquad (12) \\
x^t\alpha + \eta & = & -Q(x^t) & \qquad (13) \\
\varphi \geq 0,\ \nu \geq 0,\ \psi \geq 0, & & &
\end{array}$$

where

1. *(A, C, b) defines the system (1),*
2. *(Π, π_0) defines a collection $\Pi x - \mathbf{1}\theta \leq \pi_0$ of affine lower bounds that includes $\theta \geq L$, and*
3. *the vectors w_l^i are given by the definition of W^i in Section 3.1.*

Then any feasible solution to CGLP yields an optimality cut of the form

$$\alpha x + \theta \geq -\eta \qquad (14)$$

that is tight at x^* and thus is violated by (x^*, θ^*).

3.2 Implementation

The main difference that we are proposing with the standard implementation is the use of the CGLP-based cut (14) in place of (6). This requires keeping a list V of first-stage solutions for which $Q(x)$ has been computed and updating CGLP accordingly. Algorithm 4 shows the procedure.

A key step is found in line 7 of Algorithm 4 as $\text{conv}(S(V))$ has to be recomputed whenever a new vector x^* is added to V. Of course, we could derive CGLP from scratch every time. Doing so requires computing the sets W^i and thus creating $\mathcal{O}(n|V|)$ constraints in (11). Instead, we propose to perform marginal updates from an iteration to the next one using the fact that $W^i = \hat{V}^i \times \{0\}^{n-i}$.

Let $V_t = \{x^1, \ldots, x^t\}$ be the set of the first t solutions found along the master tree. Similarly, let V_t^i be the projection of V_t onto the first i components and set $\hat{V}_t^i := [V_t^{i-1} \times \{0,1\}] \setminus V_t^i$ with $\hat{V}_t^1 := \{0,1\} \setminus V_t^1$.

Algorithm 4 Optimality cut function with CGLP-based optimality cuts

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}$, $Q_{LP} : X \to \mathbb{R}$, $V$
Output: true if solution is feasible, false otherwise
 1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
 2:     return true
 3: end if
 4: Compute $Q_{LP}(x^*)$
 5: Add the subgradient cut (7)
 6: Compute $Q(x^*)$
 7: Update CGLP.
 8: $V \leftarrow V \cup \{x^*\}$
 9: if $\theta^* < Q(x^*)$ then
10:     Solve CGLP to obtain $\alpha$ and $\eta$
11:     Add the integer optimality cut (14)
12:     return false
13: else
14:     return true
15: end if
```

Suppose a new vector $x^{t+1} = (x_1, \ldots, x_n)$ is to be included and let V_{t+1}, V^i_{t+1}, $\hat{V}^i_{t+1}$ be the updated sets. Let $\bar{x}^i := (x_1, \ldots, x_{i-1}, x_i)$ and $\hat{x}^i := (x_1, \ldots, x_{i-1}, 1 - x_i)$. Clearly, we have $V_{t+1} = V_t \cup \{x^{t+1}\}$ and $V^i_{t+1} = V^i_t \cup \{\bar{x}^i\}$. Now, to obtain $\hat{V}^i_{t+1}$, observe that

$$\begin{aligned}
\hat{V}^i_{t+1} &= \left[V^{i-1}_{t+1} \times \{0,1\}\right] \setminus V^i_{t+1} \\
&= \left[V^{i-1}_t \times \{0,1\} \cup \{\hat{x}^i, \bar{x}^i\}\right] \setminus \left[V^i_t \cup \{\bar{x}^i\}\right] \\
&= \left[V^{i-1}_t \times \{0,1\} \cup \{\hat{x}^i\}\right] \setminus \left[V^i_t \cup \{\bar{x}^i\}\right] \\
&= \left(\left[V^{i-1}_t \times \{0,1\}\right] \setminus \left[V^i_t \cup \{\bar{x}^i\}\right]\right) \cup \left(\{\hat{x}^i\} \setminus \left[V^i_t \cup \{\bar{x}^i\}\right]\right) \\
&= \left(\hat{V}^i_t \setminus \{\bar{x}^i\}\right) \cup \left(\{\hat{x}^i\} \setminus \ V^i_t\right).
\end{aligned}$$

Therefore, if $\hat{x}^i \notin \hat{V}^i_t$ and $\hat{x}^i \notin V^i_t$, then $\hat{x}^i$ is included in $\hat{V}^i_{t+1}$. Also, if $\bar{x}^i \in \hat{V}^i_t$, then $\bar{x}^i$ is removed to obtain $\hat{V}^i_{t+1}$. Further observe that both operations cannot occur at the same iteration since the equivalence

$$\bar{x}^i \in \hat{V}^i_t \iff \hat{x}^i \in V^i_t \ \wedge \ \bar{x}^i \notin V^i_t$$

implies that $\hat{x}^i \notin V^i_t$ and $\bar{x}^i \in \hat{V}^i_t$ cannot hold true at the same time.

It follows that updating V involves adding or removing at most one vector for each W^i, totaling at most n such operations. The system CGLP is updated accordingly by appending or dropping at most n rows in (11). Also, x^{t+1} takes the place of x^t in (13) and the cut corresponding to x^t now takes the form (12) by changing the equality sign into inequality. The procedure to update CGLP is shown in Algorithm 5.

Algorithm 5 Updating CGLP

```
Input: CGLP, $V^i$, $\hat{V}^i$, $t$, $x^{t+1} = (x_1, \ldots, x_n)$
Output: Updated CGLP, $V^i$, $\hat{V}^i$
 1: for $1 \leq i \leq n$ do
 2:     $\bar{x}^i \leftarrow (x_1, \ldots, x_{i-1}, x_i)$
 3:     $\hat{x}^i \leftarrow (x_1, \ldots, x_{i-1}, 1 - x_i)$
 4:     if $\hat{x}^i \notin \hat{V}^i$ and $\hat{x}^i \notin V^i$ then
 5:         $w \leftarrow \hat{x}^i \times \{0\}^{n-i}$
 6:         Add $w\sigma + \rho_i \geq 0$ to (11)
 7:         $\hat{V}^i \leftarrow \hat{V}^i \cup \{\hat{x}^i\}$
 8:     end if
 9:     if $\bar{x}^i \in \hat{V}^i$ then
10:         $w \leftarrow \bar{x}^i \times \{0\}^{n-i}$
11:         Remove $w\sigma + \rho_i \geq 0$ from (11)
12:         $\hat{V}^i \leftarrow \hat{V}^i \setminus \{\bar{x}^i\}$
13:     end if
14:     $V^i \leftarrow V^i \cup \{\bar{x}^i\}$
15: end for
16: Add $x^t\alpha + Q(x^t) + \eta \geq 0$ to (12)
17: Replace (13) with $x^{t+1}\alpha + Q(x^{t+1}) + \eta = 0$
```

4 Combined method

Now we outline an implementation of the integer L-shaped method that combines the alternating strategy discussed in Section 2 with the new optimality cuts presented in Section 3.

We keep two disjoint lists of first-stage solutions: in V_{LP} we include solutions for which only $Q_{LP}(x)$ has been computed, while in V we keep solutions for which $Q(x)$ has been evaluated. At any given stage, we assume that for each $x \in V$ we have added an optimality cut that is tight at x. Now, when a candidate integer solution (x^*, z^*, θ^*) is found in the master tree, we check whether $x^* \in V$ or not. If so, we accept the solution as we already know $Q(x^*) \leq \theta^*$. Now, if $x^* \notin V_{LP}$, then we compute $Q_{LP}(x^*)$, we add x^* into V_{LP}, and in case $\theta < Q_{LP}(x^*)$, we add the subgradient cut (7). At this point, if (x^*, θ^*) has been neither accepted nor rejected, we have $x^* \in V_{LP}$ and $\theta^* \geq Q_{LP}(x^*)$. Thus we compute $Q(x^*)$, we move x^* from V_{LP} to V, and in case $\theta < Q(x^*)$, we add the CGLP-based cut (14) and accept the solution otherwise. Algorithm 6 presents the method.

5 Results

In this section we address the performance of the variants of the integer L-shaped method discussed so far. Given that the implementations differ in the cut strategy used and in the type of optimality cut added, we consider the following combinations:

1. *Std-Std*: standard cut strategy and standard optimality cut (6); see Section 1.
2. *Alt-Std*: alternating cut strategy and standard optimality cut (6); see Section 2.
3. *Std-CGLP*: standard cut strategy and new optimality cut (14); see Section 3.
4. *Alt-CGLP*: alternating cut strategy and new optimality cut (14); see Section 4.

Algorithm 6 Optimality cut function with alternating cut strategy and CGLP-based optimality cuts

```
Input: $(x^*, z^*, \theta^*)$ candidate integer solution, $Q : X \to \mathbb{R}, Q_{LP} : X \to \mathbb{R}, V, V_{LP}$
Output: true if solution is feasible, false otherwise
 1: if $x^* \in V$ then // We know $\theta^* \geq Q(x^*)$
 2:     return true
 3: end if
 4: if $x^* \notin V_{LP}$ then
 5:     Compute $Q_{LP}(x^*)$.
 6:     $V_{LP} \leftarrow V_{LP} \cup \{x^*\}$.
 7:     if $\theta^* < Q_{LP}(x^*)$ then
 8:         Add the subgradient cut (7).
 9:         return false
10:     end if
11: end if
    // Now we have $x^* \in V_{LP}$ and $\theta^* \geq Q_{LP}(x^*)$
12: Compute $Q(x^*)$.
13: Update CGLP.
14: $V \leftarrow V \cup \{x^*\}$.
15: $V_{LP} \leftarrow V_{LP} \setminus \{x^*\}$
16: if $\theta^* < Q(x^*)$ then
17:     Solve CGLP to obtain $\alpha$ and $\eta$
18:     Add the integer optimality cut (14)
19:     return false
20: else
21:     return true
22: end if
```

In other words, *Std-Std* corresponds to the usual implementation of the integer L-shaped method, on top of which the variants are built.

Our computational implementation uses CPLEX 12.5.0.1 as a solver and its Callable Library for advanced control routines. Since either optimality cuts (6) or (14) are part of the complete formulation (MP) but not included from the beginning, we have to add them on-the-fly through the optimality cut function. This routine is called every time the solver finds a candidate integer solution to the master problem and is in charge of generating an optimality cut if needed. In the case of CGLP, it calls additional subroutines to make the required updates to generate (14). We include the formulation of the first-stage set in CGLP, along the subgradients cuts derived from the linear relaxation of $Q(x)$ used in Benders' decomposition.

The experiments were carried out on a personal computer with 3.33 Ghz CPU, 4 Gb of RAM, and running Linux. A relative optimality gap of 0.01% was set as stopping criterion and a time limit of 7200 seconds was imposed. We do not report on the extensive form of the instances as solving them using an off-the-shelf solver is much slower than the decomposition approaches.

5.1 Stochastic server location problem

The stochastic server location problem is described in [13]. Given n locations, in the first stage we are asked to decide where to place servers so that the demand given by m potential customers is satisfied in the second stage. The uncertain data is the set of customers to be served in the second stage and the objective is to maximize the expected second-stage revenue minus the first-stage installation costs. In minimization form, the problem can be written as

$$\begin{array}{ll} \min & cx + Q(x) \\ \text{s.t.} & x \in \{0,1\}^n, \end{array}$$

where $Q(x) := \mathbb{E}_\xi[Q_\xi(x)]$ and

$$\begin{array}{rll} Q_\xi(x) := & \min & q_1 y + q_2 s \\ & \text{s.t.} & W_1 y + W_2 s \geq h(\xi) - Tx \\ & & y \in \{0,1\}^{nm} \\ & & s \in \mathbb{R}^n_+. \end{array}$$

The random right-hand-side vector $h(\xi)$ represents the set of active customers in a given scenario.

We tested our methods on the instances presented in [14] which are part of the Stochastic Integer Programming Library SIPLIB [1]. Instances `SSLP.n.m.k` have n locations, m customers, and k scenarios, leading to n binary variables in the first stage and nm binary variables and n nonnegative variables per scenario in the second stage. For each n and m, five replications with k scenarios each are considered. We did not include instances having $n = 5$ as all of them took less than 1 second to solve with any method.

Tables 1 and 2 summarize our results. In both tables, column *Instance* indicates the combination of n, m and k as above. Headers *Std-Std*, *Alt-Std*, *Std-CGLP*, *Alt-CGLP* denote the type of implementation under consideration. Here we present the averages over the five replications of each instance. Detailed results are given in Tables 6 and 7 in the Appendix.

In Table 1 we present the overall results for all four methods. Columns *Nodes* show the average number of nodes explored in the master problem. Columns *Time* show the average total time spent to reach optimality, which includes computing an initial lower bound L, solving the LP relaxation with Benders' decomposition, and exploring and evaluating candidate solutions in the master problem.

From Table 1, we see that there is no significant variation in the number of explored nodes among the different methods. In terms of solution time, the implementations that use the alternating cut strategy clearly outperform the other two methods, with speedups of one order of magnitude. On the other hand, with a few exceptions, the use of CGLP-based cuts does not cause major changes in the total running time, especially when combined with the alternating cut strategy. This can be explained by the fact that in these problems, the first-stage is very simple as $X = \{0,1\}^n$ with $n \leq 15$, which does not present a challenge for CPLEX.

To understand the effect of alternating cuts, in Table 2 we present details regarding subproblems. Recall that every time a candidate integer solution is found, we have to check whether it is feasible, by either solving a series of MIPs or LPs, one per scenario, and then add a cut to discard the solution if necessary. Headers *#LP* and *#MIP* denote the average number of times a candidate solution was checked using linear or mixed-integer subproblems, while headers *Time LP* and *Time MIP* indicate the average time spent in each case. We focus only on the implementations *Std-Std* and *Alt-Std* as the comparison for the remaining pair is similar.

From Table 2, we see that with the alternating cut strategy the number of MIP evaluations reduces considerably. This means that in the problems we tested, most of the time it is not necessary to compute the exact second-stage value of a given first-stage solution to reject it. Furthermore, only a small fraction of these solutions are visited twice, and only in those cases we have to solve MIP subproblems. The benefits are evident.

Instance	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
SSLP.10.50.50	402.4	70.9	394.8	71.5	406.8	6.8	404.2	6.8
SSLP.10.50.100	370.2	91.1	373.0	90.5	371.8	13.2	371.0	13.6
SSLP.10.50.500	381.0	548.5	385.0	561.7	386.8	64.0	385.0	65.5
SSLP.10.50.1000	360.0	1294.1	357.8	1307.1	367.4	128.2	368.2	129.3
SSLP.10.50.2000	392.2	3298.0	371.4	3160.7	404.4	339.3	404.6	336.7
SSLP.15.45.5	772.6	81.5	750.2	89.0	763.4	2.7	764.6	2.8
SSLP.15.45.10	1408.0	400.9	1370.8	353.6	1450.8	6.1	1414.0	6.5
SSLP.15.45.15	1500.0	534.3	1498.4	539.1	1526.0	11.7	1523.6	11.9
SSLP.15.45.20	495.6	358.4	481.4	347.8	500.4	8.0	502.4	8.1
SSLP.15.45.25	733.0	708.4	698.8	704.9	737.8	16.7	732.2	17.4

Table 1: Stochastic server location: overall results.

Instance	Std-Std				Alt-Std			
	#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
SSLP.10.50.50	147.6	147.6	1.8	65.8	148.6	3.4	1.7	1.8
SSLP.10.50.100	131.6	131.6	3.3	81.0	130.8	3.8	3.0	3.5
SSLP.10.50.500	131.6	131.6	16.4	497.2	130.6	3.0	14.8	15.2
SSLP.10.50.1000	132.0	132.0	33.6	1193.3	127.2	3.0	30.2	32.8
SSLP.10.50.2000	142.6	142.6	72.4	3082.5	143.4	4.2	67.3	133.2
SSLP.15.45.5	143.0	143.0	0.3	80.6	143.2	5.8	0.3	1.9
SSLP.15.45.10	262.0	262.0	1.1	398.2	268.5	5.3	1.1	3.6
SSLP.15.45.15	310.6	310.6	1.9	530.1	317.4	6.0	1.9	7.9
SSLP.15.45.20	99.4	99.4	0.7	356.1	98.4	3.2	0.7	5.9
SSLP.15.45.25	162.4	162.4	1.5	704.3	163.0	5.4	1.4	12.8

Table 2: Stochastic server location: subproblems details.

5.2 Stochastic multiple binary knapsack problem

The second benchmark set corresponds to a class of stochastic multiple binary knapsack problems. They have the form

$$\begin{aligned} \min\quad & cx + dz + Q(x) \\ \text{s.t.}\quad & Ax + Cz \geq b \\ & x \in \{0,1\}^n \\ & z \in \{0,1\}^n, \end{aligned}$$

where $Q(x) := \mathbb{E}_\xi[Q_\xi(x)]$,

$$\begin{aligned} Q_\xi(x) := \quad \min\quad & q(\xi)y \\ \text{s.t.}\quad & Wy \geq h - Tx \\ & y \in \{0,1\}^n, \end{aligned}$$

and all data are nonnegative integers. In the second-stage problem, only the objective vector $q(\xi)$ is random, following a discrete distribution with finitely many scenarios.

We generated 30 instances of the above problem with $n = 120$ and 20 equiprobable scenarios. The systems $Ax + Cz \geq b$ and $Wy \geq h - Tx$ have 50 and 5 rows, respectively. The entries of A, C, T, W, c, d, and q are i.i.d. sampled from the uniform distribution over $\{1, \ldots, 100\}$. We set $b = \frac{3}{4}(A\mathbf{1} + C\mathbf{1})$ and

$h = \frac{3}{4}(T\mathbf{1} + W\mathbf{1})$, with $\mathbf{1}$ denoting the n-dimensional vector of ones. These instances can be found in SIPLIB [1].

We divided the instances intro three groups depending on how much time the standard implementation took to solve each of them: *Easy* (less than 200 seconds, instances 1–6), *Medium* (between 200 and 1000 seconds, instances 7–18), and *Hard* (more than 1000 seconds, instances 19–29). We ommitted instance 30 since none of the methods was able to solve it to optimality within the time limit.

Tables 3, 4, and 5 below summarize the results. Column *Difficulty* denotes the instance class. The remaining headers and columns are as in Tables 1 and 2. Detailed results are given in Tables 8, 9, and 10 in the Appendix.

Difficulty	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
Easy	151531.5	87.1	127696.0	82.5	154611.7	86.0	133724.2	82.8
Medium	945487.8	520.8	714822.5	453.8	940249.3	516.1	748502.7	446.7
Hard	3356158.1	2125.7	2654448.1	1833.6	3371088.5	2065.2	2656526.5	1756.3

Table 3: Stochastic multiple knapsack: overall results.

From Table 3, we see that the application of the alternating cut strategy does not yield the time savings we saw with the stochastic server location problems. On the other hand, in most instances, adding CGLP-based cuts instead of standard cuts yields reductions in both the number of nodes and the total time, regardless of the cut strategy being used. We would like to conclude that these improvements are due to the fact that CGLP-based cuts help to explore the master tree. However, at this point, that is not completely clear, as for example, time reductions could be consequence of less evaluations of $Q(x)$ and not because of the strength of the new cuts.

To aid our analysis, in Table 4 we report the average number of candidate solutions for which $Q_{LP}(x)$ and $Q(x)$ were evaluated and the average time spent doing so. This time we compare *Std-Std* and *Std-CGLP*, and the notation is similar to that of Table 2.

Difficulty	Std-Std				Std-CGLP			
	#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
Easy	13.7	13.7	0.0	25.7	14.7	14.7	0.0	28.1
Medium	49.2	49.2	0.1	99.1	54.1	54.1	0.1	107.2
Hard	112.3	112.3	0.2	231.6	114.9	114.9	0.2	235.8

Table 4: Stochastic multiple knapsack: subproblems details.

We observe that both implementations require roughly the same number of evaluations of both $Q_{LP}(x)$ and $Q(x)$, which explains why alternating cuts does not outperform the standard cut strategy. Moreover, the difference in the time solving subproblems is very small compared to the total running times presented in Table 3. Thus, the reductions observed in Table 3 can be attributed to the better approximation of the first-stage set given by the CGLP-based cuts and not to the variability of the evaluations. In this regard, it is important to stress that, in principle, having a better description of the first-stage set does not have a direct relationship with the number of candidates solutions found in the master tree, and actually, having more candidates could hurt the total running time if their evaluation is too costly. However, in situations where after decomposing the problem the burden of the computation lies on the master problem, our improved cuts may prove beneficial as exemplified by our results.

Finally, in Table 5 we present the overhead incurred by using CGLP to generate cuts, that is, the time spent in additional operations to maintain and solve CGLP through the method. For each class, column $|V|$ shows the average final size of V, which is the number of candidate solutions for which $Q(x)$ was evaluated exactly. Headers *Update* and *Generate* denote the average total time spent updating the formulation of CGLP and actually solving the system to find an optimality cut, respectively. This additional time is already included in the total running time presented in Table 3.

Difficulty	$\lvert V \rvert$	Update	Generate
Easy	14.7	0.0	0.5
Medium	54.1	0.2	7.2
Hard	114.9	0.4	24.7

Table 5: Stochastic multiple knapsack: CGLP overhead.

As expected, the overhead increases as more solutions are included in the extended formulation. Updating CGLP takes practically no time, whereas generating the cut takes a nonnegligible amount of time. However, compared to the total running time, the overhead is very small and the effort of computing improved cuts pays off as shown in Table 3. For more complicated problems where the number of binary first-stage variables is too large or where too many candidate solutions are evaluated, the cost of maintaining CGLP is likely to be higher. In those cases, we can enforce rules to limit the number of calls to CGLP, such as using the standard optimality cuts as a baseline and applying the improved cuts only once in a while.

6 Concluding remarks

In this work, we have presented two modifications to the integer L-shaped method with the objective of reducing the running time of the algorithm. The first one, termed alternating cuts strategy, seeks to avoid expensive evaluations of the second-stage cost function, while the second, the use of CGLP-based optimality cuts, helps to better approximate the shape of the epigraph of the cost function when evaluations at different points are available. Our computational results suggest the following:

1. The alternating cuts strategy works better in problems where the computational bottleneck of (IP) is in evaluating $Q(x)$. Even when that is not the case, this modification does not seem to hurt the total running times and thus it could be considered as the base method on top of which more evolved algorithms can be built.
2. CGLP-based cuts are a viable alternative when the first-stage set is difficult to explore and computing $Q(x)$ is a relatively cheap operation. As the sole purpose of these new cuts is to have a better representation of the epigraph of the second-stage cost function within the master problem, there is no guarantee about the number or the sequence of solutions for which $Q(x)$ is evaluated, and thus, in general, this method performs well when the impact of this variability is small compared with the effort of solving the master problem.
3. We also point out that our overall computational experience indicates that CGLP-based cuts are particularly suitable for problems having additional integer variables in the set Z, since a deep cut discarding a point (x^*, θ^*) in the (x, θ)-space may also prove effective in discarding a large number of points of the form (x^*, z, θ^*) for $z \in Z$.
4. As favorable conditions for both modifications are unlikely to be attained at the same time, we observe that time reductions in a combined method are mainly consequence of one strategy or the other, but not because of the combination of both. That being said, it would be interesting to experiment with implementations where CGLP also incorporates approximations of $Q(x)$ such as subgradient cuts or ad-hoc lower bounds rather than exact evaluations only. That would require also keeping track of firt-stage vectors x for which estimates of $Q(x)$ have been computed.
5. Finally, in more general settings where $Q(x)$ is an easy-to-evaluate nonconvex function for which a tractable convex underestimator is not available, CGLP-based cuts may prove helpful in solving

problems having the form (IP). Situations where $Q(x)$ is given by black-box computations remain a case study to be explored.

Acknowledgments

This work has been supported in part by the Air Force Office of Scientific Research Award FA9550-12-1-0154 and the National Science Foundation Grant 1331426.

References

[1] S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, F. Qiu, and S. Sen, *SIPLIB: A Stochastic Integer Programming Test Problem Library*, `http://www.isye.gatech.edu/~sahmed/siplib`, 2013.

[2] M. Albareda-Sambola, M.H. van der Vlerk, and E. Fernández, *Exact solutions to a class of stochastic generalized assignment problems*, European journal of operational research **173** (2006), 465–487.

[3] G. Angulo, S. Ahmed, S.S. Dey, and V. Kaibel, *Forbidden vertices*, Optimization Online (2013).

[4] E. Balas, *Disjunctive Programming*, Discrete Optimization II (E.L. Johnson, P.L. Hammer, and B.H. Korte, eds.), Annals of Discrete Mathematics, vol. 5, Elsevier, 1979, pp. 3–51.

[5] J.F. Benders, *Partitioning procedures for solving mixed-variables programming problems*, Numerische Mathematik **4** (1962), 238–252.

[6] D. Gade, S. Küçükyavuz, and S. Sen, *Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs*, Mathematical Programming (2012), 1–26.

[7] M. Gendreau, G. Laporte, and R. Séguin, *An exact algorithm for the vehicle routing problem with stochastic demands and customers*, Transportation Science **29** (1995), 143–155.

[8] S. Küçükyavuz and M. Zhang, *Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs*, Optimization Online (2013).

[9] G. Laporte and F.V. Louveaux, *The integer L-shaped method for stochastic integer programs with complete recourse*, Operations Research Letters **13** (1993), 133–142.

[10] G. Laporte, F.V. Louveaux, and H. Mercure, *A priori optimization of the probabilistic traveling salesman problem*, Operations research **42** (1994), 543–549.

[11] G. Laporte, F.V. Louveaux, and L. van Hamme, *Exact solution to a location problem with stochastic demands*, Transportation Science **28** (1994), 95–103.

[12] ______, *An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands*, Operations Research **50** (2002), 415–423.

[13] L. Ntaimo and S. Sen, *The million-variable march for stochastic combinatorial optimization*, Journal of Global Optimization **32** (2005), 385–400.

[14] L. Ntaimo and M.W. Tanner, *Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs*, Journal of Global Optimization **41** (2008), 365–384.

[15] Y. Qi and S. Sen, *Ancestral Benders' cuts and multi-term disjunctions for mixed-integer recourse decisions in stochastic programming*, Optimization Online (2013).

[16] S. Sen and J.L. Higle, *The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification*, Mathematical Programming **104** (2005), 1–20.

[17] R.M. Van Slyke and R. Wets, *L-shaped linear programs with applications to optimal control and stochastic programming*, SIAM Journal on Applied Mathematics **17** (1969), 638–663.

Appendix

Stochastic server location problem

Instance	Rep.	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
SSLP.10.50.50	a	478	61.1	485	67.8	468	7.0	466	7.1
	b	452	91.3	434	85.5	472	6.7	466	6.7
	c	300	79.3	297	80.0	303	7.0	298	7.1
	d	237	25.3	224	26.9	230	5.0	230	5.0
	e	545	97.5	534	97.2	561	8.2	561	8.3
SSLP.10.50.100	a	452	109.7	434	106.2	462	17.1	470	18.3
	b	497	80.3	494	83.7	493	11.0	493	11.1
	c	313	95.3	291	98.0	302	12.8	289	13.6
	d	216	49.5	224	43.6	229	10.7	229	10.5
	e	373	120.5	422	121.0	373	14.2	374	14.4
SSLP.10.50.500	a	466	605.5	470	643.9	472	63.3	476	63.5
	b	441	482.6	447	492.9	449	57.7	449	57.8
	c	277	571.7	292	557.6	275	64.0	271	64.8
	d	235	348.8	239	353.5	247	57.5	247	57.6
	e	486	733.8	477	760.8	491	77.5	482	84.0
SSLP.10.50.1000	a	481	1542.1	473	1549.6	486	134.5	487	135.4
	b	473	1128.7	477	1142.2	460	114.5	466	116.8
	c	276	1509.3	261	1509.7	282	124.2	279	125.5
	d	225	752.8	227	782.2	229	113.2	229	113.7
	e	345	1537.6	351	1551.8	380	154.4	380	155.3
SSLP.10.50.2000	a	466	3777.1	467	3769.2	472	382.7	478	373.2
	b	472	2565.3	471	2751.8	483	246.7	478	251.0
	c	286	3189.4	286	3158.8	302	368.9	300	360.5
	d	219	1937.1	219	1994.5	223	249.0	225	249.4
	e	518	5021.2	414	4129.2	542	449.4	542	449.6
SSLP.15.45.5	a	230	11.3	233	11.6	244	0.7	244	0.7
	b	261	2.9	262	3.0	270	0.5	262	0.5
	c	2364	320.9	2288	354.9	2298	9.9	2294	10.2
	d	870	56.2	826	58.7	872	1.4	888	1.7
	e	138	16.4	142	16.8	133	1.0	135	1.0
SSLP.15.45.10	a	430	79.0	442	80.1	429	2.7	428	2.8
	b	284	189.0	251	190.9	256	6.2	278	7.6
	c	2384	245.0	2240	236.6	2512	7.4	2449	7.7
	d	2534	1090.7	2550	906.8	2606	7.9	2501	8.0
SSLP.15.45.15	a	1408	1646.1	1329	1594.5	1368	13.1	1358	13.3
	b	223	55.7	216	55.5	212	2.3	219	2.3
	c	2676	580.6	2718	611.0	2791	19.0	2785	18.9
	d	2986	359.1	2994	404.1	3038	22.3	3024	23.0
	e	207	30.1	235	30.2	221	1.9	232	1.9
SSLP.15.45.20	a	498	186.4	469	181.4	506	4.0	523	4.1
	b	351	87.2	335	87.5	341	7.6	331	7.6
	c	380	196.8	358	193.4	380	5.1	387	5.2
	d	552	873.0	548	898.1	560	20.7	562	20.9
	e	697	448.4	697	378.5	715	2.8	709	2.8
SSLP.15.45.25	a	658	554.1	629	532.0	662	18.4	633	18.5
	b	671	324.7	620	435.1	670	9.0	680	6.7
	c	433	165.2	399	160.7	447	11.8	422	11.9
	d	965	435.2	946	465.7	967	26.9	1001	32.3
	e	938	2062.7	900	1931.0	943	17.6	925	17.8

Table 6: Stochastic server location: overall results per instance.

Instance	Rep.	Std-Std				Alt-Std			
		#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
SSLP.10.50.50	a	189	189	2.2	55.5	185	3	2.1	1.7
	b	154	154	1.6	86.2	166	3	1.6	1.6
	c	113	113	1.6	74.5	109	4	1.6	2.3
	d	44	44	0.6	21.3	45	2	0.6	0.9
	e	238	238	2.8	91.6	238	5	2.7	2.6
SSLP.10.50.100	a	181	181	4.3	98.4	183	6	3.9	6.4
	b	176	176	4.0	69.7	175	2	3.7	0.9
	c	112	112	3.1	86.2	109	5	2.9	4.1
	d	51	51	1.3	40.9	50	2	1.2	2.2
	e	138	138	3.7	109.8	137	4	3.3	4.0
SSLP.10.50.500	a	178	178	21.1	549.8	179	2	19.1	11.0
	b	152	152	17.8	428.5	150	2	15.2	7.4
	c	89	89	13.7	523.9	89	4	12.0	18.6
	d	56	56	8.1	303.3	56	2	8.1	12.4
	e	183	183	21.1	680.4	179	5	19.8	26.7
SSLP.10.50.1000	a	188	188	46.4	1429.6	185	3	41.9	29.4
	b	163	163	36.6	1028.5	156	2	32.4	21.2
	c	106	106	29.6	1410.1	95	3	25.8	30.2
	d	56	56	16.0	665.2	55	2	15.3	27.3
	e	147	147	39.4	1433.1	145	5	35.5	56.1
SSLP.10.50.2000	a	184	184	92.6	3548.0	181	5	82.4	169.9
	b	158	158	70.3	2352.7	156	2	65.4	44.2
	c	98	98	60.7	2980.4	103	5	56.6	167.0
	d	59	59	34.2	1746.0	58	2	31.2	62.1
	e	214	214	104.3	4785.4	219	7	101.0	222.7
SSLP.15.45.5	a	28	28	0.1	10.9	28	2	0.1	0.2
	b	42	42	0.1	2.5	41	4	0.1	0.2
	c	481	481	1.0	318.3	496	17	0.9	7.7
	d	154	154	0.3	55.2	141	4	0.3	0.6
	e	10	10	0.0	16.1	10	2	0.0	0.7
SSLP.15.45.10	a	93	93	0.3	77.8	90	2	0.3	1.6
	b	68	68	0.2	188.2	67	5	0.2	5.4
	c	501	501	2.2	240.1	538	9	2.3	2.9
	d	386	386	1.7	1086.8	379	5	1.7	4.4
SSLP.15.45.15	a	263	263	1.6	1642.3	262	4	1.5	9.7
	b	41	41	0.2	54.4	39	2	0.2	1.0
	c	623	623	4.3	572.8	645	16	4.4	11.8
	d	597	597	3.3	352.0	613	6	3.1	16.2
	e	29	29	0.2	29.0	28	2	0.2	0.8
SSLP.15.45.20	a	134	134	0.9	183.7	132	2	0.9	1.4
	b	63	63	0.4	85.1	61	2	0.4	5.6
	c	61	61	0.4	195.2	60	4	0.4	3.6
	d	148	148	1.1	870.2	145	6	1.0	18.0
	e	91	91	0.7	446.2	94	2	0.7	0.7
SSLP.15.45.25	a	156	156	1.3	550.0	147	4	1.2	14.4
	b	135	135	1.3	321.0	148	4	1.4	5.3
	c	73	73	0.6	162.1	74	4	0.6	8.8
	d	213	213	2.2	430.0	215	7	2.1	21.9
	e	235	235	2.0	2058.5	231	8	1.9	13.7

Table 7: Stochastic server location: subproblems details per instance.

Stochastic multiple binary knapsack problem

Instance	Std-Std		Std-CGLP		Alt-Std		Alt-CGLP	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
1	27705	26.4	27615	27.0	27837	25.5	26295	24.9
2	63528	41.1	55448	38.6	65213	41.1	57170	38.2
3	93185	59.9	87560	60.2	101121	57.8	81480	50.9
4	137303	101.1	121687	97.3	132782	89.1	127963	89.1
5	224063	107.2	183462	94.1	244755	112.1	251017	128.3
6	363405	186.6	290404	177.5	355962	190.1	258420	165.5
7	503998	245.8	401809	204.4	517313	250.4	397677	200.2
8	436738	267.5	310136	218.0	431356	249.3	334569	214.8
9	470356	273.3	451931	269.7	502174	280.5	450104	254.8
10	507120	315.6	320672	251.1	518329	333.1	342582	257.5
11	623424	379.4	675292	404.9	637749	422.5	615580	342.6
12	887595	468.7	672117	422.7	954211	502.8	741931	436.2
13	1099397	541.0	1024147	692.2	1172464	579.1	984003	527.4
14	1416129	686.6	880154	516.9	1484427	711.5	1057895	600.3
15	1650580	714.4	1120524	509.8	1692521	726.8	1148229	516.0
16	1322774	749.9	832266	533.7	1013473	572.2	956447	579.2
17	1197577	771.1	900476	652.4	1192205	753.2	974525	686.0
18	1230166	836.7	988346	769.7	1166769	811.6	978490	745.9
19	2189204	1158.0	1618305	950.0	2225393	1160.4	1713778	962.0
20	2395096	1460.9	1663945	1142.5	2383548	1404.2	1756720	1109.5
21	3277812	1488.2	2789613	1328.8	3563188	1603.1	3144784	1499.3
22	2702878	1664.7	2244862	1422.6	2816341	1714.0	2087732	1430.1
23	2309196	1825.3	1919811	1711.6	2306792	1715.5	1833302	1520.5
24	3301135	1998.1	2690441	1771.6	3101311	1816.8	2580654	1620.4
25	3346788	2310.7	2987190	2149.9	3541754	2346.8	2998747	2068.1
26	3024670	2319.8	2966064	2373.0	3087399	2258.1	2806757	2172.3
27	3890594	2344.4	3225433	2099.7	3787260	2210.1	3128508	1980.4
28	4762714	3223.2	3253202	2425.3	4449516	2890.2	3285741	2311.3
29	5717652	3589.2	3840063	2795.1	5819471	3597.8	3885068	2645.9

Table 8: Stochastic multiple knapsack: overall results per instance.

Instance	Std-Std				Std-CGLP			
	#LP	#MIP	Time LP	Time MIP	#LP	#MIP	Time LP	Time MIP
1	9	9	0.0	14.4	9	9	0.0	14.7
2	9	9	0.0	15.7	9	9	0.0	15.8
3	14	14	0.0	24.7	14	14	0.0	26.0
4	24	24	0.0	46.2	24	24	0.0	45.9
5	12	12	0.0	21.0	12	12	0.0	19.8
6	14	14	0.0	32.1	20	20	0.0	46.2
7	10	10	0.0	17.3	10	10	0.0	17.2
8	40	40	0.1	80.3	40	40	0.1	80.7
9	34	34	0.1	74.5	36	36	0.1	74.6
10	46	46	0.1	97.9	49	49	0.1	102.1
11	46	46	0.1	77.6	47	47	0.1	78.9
12	45	45	0.1	108.5	51	51	0.1	123.0
13	45	45	0.1	87.2	87	87	0.2	160.9
14	51	51	0.1	124.4	51	51	0.1	123.9
15	22	22	0.0	29.9	26	26	0.1	36.2
16	79	79	0.2	128.8	74	74	0.2	119.3
17	80	80	0.2	168.0	81	81	0.2	167.5
18	92	92	0.2	194.7	97	97	0.2	202.1
19	66	66	0.1	134.3	65	65	0.1	131.9
20	97	97	0.2	193.0	98	98	0.2	193.9
21	49	49	0.1	99.8	48	48	0.1	97.7
22	93	93	0.2	245.2	91	91	0.2	237.2
23	175	175	0.4	341.7	176	176	0.4	339.1
24	89	89	0.2	211.2	92	92	0.2	221.1
25	127	127	0.3	222.2	127	127	0.3	221.8
26	155	155	0.3	331.5	157	157	0.3	331.4
27	103	103	0.2	246.0	111	111	0.2	264.1
28	150	150	0.3	263.7	152	152	0.3	268.0
29	131	131	0.3	259.4	147	147	0.3	287.6

Table 9: Stochastic multiple knapsack: subproblems details per instance.

Instance	$\|V\|$	Update	Generate
1	9	0.0	0.2
2	9	0.0	0.2
3	14	0.0	0.4
4	24	0.0	1.0
5	12	0.0	0.3
6	20	0.0	0.8
7	10	0.0	0.2
8	40	0.1	2.7
9	36	0.1	2.5
10	49	0.1	3.4
11	47	0.1	4.7
12	51	0.1	4.5
13	87	0.2	13.2
14	51	0.1	5.3
15	26	0.1	1.3
16	74	0.2	11.9
17	81	0.3	15.7
18	97	0.4	21.2
19	65	0.2	6.9
20	98	0.3	21.7
21	48	0.1	4.0
22	91	0.3	18.8
23	176	0.9	41.2
24	92	0.3	13.8
25	127	0.4	29.8
26	157	0.6	37.0
27	111	0.4	25.1
28	152	0.6	39.6
29	147	0.6	33.3

Table 10: Stochastic multiple knapsack: CGLP overhead per instance.