Online Supplement for "On Generating Lagrangian Cuts for Two-Stage Stochastic Integer Programs"

Rui Chen, James Luedtke
University of Wisconsin-Madison
rchen234@wisc.edu, jim.luedtke@wisc.edu

February 20, 2022

S.1 A Branch-and-Cut Algorithm

A standard branch-and-cut algorithm for solving two-stage SIPs with either continuous recourse or pure binary first-stage variables is described in Algorithm 1. The algorithm starts with the approximations $\{\hat{Q}_s\}_{s\in S}$ of $\{Q_s\}_{s\in S}$ uses it to create an LP initial LP relaxation of problem (4) represented by the root node 0 which is added to the list of candidate nodes $\mathcal{L}$. In our case, the approximations $\{\hat{Q}_s\}_{s\in S}$ are constructed by Benders cuts and Lagrangian cuts. Within each iteration, the algorithm picks a node from $\mathcal{L}$ and solves the corresponding LP which is a relaxation of problem (4) with the addition of the constraints adding from branching, and generates a candidate solution $(\hat{x}, \{\hat{\theta}_s\}_{s\in S})$. If $\hat{x}$ does not satisfy the integrality constraints ($\hat{x} \notin X$), the algorithm creates two new subproblems (nodes) by subdividing the feasible solutions to that node. The subproblems are created by choosing an integer decision variable x_j having $\hat{x}_j \notin \mathbb{Z}$, and adding the constraint $x_j \leq \lfloor \hat{x}_j \rfloor$ in one of the subproblems and adding the constraint $x_j \geq \lceil \hat{x}_j \rceil$ in the other subproblem. If $\hat{x}$ satisfies the integrality constraints ($\hat{x} \in X$), the algorithm evaluates $Q_s(\hat{x})$ for all $s \in S$. If any violated Benders cuts and integer L-shaped cuts are identified, these are used to update $\{\hat{Q}_s\}_{s\in S}$, and the relaxation at that node is then solved again. This process of searching for and adding cuts if violated when $\hat{x} \in X$ can be implemented using the "Lazy constraint callback" in commercial solvers. If no violated Benders or integer L-shaped cuts are identified, then the current solution is feasible and thus the upper bound $\bar{z}$ and best known solution (the incumbent x^*) is updated. The algorithm terminates when there are no more nodes to explore in the candidate list $\mathcal{L}$.

Algorithm 1: Branch-and-cut for SIPs.

```
   Input: $\{\hat{Q}_s\}_{s\in S}$
   Output: $(x^*, \{\theta_s^*\}_{s\in S})$
1  Initialize $\mathcal{L} \leftarrow \{0\}$, LP$_0 \leftarrow \min_{x,\theta_s}\{c^T x + \sum_{s\in S} p_s\theta_s : \theta_s \geq \hat{Q}_s(x), s\in S, Ax \geq b\}$, $\bar{z} \leftarrow +\infty$,
    $(x^*, \{\theta_s^*\}_{s\in S}) \leftarrow \emptyset$
2  while $\mathcal{L} \neq \emptyset$ do
3      Choose a node $i \in \mathcal{L}$
4      Solve LP$_i$ and, if feasible, obtain optimal solution $(\hat{x}, \{\hat{\theta}_s\}_{s\in S})$ and optimal value $\hat{z}$
5      if $LP_i$ is feasible and $\hat{z} < \bar{z}$ then
6          if $\hat{x} \in X$ then
7              for $s \in S$ do
8                  Evaluate $Q_s(\hat{x})$
9                  Add the Benders cut (7) to update $\hat{Q}_s$ if violated
10                 if $X = \{0,1\}^n$ then
11                     Add the integer L-shaped cut (8) to update $\hat{Q}_s$ if violated
12                 end
13             end
14             if $\hat{\theta}_s = Q_s(\hat{x})$ for all $s \in S$ then
15                 $\mathcal{L} \leftarrow \mathcal{L} \setminus \{i\}$
16                 $(x^*, \{\theta_s^*\}_{s\in S}) \leftarrow (\hat{x}, \{\hat{\theta}_s\}_{s\in S})$
17                 $\bar{z} \leftarrow \hat{z}$
18             end
19         else
20             Choose an integer variable $x_j$ with $\hat{x}_j \neq \mathbb{Z}$, then branch on LP$_i$ to construct LP$_{i_1}$ and
                LP$_{i_2}$ such that LP$_{i_1}$ is LP$_i$ with an additional constraint $x_j \leq \lfloor \hat{x}_j \rfloor$ and LP$_{i_2}$ is LP$_i$ with
                an additional constraint $x_j \geq \lceil \hat{x}_j \rceil$
21             $\mathcal{L} \leftarrow (\mathcal{L} \setminus \{i\}) \cup \{i_1, i_2\}$
22         end
23     else
24         $\mathcal{L} \leftarrow \mathcal{L} \setminus \{i\}$
25     end
26 end
```

S.2 Test Problems

The SSLP problem

The SSLP problem [1] is a two-stage SIP with pure binary first-stage and mixed-binary second-stage variables. In this problem, the decision maker has to choose from m sites to allocate servers with some cost in the first stage. Then in the second stage, the availability of each client i would be observed and every available client must be served at some site. The first-stage variables are denoted by x with $x_j = 1$ if and only if a server is located at site j. The second-stage variables are denoted by y and y_0 where $y^s_{ij} = 1$ if and only if client i is served at site j in scenario s and y^s_{0j} is the amount of resource shortage at at site j in scenario s. Allocating a server costs c_j at site j. Each allocated server can provide up to u units of resource. Client i uses d_{ij} units of resource and generates revenue q_{ij} if served at site j. Each unit of resource shortage at site j incurs a penalty q_{0j}. The client availability in scenario s is represented by h^s with $h^s_i = 1$ if and only if client i is present in scenario s. The extensive formulation of the problem is as follows:

$$
\begin{aligned}
\min_{x,y^s,y^s_0} \quad & \sum_{j=1}^{m} c_j x_j + \sum_{s\in S} p_s \Big(\sum_{j=1}^{m} q_{0j} y^s_{0j} - \sum_{i=1}^{n}\sum_{j=1}^{m} q_{ij} y^s_{ij} \Big) & \\
\text{s.t.} \quad & \sum_{i=1}^{n} d_{ij} y^s_{ij} - y^s_{0j} \le u x_j, & j = 1,\dots,m,\ s\in S, \\
& \sum_{j=1}^{m} y^s_{ij} = h^s_i, & i = 1,\dots,n,\ s\in S, \\
& x_j \in \{0,1\}, & j = 1,\dots,m, \\
& y^s_{ij} \in \{0,1\}, & i = 1,\dots,n,\ j = 1,\dots,m,\ s\in S, \\
& y^s_{0j} \ge 0, & j = 1,\dots,m,\ s\in S.
\end{aligned}
$$

All SSLP instances are generated with $(m,n) \in \{(20,100),(30,70),(40,50),(50,40)\}$ and $|S| \in \{50,200\}$. For each size $(m,n,|S|)$, three instances are generated with $k \in \{1,2,3\}$. For each SSLP instance, the data are generated as:

- $p_s = 1/|S|$, $s \in S$.
- c_j ~Uniform($\{40,41,\dots,80\}$), $j = 1,\dots,m$.
- $d_{ij} = q_{ij}$ ~Uniform($\{0,1,\dots,25\}$), $i = 1,\dots,n$, $j = 1,\dots,m$.
- $q_{0j} = 1000$, $j = 1,\dots,m$.
- $u = \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}/m$.
- h^s_i ~Bernoulli(1/2), $i = 1,\dots,n$, $s \in S$.

All SSLP instances are named of the form sslp$k_m_n_|S|$ where m denotes the number of sites, n denotes number of clients, $|S|$ is the number of scenarios and k is the instance number.

The SSLPV problem

The SSLPV problem is a variant of the SSLP problem that has mixed-binary first-stage and pure continuous second-stage variables. There are two main differences between SSLP and SSLPV. First, in the SSLPV problem, in addition to choosing which sites to "open" for servers, the decision maker also has an option to add more servers than the "base number" if the site is open. Second, in the second stage, we allow the clients' demands to be partly met by the servers, and only unmet shortage will be penalized. Thus, in this problem the recourse problem is a linear program. We assume that if a site j is selected (binary decision) it has base capacity of u^b (independent of j), and installing this has cost c_j^b. In addition, it is possible to install any fraction of an additional u^n units of capacity at site j, with total cost c_j^n if the total new capacity is installed and the cost reduced proportionallly if a fraction is installed. The variable representing the amount of additional server capacity added at site j is denoted by a continuous variable $x_j^+ \in [0,1]$. The extensive formulation of the problem is as follows:

$$\begin{aligned}
\min_{x,x^+,y^s,y_0^s} \quad & \sum_{j=1}^{m} \left(c_j^b x_j + c_j^n x_j^+ \right) + \sum_{s\in S} p_s \Big(\sum_{j=1}^{m} q_{0j} y_{0j}^s - \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij} y_{ij}^s \Big) & \\
\text{s.t.} \quad & \sum_{i=1}^{n} d_{ij} y_{ij}^s - y_{0j}^s \le u^b x_j + u^n x_j^+, & j = 1,\dots,m,\ s\in S, \\
& \sum_{j=1}^{m} y_{ij}^s = h_i^s, & i = 1,\dots,n,\ s\in S, \\
& x_j \in \{0,1\},\ x_j^+ \in [0,1],\ x_j^+ \le x_j, & j = 1,\dots,m, \\
& y_{ij}^s \in [0,1], & i = 1,\dots,n,\ j = 1,\dots,m,\ s\in S, \\
& y_{0j}^s \ge 0, & j = 1,\dots,m,\ s\in S.
\end{aligned}$$

Each SSLPV instance corresponds to a SSLP instance. In all SSLPV instances, we set $u^b = (1-\rho)u$, $u^n = \rho u$, and for each site j, we set $c_j^b = (1-\rho)c_j$, $c_j^n = \rho c_j$, where $\rho = 0.5$ and u and c_j are the data from the SSLP instance. The other data is the same as the data of the SSLP instances. All SSLP instances are named of the form sslp$k$_m_n_$|S|$_var where m denotes the number of sites, n denotes number of clients, $|S|$ is the number of scenarios and k is the instance number.

The SNIP problem

The SNIP problem [2] is a two-stage SIP with pure binary first-stage and continuous second-stage variables. In this problem the defender interdicts arcs on a directed network by installing sensors in the first stage to maximize the probability of finding an attacker. Then in the second stage, the origin and destination of the attacker are observed and the attacker chooses the maximum reliability path from its origin to its destination. Let N and A denote the node set and the arc set of the network and let $D \subseteq A$ denote the set of interdictable arcs. The first-stage variables are denoted by x with $x_a = 1$ if and only if the defender installs a sensor on arc $a \in D$. The scenarios $s \in S$ define the possible origin/destination combinations of the attacker, with u^s representing the origin and v^s representing the destination of the attacker for each scenario $s \in S$. The second-stage variables are denoted by π where π_i^s denotes the maximum probability of reaching destination v^s undetected from node i in scenario s. The budget for installing sensors is b, and the cost of installing a sensor on arc a is c_a for each arc $a \in D$. For each arc $a \in A$, the probability of traveling on arc a undetected is r_a if the arc is not interdicted, or q_a if the arc is interdicted. Finally, let $\bar{\pi}_j^s$ denote the maximum probability of reaching the destination undetected from node j when no sensors are installed. The extensive formulation of the problem is as follows:

$$\begin{aligned}
\min_{x,\pi^s} \quad & \sum_{s \in S} p_s \pi_{u^s}^s \\
\text{s.t.} \quad & \sum_{a \in A} c_a x_a \leq b, \\
& \pi_i^s - r_a \pi_j^s \geq 0, && a = (i,j) \in A \setminus D,\ s \in S, \\
& \pi_i^s - r_a \pi_j^s \geq -(r_a - q_a)\bar{\pi}_j^s x_a, && a = (i,j) \in D,\ s \in S, \\
& \pi_i^s - q_a \pi_j^s \geq 0, && a = (i,j) \in D,\ s \in S, \\
& \pi_{v^s}^s = 1, && s \in S, \\
& x_a \in \{0,1\}, && a \in D.
\end{aligned}$$

All of the SNIP instances we used in this paper are from [2]. We consider instances with snipno = 3, and budget $b \in \{30, 50, 70, 90\}$. All instances have 320 first-stage binary variables, 2586 second-stage continuous variables per scenario and 456 scenarios.

S.3 Some Convergence Profiles

Figures 2 - 5 present the evolution of the progress in the lower bound over time on one representative example SSLP instance and one representative example SNIP instance using various cut generation approaches and parameters.

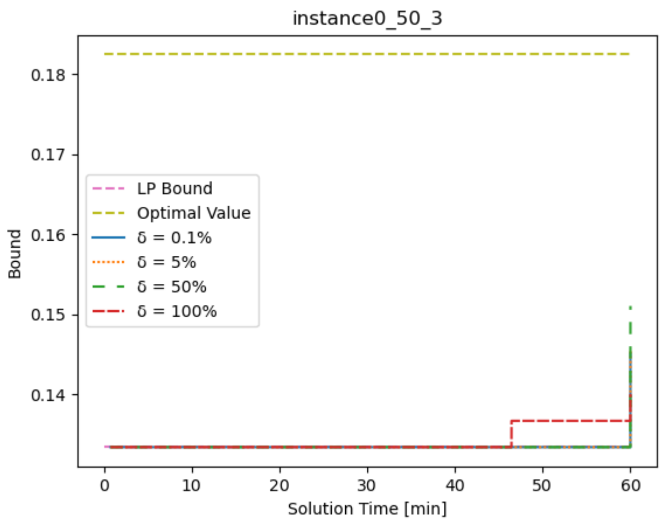

Figure 1: Convergence profile of an SNIP instance (instance0_50_3) with varying δ values.

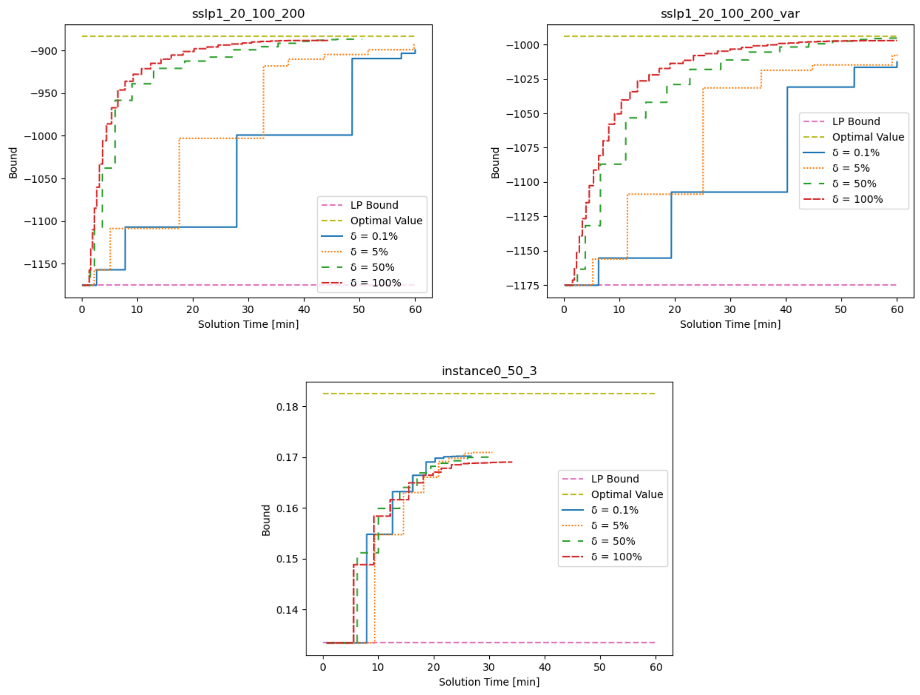

Figure 2: Convergence profile of *Rstr1* on an SSLP instance (top left), an SSLPV instance (top right), and a SNIP instance (bottom) with $K = 10$ and varying δ values.

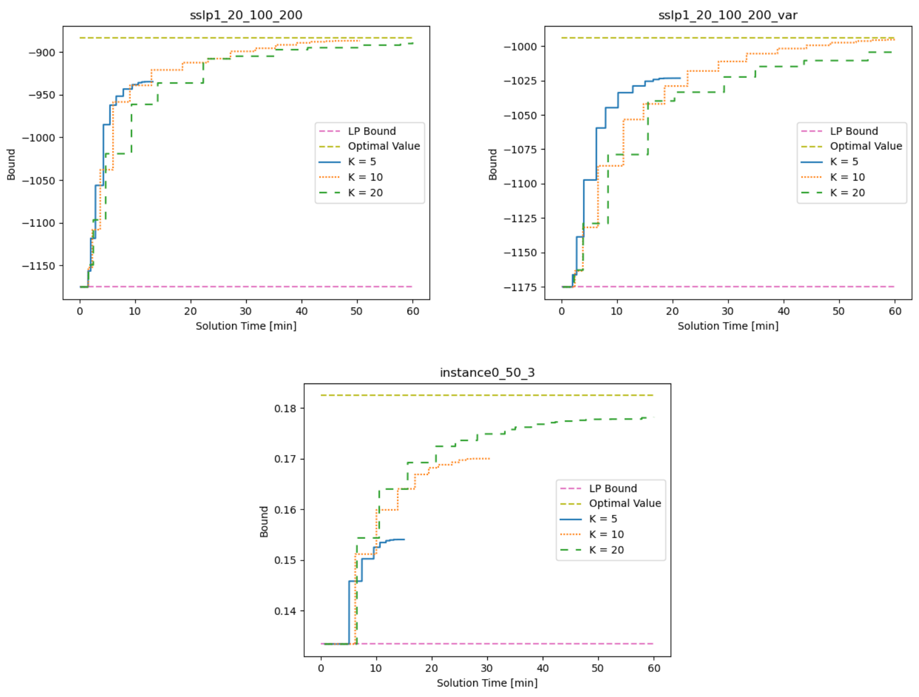

Figure 3: Convergence profile of *Rstr1* on an SSLP instance (top left), an SSLPV instance (top right), and a SNIP instance (bottom) with $\delta = 50\%$ and varying K values.

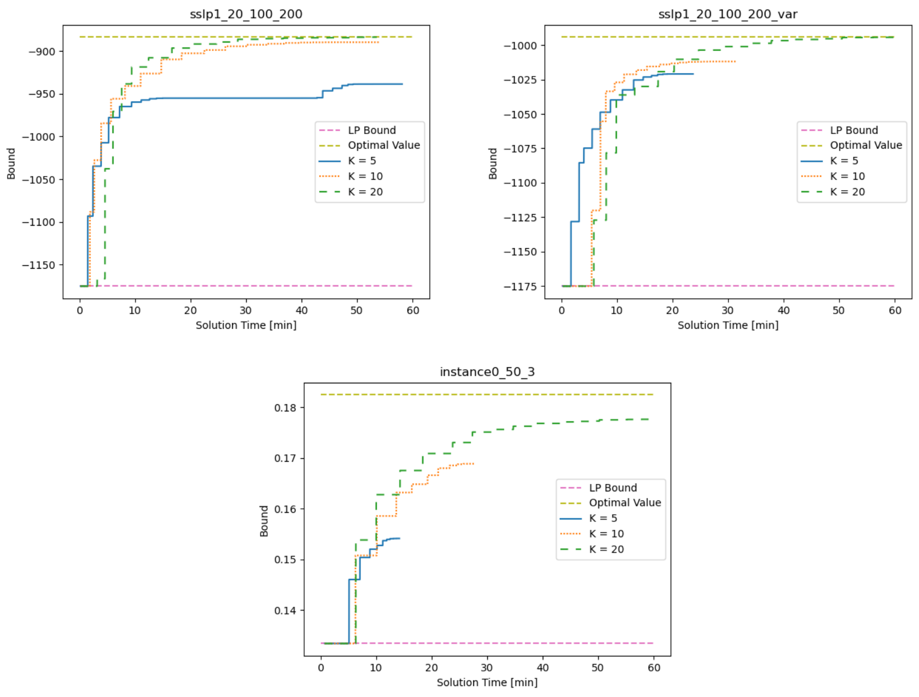

Figure 4: Convergence profile of *Rstr2* on an SSLP instance (top left), an SSLPV instance (top right), and a SNIP instance (bottom) with $\delta = 50\%$ and varying K values.

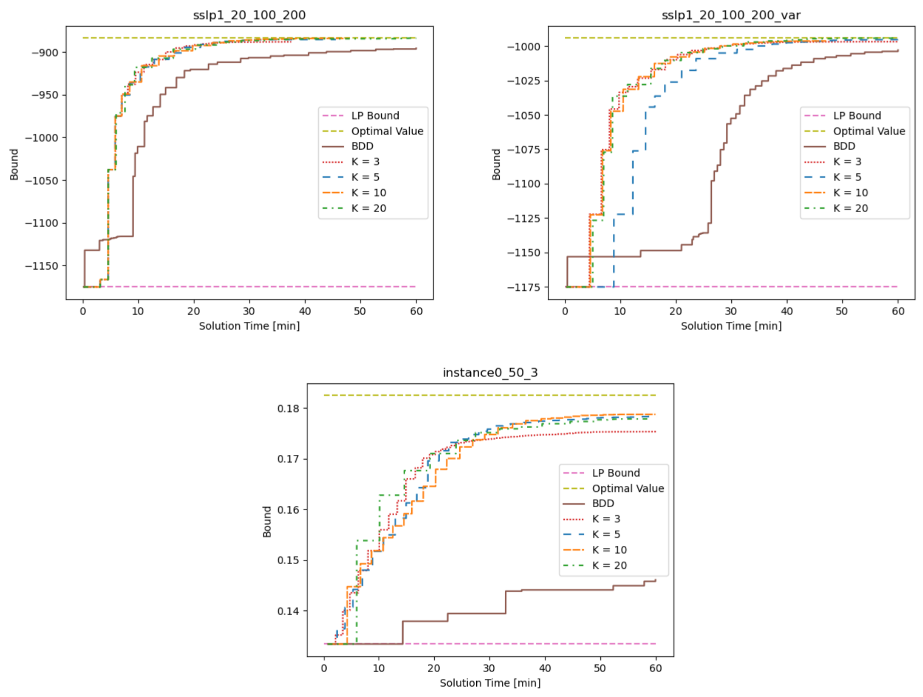

Figure 5: Convergence profile of *RstrMIP* on an SSLP instance (top left), an SSLPV instance (top right), and an SNIP instance (bottom) with $\delta = 50\%$ and varying K values and comparison with *BDD*.

From Figure 5 we see that for the example SSLP instance, *BDD* does not make much progress in the first 10 minutes, but then makes steady improvement. The shift occurs when *BDD* switches from its initial phase of generating only strengthened Benders cuts to the second phase of heuristically generating Lagrangain cuts. In the initial phase *BDD* the strengthened Benders cuts are not strong enough to substantially improve the bound after the first iteration, so that significant bound progress only occurs again after switching to the second phase. We experimented with making this transition quicker (or even skipping it altogether) but we found that this led to poor performance of *BDD*. It appears that the first phase is important for building an inner approximation that is used for separation of Lagrangian cuts in the next phase. The convergence profiles of *BDD* on the SNIP instance and the SSLPV instance clearly show its slow convergence relative to all variations of our proposed restricted Lagrangian cut generation approach.

References

[1] L. Ntaimo and S. Sen, "The million-variable "march" for stochastic combinatorial optimization," *Journal of Global Optimization*, vol. 32, no. 3, pp. 385–400, 2005.

[2] F. Pan and D. P. Morton, "Minimizing a stochastic maximum-reliability path," *Networks: An International Journal*, vol. 52, no. 3, pp. 111–119, 2008.