Proximity Benders: A Decomposition Heuristic for Stochastic Programs

Natashia Boland[a], Matteo Fischetti[b], Michele Monaci[b], and Martin Savelsbergh[a]

[a] *Georgia Institute of Technology, USA*
[b] *University of Padova, Italy*

December 7, 2015

Abstract

is eventually found and the method terminates. As convergence may require a large amount of computing time for hard instances, the method unsatisfactory from a heuristic point of view. Proximity Search is a recently-proposed heuristic paradigm in which the problem at hand is modified and iteratively solved with the aim of producing a sequence of improving feasible solutions. As such, Proximity Search and Benders decomposition naturally complement each other, in particular when the emphasis is on seeking high-quality, but not necessarily optimal, solutions. In this paper, we investigate the use of Proximity Search as a tactical tool to drive Benders decomposition, and computationally evaluate its performance as a heuristic on instances of different stochastic programming problems.

1 Introduction

Stochastic Programming (SP) is an important framework for dealing with uncertainty in optimization. SP models tend to be huge, and their solution typically requires a decomposition method to break the model into manageable parts. Benders decomposition (Benders [3]) is one of the most widely used approach for SP. In Benders decomposition,

Proximity Search (PS) is a general approach focused on improving a given feasible "reference solution", and seeking to quickly produce a sequence of improving feasible solutions (Fischetti and Monaci [10]). The approach is related to *Large-Neighborhood Search* (LNS) heuristics (Shaw [15]) that explore a neighborhood defined by constraints restricting the search space, in particular to *Local Branching* (Fischetti and Lodi [9]), which adds a constraint that eliminates all solutions that are not "sufficiently close" to a reference solution.

The generic Mixed-Integer Linear Program (MILP) of interest has the form

$$(MILP) \qquad \begin{aligned} &\min c^T x \\ &Ax \geq b, \\ &x_j \in \{0,1\},\ \forall j \in \mathcal{B}, \\ &x_j \in \mathbb{Z},\ \forall j \in \mathcal{G}, \\ &x_j \in \mathbb{R},\ \forall j \in \mathcal{C}, \end{aligned}$$

where A is an $m \times n$ input matrix, b and c are input vectors of dimension m and n, respectively, and the variable index set $\mathcal{N} := \{1, \ldots, n\}$ is partitioned into $\mathcal{B}, \mathcal{G}$, and $\mathcal{C}$, with $\mathcal{B}$ the index set of the 0-1 variables, $\mathcal{G}$ the index set of general integer variables, and $\mathcal{C}$ the index set of continuous variables. Removing the integrality requirement on variables with indices in $\mathcal{B} \cup \mathcal{G}$ gives the LP relaxation.

PS works in stages, each aimed at producing an improved feasible solution. In each stage, a reference solution $\widetilde{x}$ is given, and one seeks to improve it. To this end, an explicit cutoff constraint

$$c^T x \leq c^T \widetilde{x} - \theta \tag{1}$$

is added to the original MILP, where $\theta > 0$ is a given tolerance that specifies the minimum improvement required. The objective function of the problem can then be replaced by the proximity function

$$\Delta(x, \widetilde{x}) = \sum_{j \in \mathcal{B}: \widetilde{x}_j = 0} x_j + \sum_{j \in \mathcal{B}: \widetilde{x}_j = 1} (1 - x_j) \tag{2}$$

to be minimized. One then applies the MILP solver, as a black box, to the modified problem in the hope of finding a nearby solution better than $\widetilde{x}$ (see Algorithm 1 for more details).

Algorithm 1 *Proximity Search*

Let $\widetilde{x}$ be the initial feasible solution to improve
repeat
  Explicitly add cutoff constraint (1) to the MILP model
  Install the new objective function $\Delta(x, \widetilde{x})$, to be minimized
  Run the MILP solver on the new model until a termination condition is reached, and let x^* be the best feasible solution found
  Refine x^* by solving (possibly heuristically) the original MILP model after fixing $x_j = x_j^*$ for all $j \in \mathcal{B}$
  Recenter $\Delta(x, \cdot)$ by setting $\widetilde{x} := x^*$, and/or update θ
until *an overall termination condition is reached*

A powerful variant of the above scheme, called "proximity search with incumbent", is based on the idea of providing $\widetilde{x}$ to the MILP solver as a starting solution. To avoid $\widetilde{x}$ being rejected because of the cutoff constraint (1), the latter is weakened to its "soft" version

$$c^T x \leq c^T \widetilde{x} - \theta(1 - z) \tag{3}$$

while minimizing $\Delta(x, \widetilde{x}) + Mz$ instead of just $\Delta(x, \widetilde{x})$, where $z \geq 0$ is a continuous

Computational experience confirms that PS is quite successful (at least, on some classes of problems), due to the fact that the proximity function improves the "relaxation grip" of the model, meaning that the solutions of the LP relaxation tend to have a large number of integer components, thus improving the success rate of the heuristics embedded within the MILP solver. This is true, in particular, for the MILP models where feasibility is enforced dynamically through cut generation. In fact, the solutions to the LP-relaxation tend to be similar to the reference solution, thus most feasibility constraints are likely to be satisfied without the need to explicitly impose them.

PS has a "primal nature", meaning that it proceeds from a feasible solution to a "nearby" feasible solution of improved value. As such, PS and Benders decomposition naturally complement each other, in particular when the emphasis is on seeking heuristic solutions. In this paper, we investigate the use of PS as a tactical tool to drive Benders decomposition, and computationally evaluate its performance as a heuristic on instances of different stochastic programming problems.

The main contribution of our work is the development of a method that is able to produce good-quality feasible solutions to large-scale instances of (certain types of) stochastic programming problems, for which standard MILP solvers are unable to do so either because the instances are too big to load into memory or the time to solve even the root node relaxation is prohibitive.

The remainder of the paper is organized as follows. In Section 2, we introduce *Proximity Benders*, a variant of Proximity Search that is specifically designed to handle MILPs arising in stochastic programming. In Section 3, we present a heuristic that can enhance the performance of Proximity Benders. In Section 4, we discuss the results of a set of computational experiments that demonstrate the efficacy of Proximity Benders on instances of three stochastic programming problems. We conclude, in Section 5, with some final remarks and future research directions.

2 Proximity Benders

In what follows, we will concentrate on a MILP of the form:

$$
\begin{array}{llllll}
(P) & \min c_0^T y & + \sum_{k=1}^K \mu_k & & & \\
& & c_k^T x_k & = \mu_k & & (4)\\
& A_0 y & + A_k x_k & \geq b_k & k = 1, \ldots, K & (5)\\
& y & & \in Y & & (6)\\
& & x_k & \in P_k & k = 1, \ldots, K & (7)
\end{array}
$$

where $Y = P_0 \cap \{0,1\}^m$ and each set P_k $(k = 0, \ldots, K)$ is a polyhedron. In other words, we assume that problem P has a block structure allowing for a decomposition into a master problem that involves binary variables y along with K continuous variables μ_k, plus K arising after fixing y, each being a Linear Program (LP) on the (x_k, μ_k) variables. In addition, we assume that it is not hard to determine a feasible (sub-optimal) solution of P. MILPs of this type frequently arise in stochastic programming, and are typically attacked by Benders decomposition.

We next describe our PS heuristic scheme for problem P. Given a feasible solution $(\widetilde{y}, \widetilde{x}, \widetilde{\mu})$, PS adds the cutoff constraint

$$c_0^T y + \sum_{k=1}^{K} \mu_k \leq U - \theta$$

to P, where $U = c_0^T \widetilde{y} + \sum_{k=1}^K \widetilde{\mu}_k$ and $\theta > 0$ is a given tolerance, and replaces the objective function with the Hamming distance

$$\Delta(y, \widetilde{y}) = \sum_{j:\widetilde{y}_j=0} y_j + \sum_{j:\widetilde{y}_j=1} (1 - y_j) \tag{8}$$

with respect to $\widetilde{y}$. In our implementation, we used the "proximity search with incumbent" the cutoff constraint is imposed in a soft way by introducing a new continuous variable $z_0 \geq 0$ with a large cost $M_0 \gg 0$ in the objective function (8), along with the constraint

$$c_0^T y + \sum_{k=1}^{K} \mu_k \leq U - \theta\,(1 - z_0) \tag{9}$$

At any stage of the algorithm, we have a collection of previously-generated Benders cuts involving the master variables, which we enforce by requiring that $(y, \mu) \in \Gamma$, where Γ is the polyhedron defined by the current collection of cuts (initially, $\Gamma = \Re^{m+K}$). Therefore, the master problem has the form,

$$(P_M(\widetilde{y}, U, \Gamma)) \qquad \min\{\Delta(y, \widetilde{y}) + M_0 z_0 : c_0^T y + \sum_{k=1}^{K} \mu_k \leq U - \theta\,(1 - z_0),\ y \in Y,\ (y, \mu) \in \Gamma\}, \tag{10}$$

and can be solved by any black-box MILP solver.

According to classical Benders' decomposition, given an optimal solution of the master problem, say $(\overline{y}, \overline{\mu})$, one solves each ,

$$(P_k(\overline{y}, \overline{\mu}_k)) \qquad \min\{c_k^T x_k :\ A_k x_k \geq b_k - A_0 \overline{y},\ x_k \in P_k\}, \tag{11}$$

independently to possibly derive violated cuts to be added to the master. In our implementation, we followed the cut-generation recipe described in Fischetti et al. [11], i.e., for each k, we introduce two additional nonnegative variables z_k and ν_k and rewrite the corresponding as

$$
\begin{array}{lll}
(\widetilde{P}_k(\overline{y}, \overline{\mu}_k)) \qquad & \min\{M_k z_k + \nu_k : & A_k x_k + \mathbf{1} z_k \geq b_k - A_0 \overline{y}, \quad , \\
& & c_k^T x_k - \nu_k \leq \overline{\mu}_k, \\
& & x_k \in P_k,\ z_k \geq 0,\ \nu_k \geq 0\}
\end{array}
\tag{12}
$$

where $M_k \gg 0$ is a sufficiently large penalty used to drive z_k as close to zero as possible and $\mathbf{1}$ is the vector of all ones. If the optimal solution value of $\widetilde{P}_k(\overline{y}, \overline{\mu}_k)$ is strictly positive, a new cut can be derived and added to the master. Specifically, let (x_k^*, z_k^*, ν_k^*) denote the optimal solution found for the k-th

$\widetilde{P}_k(\overline{y}, \overline{\mu}_k)$. If $z_k^* > 0$, problem $P_k(\overline{y}, \overline{\mu}_k)$ is infeasible, and one can derive a Benders' *feasibility cut* of the form

$$\alpha^T y \leq \gamma \tag{13}$$

to add to the master. Otherwise, i.e., $z_k^* = 0$ and $\nu_k^* > 0$, a Benders' *optimality cut* of the form

$$\alpha^T y + \beta \mu_k \leq \gamma \tag{14}$$

can be added to the master.

Obviously, if $z_k^* = 0$ for all k and $c_0^T \overline{y} + \sum_{k=1}^K c_k^T x_k^* < U$, then one can update the incumbent $(\widetilde{y}, \widetilde{x}, \widetilde{\mu})$ and iterate proximity search starting from this new solution. A more formal description of Proximity Benders is given in Algorithm 2.

Algorithm 2 *Proximity Benders*

Use a heuristic to find an initial feasible solution $(\widetilde{y}, \widetilde{x})$ to problem P
Set $U := c_0^T \widetilde{y} + \sum_{k=1}^K c_k^T \widetilde{x}_k$
Initialize $\Gamma := \Re^{m+K}$ (no Benders cuts)

while **do**
  Solve master problem $P_M(\widetilde{y}, U, \Gamma)$ solution $(\overline{y}, \overline{\mu}, \overline{z}_0)$

  for all $k = 1, \ldots, K$ **do**
    Solve $\widetilde{P}_k(\overline{y}, \overline{\mu}_k)$ to get an optimal solution (x_k^*, z_k^*, ν_k^*)
    if $z_k^* > 0$ **then**
      Add a Benders feasibility cut from $\widetilde{P}_k(\overline{y}, \overline{\mu}_k)$ to the description of Γ

    else if $\nu_k^* > 0$ **then**
      Add a Benders optimality cut from $\widetilde{P}_k(\overline{y}, \overline{\mu}_k)$ to the description of Γ

    end if
  end for
  if $z_1^* = \cdots = z_K^* = 0$ and $c_0^T \overline{y} + \sum_{k=1}^K c_k^T x_k^* < U$ **then**
    Update $\widetilde{y} := \overline{y}$
    Update $U := c_0^T \overline{y} + \sum_{k=1}^K c_k^T x_k^*$

  end if
end while

A sequence of passes through the "while" loop in which the incumbent $\widetilde{y}$ is *not* updated is called is a single pass through the "while" loop where the master problem and the K are solved.

The idea of mixing Benders decomposition with local search is not new. In particular, Rei et al. [14] use local branching to produce a pool of (possibly infeasible) integer solutions of the master problem, from which diversified Benders cuts can be derived. However, our approach reverses the role of local search and Benders decomposition: instead of using local search inside a Benders method, we apply a black-box Benders solver within an external local search scheme, i.e., PS. In doing so, we modify the objective function of the master problem, thus drastically changing the sequence of points provided by the various masters, as well as their associated Benders cuts.

3 A repair heuristic

Due to its particular structure, finding a heuristic solution to problem P involves deciding the values of the y variables, as the x and μ variables can easily be computed by solving a sequence of LP , provided of course that these subproblems are feasible.

We say that $\overline{y}$ is feasible (for problem P) if fixing $y = \overline{y}$ does not produce an infeasible . In many applications, the feasible y's satisfy the following monotonicity property: let $\overline{y} \in \{0,1\}^m$ and $y' \in \{0,1\}^m$ with $y' \geq y$; if $\overline{y}$ is feasible, then y' is also feasible. This implies, in particular, that $\overline{y} = (1, \ldots, 1)$ is always a feasible (though likely very bad) solution.

Assuming monotonicity, one can easily derive a repair heuristic that starts with an infeasible $\overline{y}$, and iteratively increases some of its components until a feasible y' is found. Of course, the quality of the final solution depends on how cleverly the components to be increased are chosen. In this respect, the following *repair heuristic* seems a reasonable option, and is applied in iteration of Algorithm 2 to enforce feasibility of the current $\overline{y}$ when only a few k have $z_k^* > 0$.

Algorithm 3 *Repair Heuristic*

Let $\overline{y}$ be the initial infeasible solution to repair
for all $k = 1, \ldots, K$ **do**
  if $z_k^* > 0$ **then**
    Solve auxiliary problem $(\overline{P}_k(\overline{y}))$ defined by (15) and get a solution (y', x_k)
    Set $\overline{y} = y'$
  end if
end for

4 Computational experiments

The Proximity Benders algorithm presented in Section 2, denoted as `ProxyBenders` in the following, has been implemented in C using IBM-ILOG Cplex 12.6.1 as the MILP solver. We ran the algorithm on different benchmark instances in order to test its effectiveness, namely:

- instances of a stochastic capacitated facility location problem described, for example, by Bodur et al. [5];
- instances of a stochastic network interdiction problems described, for example, by Bodur et al. [5]; and
- instances of a stochastic fixed charge multi-commodity network design problem derived from those introduced by Crainic et al. [7].

To evaluate the performance of Proximity Benders, we also considered alternative approaches and ran, on the same benchmark instances, the following algorithms:

- IBM-ILOG Cplex 12.6.1 in its default settings (`Cplex` in the following) on the MILP associated with an instance; and
-

The minimum improvement required in the cutoff constraint of `ProxyBenders` is set as $\theta = 10^{-5} z_0$, where z_0 is the incumbent solution value. Thus, we use a non-aggressive policy and we let the value of θ decrease during the execution of the algorithm. Each algorithm was run in single-thread mode with a time limit of 1 hour per instance on an Intel Xeon E3-1220V2 running at 3.10 GHz, with 16GB of RAM.

4.1 Metrics

To compare the performance of the different heuristics, we use an indicator recently proposed in Achterberg et al. [1] and Berthold [4], aimed at measuring the trade-off between the computational effort required to produce a solution and the quality of the solution itself. Specifically, let $\widetilde{z}_{opt}$ denote the optimal solution value for a given problem, and $z(t)$ be the value of the best heuristic solution found at a time t. Then, a *primal gap function* p can be computed as

$$p(t) = \begin{cases} 1 & \text{if no incumbent found until time } t \\ \gamma(z(t)) & \text{otherwise} \end{cases}$$

where $\gamma(\cdot) \in [0,1]$ is the *primal gap*, defined as

$$\gamma(z) = \begin{cases} 0 & \text{if } |\widetilde{z}_{opt}| = |z| = 0, \\ 1 & \text{if } \widetilde{z}_{opt} \cdot z < 0, \\ \frac{z - \widetilde{z}_{opt}}{\max\{|\widetilde{z}_{opt}|, |z|\}} & \text{otherwise.} \end{cases}$$

Finally, the *primal integral* (PI) of a run until time $t_{\max}$ is defined as

$$P(t_{\max}) = \int_0^{t_{\max}} p(t)\,\mathrm{d}t \tag{15}$$

and is actually used to measure the quality of primal heuristics: the smaller $P(t_{\max})$, the better the expected quality of the incumbent solution if we stopped computation at an arbitrary time before $t_{\max}$.

We also count the number of instances for which an algorithm produced the best solution at the time limit (#w for number of wins), the number of instances for which a solution is found that improves the initial feasible solution (#i), and the total number of improving solutions found (#s).

4.2 Stochastic Capacitated Facility Location

Our first benchmark includes instances of a stochastic variant of the capacitated facility location problem (CAP), as described by Louveaux [12]. In this variant, first-stage variables determine the set of facilities to be opened before observing the actual realizations of customer demands. The second-stage variables determine the fraction of customer demands allocated to the open facilities. Denoting by I the set of potential facilities, by J the set of customers, and by K the set of scenarios, the problem can be formulated as follows (see Bodur et al. [5] for further details)

$$\begin{aligned} \min \sum_{i \in I} f_i y_i + \frac{1}{|K|} \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} q_{ij} x_{ij}^k & \\ \sum_{i \in I} x_{ij}^k \geq \lambda_j^k & \qquad j \in J; k \in K \\ \sum_{j \in J} x_{ij}^k \leq s_i y_i & \qquad i \in I; k \in K \\ \sum_{i \in I} s_i y_i \geq \max_{k \in K} \sum_{j \in J} \lambda_j^k & \\ y_i \in \{0,1\} & \qquad i \in I \\ x_{ij}^k \geq 0 & \qquad i \in I; j \in J; k \in K \end{aligned} \tag{16}$$

where f_i and s_i represent the fixed cost and capacity, respectively, of facility $i \in I$, λ_j^k is the realized demand of customer $j \in J$ in scenario $k \in K$, and q_{ij} denotes the cost of sending a unit of demand from facility $i \in I$ to customer $j \in J$.

We used all the CAP instances considered in Bodur et al. [5], obtained using networks from the OR-Library (Beasley [2]) and randomly generating the customers' demands. In particular, we have 4 classes each with 4 instances with 250 scenarios and 4 classes each with 4 instances with 500 scenarios.

Table 1 reports, for each algorithm and for different time limits (namely, 100 seconds, 600 seconds and 1 hour), the outcome of our experiments for each instance class. Instances are grouped as in Bodur et al. [5] according to (K, CAP #), thus each row refers to 4 instances. Note that for all these instances, the optimal solution value is known, allowing for an exact computation of the primal integral,

The results in Table 1 show that `ProxyBenders Cplex` for small time limits, but that it loses its edge when more computation time is available. This is due to the fact that these instances are not extremely hard and that all but two of them can be solved to optimality by `Cplex` within the 1-hour time limit.

The results above are encouraging and demonstrate the potential of Proximity Benders, but the instances are inadequate to fully reveal the power of Proximity Benders. As mentioned, most of the instances can be solved to optimality by Cplex in less than one hour. That is not the setting for which is designed. Proximity Benders is designed to be used in settings where the instances are large, difficult, and cannot be solved in a reasonable amount of time by providing the MILP formulation to a solver, indeed for which solving the root relaxation may already be computationally prohibitive. In the next two

Instances		Time limit = 100 sec									Time limit = 600 sec									Time limit = 3,600 sec								
		Cplex			Benders			ProxyBenders			Cplex			Benders			ProxyBenders			Cplex			Benders			ProxyBenders		
		PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s
250	101-104	0.39	4	4/5	2.30	0	2/2	0.85	2	4/17	0.39	4	4/5	13.36	0	2/2	2.84	0	4/20	0.39	4	4/5	78.29	0	3/3	7.91	1	4/22
	111-114	4.95	0	0/0	4.95	0	0/0	2.09	1	4/19	22.67	2	2/6	29.73	0	0/0	5.70	0	4/27	26.67	4	4/8	178.36	0	0/0	16.27	1	4/35
	121-124	5.19	2	2/2	6.31	0	0/0	3.15	2	4/26	18.22	3	3/7	37.88	0	0/0	10.89	1	4/36	22.31	4	4/9	227.28	0	0/0	37.19	1	4/47
	131-134	2.87	1	1/2	3.95	1	1/1	2.78	2	4/17	6.65	4	4/5	22.57	0	2/2	9.06	0	4/29	6.65	4	4/5	131.83	0	2/2	29.99	0	4/32
summary		3.35	7	7/9	4.38	1	3/3	2.22	5	16/79	11.98	13	13/23	25.89	0	4/4	7.12	1	16/112	14.00	16	16/27	153.94	0	5/5	22.84	3	16/136
500	101-104	1.23	2	2/3	2.55	0	0/0	1.35	2	4/20	1.33	4	4/5	15.09	0	1/1	2.64	1	4/25	1.33	4	4/5	89.61	0	1/1	7.15	3	4/27
	111-114	5.12	0	0/0	5.08	0	1/1	2.55	1	4/22	29.07	0	1/1	30.15	0	1/1	11.19	1	4/26	65.63	3	4/8	180.58	0	1/1	53.43	1	4/40
	121-124	5.22	1	1/1	6.56	0	0/0	4.38	2	4/14	27.42	1	2/4	39.33	0	0/0	18.96	3	4/24	80.20	2	4/6	235.98	0	0/0	56.66	2	4/46
	131-134	4.53	0	0/0	4.53	0	0/0	3.65	4	4/13	17.52	1	1/3	24.39	0	1/1	12.23	3	4/31	38.05	4	4/9	135.34	0	1/1	31.00	0	4/37
summary		4.02	3	3/4	4.68	0	1/1	2.98	9	16/69	18.84	6	8/13	27.24	0	3/3	11.26	8	16/106	46.30	13	16/28	160.38	0	3/3	37.06	6	16/150

Table 1: Results on instances of the Stochastic Capacitated Facility Location Problem.

subsections, we present results on instances that are (somewhat) more appropriate to show the benefits of

4.3 Stochastic Network Interdiction

Our second set of benchmark instances among those described by Bodur et al. [5] and includes instances of the stochastic network interdiction problem (SNIP) described by Pan and Morton [13]. In SNIP one is given a directed graph $G = (N, A)$ and a subset of candidate arcs D on which sensors can be installed, so as to maximize the probability of catching an intruder that traverses some path in the graph. First-stage decisions concern the installation of the sensors. In the second stage, a scenario corresponds to an intruder selecting a path that has minimum probability of being detected when traversing the path from the intruder's origin to his destination. Denoting by K the set of scenarios, the problem can be formulated as follows

$$
\begin{aligned}
\min \sum_{k \in K} p_k x^k_{s^k} & \\
\sum_{(i,j) \in D} c_{ij} y_{ij} \leq b & \\
x^k_{t^k} = 1 & \qquad k \in K \\
x^k_i - q_{ij} x^k_j \geq 0 & \qquad (i,j) \in D; k \in K \\
x^k_i - r_{ij} x^k_j \geq 0 & \qquad (i,j) \in A \setminus D; k \in K \\
x^k_i - r_{ij} x^k_j \geq -(r_{ij} - q_{ij}) \psi^k_j y_{ij} & \qquad (i,j) \in D; k \in K \\
y_{ij} \in \{0,1\} & \qquad (i,j) \in D \\
x^k_i \geq 0 & \qquad i \in N; k \in K
\end{aligned}
$$

where p_k is the probability of scenario k, which corresponds to a path from origin s^k to destination t^k, c_{ij} is the cost of installing a sensor on arc $(i,j) \in D$, b is the available budget, and q_{ij} and r_{ij} denote the probability of failing to detect the intruder with and without a sensor on each arc (i,j), respectively. Finally, coefficients ψ^k_j represent the value of the maximum-reliability path from j to t^k when no sensors are placed and can be determined by shortest-path computation. In this case too, the reader is referred to Bodur et al. [5] for a complete description of objective function and constraints.

We focus our experiments on the more difficult instances, which are obtained by drawing r_{ij} uniform randomly from $[0.3, 0.6]$ and setting $q_{ij} = 0.1 r_{ij}$ (Class 3 instances) and $q_{ij} = 0$ (Class 4 instances) for all $(i,j) \in A$, respectively. The available budget ranges from 30 to 90. Table 2 gives the corresponding results.

In Figure 1, we show the best known solution value for each of the heuristics over time for one of the instances (namely, We see that all algorithms start with the same initial solution of value 0.199, and `ProxyBenders` finds many improving feasible solutions during its execution; the final one after about

4.4 Stochastic Fixed-Charge Multi-Commodity Network Design

Finally, we consider large instances of a stochastic fixed charge multi-commodity network design problem. Given a directed network $G = (N, A)$ and a set K of commodities, the deterministic problem seeks a minimum cost set of arcs that allows the required amount of flow for each commodity to be send from its origin to its destination (i.e., to satisfy demand). In the stochastic version of the problem, first-stage binary variables determine the network design, i.e., the set of arcs to be installed, whereas second-stage continuous variables determine the flow of a commodity along an arc for a given scenario (i.e., for a given

Instances		Time limit = 100 sec									Time limit = 600 sec									Time limit = 3,600 sec								
		`Cplex`			`Benders`			`ProxyBenders`			`Cplex`			`Benders`			`ProxyBenders`			`Cplex`			`Benders`			`ProxyBenders`		
		PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s
3	30	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0
	40	0.74	0	0/0	0.74	0	0/0	0.74	0	0/0	4.43	0	0/0	4.13	0	1/1	4.14	2	2/3	26.55	0	0/0	23.65	1	2/3	12.32	4	4/26
	50	0.53	0	0/0	0.53	0	0/0	0.53	0	0/0	3.18	0	0/0	3.18	0	0/0	3.18	0	0/0	19.08	0	0/0	9.05	1	1/4	5.21	1	1/3
	60	1.23	0	0/0	1.23	0	0/0	1.14	2	2/3	7.36	0	0/0	7.36	0	0/0	6.15	2	2/3	44.17	0	0/0	38.08	2	3/4	20.31	5	5/27
	70	1.76	0	0/0	1.76	0	0/0	1.74	1	1/1	10.58	0	0/0	10.58	0	0/0	10.08	1	1/1	63.46	0	0/0	36.27	2	3/9	21.93	2	3/18
	80	2.15	0	0/0	2.15	0	0/0	2.15	0	0/0	12.88	0	0/0	12.88	0	0/0	12.73	3	3/3	77.27	0	0/0	52.48	0	3/6	23.20	4	4/30
	90	4.10	0	0/0	4.10	0	0/0	4.10	0	0/0	24.60	0	0/0	24.60	0	0/0	24.00	2	2/5	147.59	0	0/0	145.12	0	2/2	101.83	4	4/58
summary		1.50	0	0/0	1.50	0	0/0	1.48	3	3/4	9.00	0	0/0	8.96	0	1/1	8.61	10	10/15	54.02	0	0/0	43.52	6	14/28	26.40	20	21/162
4	30	1.73	0	0/0	1.71	0	1/2	1.49	1	1/2	10.39	0	0/0	7.22	1	1/6	6.70	0	1/16	62.36	0	0/0	7.22	1	1/6	8.93	1	1/31
	40	1.54	0	0/0	1.54	0	0/0	1.50	2	2/3	9.26	0	0/0	9.26	0	0/0	4.68	2	2/18	55.57	0	0/0	47.46	0	1/2	6.36	2	2/25
	50	0.66	0	0/0	0.66	0	0/0	0.66	0	0/0	3.96	0	0/0	3.96	0	0/0	3.12	2	2/4	23.73	0	0/0	14.07	1	1/1	5.67	3	3/11
	60	6.09	0	0/0	5.81	0	1/4	5.07	2	2/7	36.57	0	0/0	23.01	1	1/7	24.02	3	4/22	189.82	0	1/2	84.93	1	3/11	62.10	4	4/50
	70	3.94	0	0/0	3.94	0	0/0	3.81	1	1/3	23.66	0	0/0	23.39	0	1/1	19.26	2	2/14	141.99	0	0/0	97.35	0	2/5	23.15	4	4/37
	80	11.48	0	0/0	11.48	0	0/0	11.44	1	1/1	68.90	0	0/0	68.68	0	1/1	66.67	4	4/9	398.63	0	1/1	357.64	1	3/7	272.03	5	5/37
	90	92.06	0	0/0	92.06	0	0/0	92.05	1	1/2	552.33	0	0/0	552.33	0	0/0	551.45	5	5/21	3314.00	0	0/0	3309.03	2	3/5	3296.69	4	5/23
summary		16.79	0	0/0	16.74	0	2/6	16.57	8	8/18	100.72	0	0/0	98.26	2	4/15	96.56	18	20/104	598.01	0	2/3	559.67	6	14/37	524.99	23	24/214

Table 2: Results on instances of the Stochastic Network Interdiction Problem.

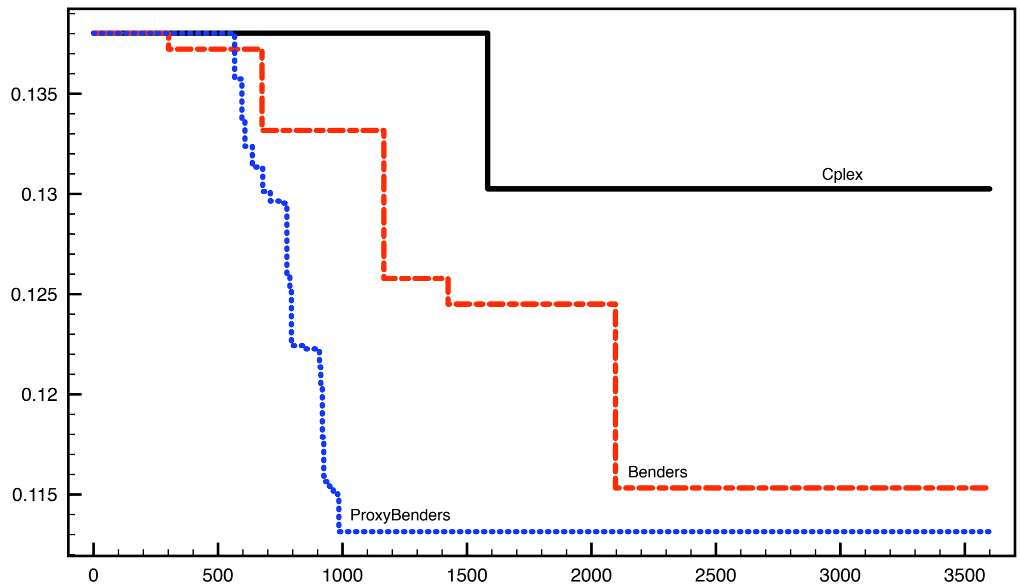

Figure 1: The best known solution value over time for the three heuristics.

demand realization). Denoting by S the set of scenarios, the model reads as follows

$$
\begin{aligned}
\min \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{s\in S} p_s (\sum_{k\in K} \sum_{(i,j)\in A} c^k_{ij} x^{ks}_{ij}) & \\
\sum_{j\in N^+(i)} x^{ks}_{ij} - \sum_{j\in N^-(i)} x^{ks}_{ji} = d^{ks}_i & \qquad i \in N; k \in K; s \in S \\
\sum_{k\in K} x^{ks}_{ij} \le u_{ij} y_{ij} & \qquad (i,j) \in A; s \in S \\
y_{ij} \in \{0,1\} & \qquad (i,j) \in A \\
x^{ks}_{ij} \ge 0 & \qquad (i,j) \in A; k \in K; s \in S,
\end{aligned}
$$

where f_{ij} and u_{ij} denote the fixed cost and capacity, respectively, of arc $(i,j) \in A$, c^k_{ij} denotes the cost of sending one unit of commodity $k \in K$ along arc $(i,j) \in A$, d^{ks}_i denotes the net flow (i.e., the difference between inflow and outflow) at node $i \in N$ for commodity $k \in K$ in scenario $s \in S$, and p_s denotes the probability of scenario $s \in S$. Further details about the model can be found in Crainic et al. [7].

We consider seven of the instance classes (namely, instance classes 4 through 10) used in Crainic et al. [7], which, in turn, were derived from the set of R-instances of Crainic et al. [6]. For each instance class, we consider five networks (namely, networks 1, 3, 5, 7, and 9) with an increasing ratio of fixed to variable costs and total demand to capacity. For each of the 35 networks, Crainic et al. [7] generated 5 instances with 64 scenarios. In order to have instances with a larger number of scenarios, we generate scenarios as follows. For each commodity and for each network, we determine the minimum and maximum demand over all scenarios considered in Crainic et al. [7]. Then, we generate the demand for the commodity in each of the $|S|$ scenarios by drawing uniform randomly from the interval determined by the minimum and maximum demand.

Table 3 provides results for instances with $2,000$ scenarios. The table reports the same information as Tables 1 and 2, but each line now refers to a specific instance. Because for many instances the optimal solution value is not known, the primal integral has been computed with respect to the best known solution value.

For instances, none of the heuristics was able to improve on the initial feasible solution in one hour. For these instances all entries in the table are set to zero. For twenty instances, `Cplex` was unable to even solve the LP relaxation in one hour. These instances are marked by a star.

Instance		Time limit = 100 sec									Time limit = 600 sec									Time limit = 3,600 sec								
		Cplex			Benders			ProxyBenders			Cplex			Benders			ProxyBenders			Cplex			Benders			ProxyBenders		
		PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s	PI	#w	#i/#s
4	1	5.23	0	0/0	5.23	0	0/0	4.77	1	1/1	23.85	1	1/1	31.39	0	0/0	26.75	0	1/3	24.30	1	1/2	188.35	0	0/0	113.92	0	1/3
4	3	1.48	0	0/0	1.48	0	0/0	1.48	0	0/0	8.90	0	0/0	8.90	0	0/0	8.90	0	0/0	23.79	0	1/1	53.41	0	0/0	9.75	1	1/1
4	5	2.43	0	0/0	2.43	0	0/0	2.43	0	0/0	14.60	0	0/0	14.60	0	0/0	14.60	0	0/0	81.74	1	1/1	87.57	0	0/0	87.57	0	1/0
4	7	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	1	1/1	0.00	0	1/0
4	9	0.37	0	0/0	0.37	0	0/0	0.37	0	0/0	2.23	0	0/0	2.23	0	0/0	0.79	1	1/1	13.40	0	0/0	13.40	0	0/0	0.79	1	1/1
5	1	22.44	0	0/0	22.44	0	0/0	22.44	0	0/0	62.38	1	1/1	134.67	0	0/0	121.74	0	1/5	98.68	1	1/3	808.00	0	0/0	417.11	0	1/1
5	*3	13.37	0	0/0	13.37	0	0/0	13.37	0	0/0	80.19	0	0/0	80.19	0	0/0	76.94	1	1/3	481.17	0	0/0	481.17	0	0/0	214.82	1	1/7
5	*5	19.07	0	0/0	19.07	0	0/0	18.92	1	1/1	114.43	0	0/0	114.43	0	0/0	97.03	1	1/3	686.56	0	0/0	686.56	0	0/0	370.66	1	1/7
5	7	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	1	1/1	0.00	0	1/0
5	*9	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	1	1/1	0.00	0	1/0	0.00	0	0/0	0.00	1	1/2	0.00	0	1/0
6	*1	19.70	0	0/0	19.70	0	0/0	19.70	0	0/0	118.17	0	0/0	118.17	0	0/0	98.78	1	1/2	709.03	0	0/0	709.03	0	0/0	311.57	1	1/8
6	*3	8.15	0	0/0	8.15	0	0/0	8.15	0	0/0	48.90	0	0/0	48.90	0	0/0	48.90	0	0/0	293.39	0	0/0	293.39	0	0/0	239.41	1	1/3
6	*5	10.01	0	0/0	10.01	0	0/0	10.01	0	0/0	60.08	0	0/0	60.08	0	0/0	55.83	1	1/1	360.49	0	0/0	360.49	0	0/0	140.78	1	1/9
6	7	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	1	1/2	0.00	0	1/0
6	*9	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0
7	1	10.21	1	1/1	27.77	0	0/0	27.38	0	1/1	10.52	1	1/2	166.63	0	0/0	84.93	0	1/10	10.52	1	1/2	994.25	0	1/1	88.29	0	1/11
7	3	11.13	0	0/0	11.13	0	0/0	11.13	0	0/0	66.77	0	0/0	66.77	0	0/0	66.77	0	0/0	303.91	0	1/1	400.64	0	0/0	296.79	1	1/4
7	5	14.55	0	0/0	14.55	0	0/0	14.14	1	1/2	87.30	0	0/0	87.30	0	0/0	80.87	1	1/3	195.00	0	1/1	523.82	0	0/0	176.22	1	1/10
7	7	4.04	0	0/0	4.04	0	0/0	2.91	1	1/2	24.22	0	0/0	24.22	0	0/0	13.70	1	1/2	145.32	0	0/0	145.32	0	0/0	55.48	1	1/9
7	*9	2.50	0	0/0	2.50	0	0/0	1.45	1	1/2	15.02	0	0/0	15.02	0	0/0	1.45	1	1/2	90.13	0	0/0	90.13	0	0/0	1.45	1	1/2
8	1	20.62	0	0/0	20.62	0	0/0	20.62	0	0/0	46.54	1	1/1	123.70	0	0/0	90.71	0	1/5	110.25	0	1/1	742.20	0	0/0	219.68	1	1/10
8	*3	6.16	0	0/0	6.16	0	0/0	6.16	0	0/0	36.94	0	0/0	36.94	0	0/0	24.30	1	1/1	221.62	0	0/0	221.62	0	0/0	82.76	1	1/2
8	*5	9.32	0	0/0	9.32	0	0/0	9.32	0	0/0	55.92	0	0/0	55.92	0	0/0	50.25	1	1/2	335.53	0	0/0	335.53	0	0/0	280.18	1	1/5
8	*7	8.82	0	0/0	8.82	0	0/0	8.67	1	1/3	52.89	0	0/0	52.89	0	0/0	33.58	1	1/12	317.35	0	0/0	235.90	1	1/6	80.05	0	1/17
8	*9	3.64	0	0/0	3.64	0	0/0	3.64	0	0/0	21.83	0	0/0	21.83	0	0/0	21.83	0	0/0	131.01	0	0/0	131.01	0	0/0	38.24	1	1/4
9	1	44.31	0	0/0	44.31	0	0/0	44.31	0	0/0	265.85	0	0/0	265.85	0	0/0	230.87	1	1/3	1544.0	1	1/1	1595.09	0	0/0	1043.82	0	1/11
9	*3	13.37	0	0/0	13.37	0	0/0	13.37	0	0/0	80.24	0	0/0	80.24	0	0/0	79.14	1	1/3	481.44	0	0/0	481.44	0	0/0	417.91	1	1/6
9	*5	12.54	0	0/0	12.54	0	0/0	12.54	0	0/0	75.23	0	0/0	75.23	0	0/0	67.96	1	1/1	451.41	0	0/0	451.41	0	0/0	264.83	1	1/8
9	7	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	1	1/1	0.00	0	1/0
9	*9	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	1	1/2	0.00	0	1/0
10	*1	12.27	0	0/0	12.27	0	0/0	12.27	0	0/0	73.63	0	0/0	73.63	0	0/0	73.63	0	0/0	441.81	0	0/0	441.81	0	0/0	224.93	1	1/5
10	*3	0.46	0	0/0	0.46	0	0/0	0.46	0	0/0	2.76	0	0/0	2.76	0	0/0	2.76	0	0/0	16.57	0	0/0	16.57	0	0/0	8.59	1	1/2
10	*5	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0	0.00	0	0/0
10	*7	16.59	0	0/0	16.59	0	0/0	16.59	0	0/0	99.54	0	0/0	99.54	0	0/0	99.54	0	0/0	597.27	0	0/0	597.27	0	0/0	460.63	1	1/8
10	*9	3.35	0	0/0	3.35	0	0/0	3.35	0	0/0	20.10	0	0/0	20.10	0	0/0	20.10	0	0/0	120.63	0	0/0	120.63	0	0/0	86.11	1	1/5
summary		8.46	1	1/1	8.96	0	0/0	8.86	6	7/12	44.83	4	4/5	53.78	1	1/1	45.50	14	19/62	236.75	5	9/13	320.17	7	8/16	163.78	21	33/172

Table 3: Results on instances of the Stochastic Fixed-Charge Multi-Commodity Network Design Problem.

We see that On the other hand, ProxyBenders is reasonably effective, especially for the largest instance classes (namely, instance classes 8, 9 and 10). When restricting ourselves to these instance classes, we see that Cplex finds improving solutions for only . Although ProxyBenders is not guaranteed to find a feasible solution at each iteration, using a large value M_k in problem $\tilde{P}_k(\overline{y}, \overline{\mu}_k)$ enforces feasibility in many cases. And, whenever this is not the case, the repair heuristic recovers feasibility, which may or may not provide an improving solution.

These result reinforce that Proximity Benders should be the method of choice in situations where high-quality solutions to very large instances of stochastic programming problems need to be found quickly.

5 Final remarks

The computational results presented in this paper provide a proof-of-concept demonstration of the potential of Proximity Benders. The computational results suggest that Proximity Benders should be the method of choice in situations where high-quality solutions to very large instances of certain types of stochastic programming problems need to be found quickly. This is true, in particular, for situations in which solving the LP relaxation is already computationally prohibitive. Proximity Benders naturally and easily parallelizes, which will further amplify its advantages and benefits.

We next mention just a few possible directions for future research. Investigating the sensitivity of Proximity Benders to the quality of the initial feasible solution is of interest. We have used a non-aggressive policy for setting the minimum improvement required in the cutoff constraint of Proximity Benders. Investigating the performance of Proximity Benders for different (more aggressive) schemes for setting and updating the minimum improvement required is of interest. Exploring whether the efficiency of Proximity Benders can be improved by incorporating more sophisticated cut management schemes or by not solving all in each iteration is also of interest. There are similarities between Proximity Benders and the Feasibility Pump for finding feasible solutions to MILP (Fischetti et al. [8]). The use of randomization is critical to the performance of the feasibility pump and, therefore, randomization schemes for Proximity Benders, e.g., randomizing $\tilde{y}$, are worth studying.

Finally, we observe that our proof-of-concept implementation did not target a specific problem, and we anticipate that tailored implementations for specific problems would likely be far more efficient. Thus, we expect that Proximity Benders can lead to quite effective ad-hoc heuristics for very large stochastic programming applications.

Acknowledgments

This research of Fischetti and Monaci was supported by the University of Padova (Progetto di Ateneo "Exploiting randomness in Mixed Integer Linear Programming"), and by MiUR, Italy (PRIN project "Mixed-Integer Nonlinear Optimization: Approaches and Applications"). We thank Merve Bodur and Mike Hewitt for their support and for providing the data used in the computational analysis.

References

[1] Achterberg, T., Berthold, T., and Hendel, G. (2012). Rounding and propagation heuristics for mixed integer programming. *Operations Research Proceedings 2011*, pages 71–76.

[2] Beasley, J. (1990). Or-library: Distributing test problems by electronic mail. *Journal of Operational Research Society*, 41:1069–1072.

[3] Benders, J. (1962-63). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

[4] Berthold, T. (2013). Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614.

[5] Bodur, M., Dash, S., Günlück, O., and Luedtke, J. (2014). Strengthened Benders cuts for stochastic integer programs with continuous recourse. Technical report. available as Optimization Online 2014-03-4263.

[6] Crainic, T., Fu, X., Gendreau, M., Rei, W., and Wallace, S. (2011). Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58(2):114–124.

[7] Crainic, T., Hewitt, M., and Rei, W. (2014). Partial decomposition strategies for two-stage stochastic integer programs. Technical Report 13, CIRRELT.

[8] Fischetti, M., Glover, F., and Lodi, A. (2005). The feasibility pump. *Mathematical Programming*, 104(1):91–104.

[9] Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, pages 23–47.

[10] Fischetti, M. and Monaci, M. (2014). Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731.

[11] Fischetti, M., Salvagnin, D., and Zanette, A. (2010). A note on the selection of Benders cuts. *Mathematical Programming*, 124(1-2):175–182.

[12] Louveaux, L. (1986). Discrete stochastic location models. *Annals of Operations Research*, 6:23–34.

[13] Pan, F. and Morton, D. (2008). Minimizing a stochastic maximum-reliability path. *Networks*, 52(3):111–119.

[14] Rei, W., Cordeau, J.-F., Gendreau, M., and Soriano, P. (2009). Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345.

[15] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin Heidelberg.