A scenario decomposition algorithm for 0-1 stochastic programs

Shabbir Ahmed
School of Industrial & Systems Engineering,
Georgia Institute of Technology, 765 Ferst Drive, Atlanta, GA 30332.

June 26, 2013

Abstract

We propose a scenario decomposition algorithm for stochastic 0-1 programs. The algorithm recovers an optimal solution by iteratively exploring and cutting-off candidate solutions obtained from solving scenario subproblems. The scheme is applicable to quite general problem structures and can be implemented in a distributed framework. Illustrative computational results on standard two-stage stochastic integer programming and nonlinear stochastic integer programming test problems are presented.

1 Introduction

We consider stochastic programs of the following form

$$\min\{\mathbb{E}[f(x,\boldsymbol{\xi})]:\ x\in X\subseteq\{0,1\}^n\}, \tag{1}$$

where $\boldsymbol{\xi}$ is a random vector with support Ξ and known distribution P, and the expectation in (1) is with respect to P. An important example of (1) is the class of two-stage stochastic programs with 0-1 first stage variables with

$$f(x,\xi)=c^\top x+\min\{\phi(y(\xi),\xi):\ y(\xi)\in Y(\xi,x)\}$$

where, for realization ξ of $\boldsymbol{\xi}$, $y(\xi)$ is the second stage decision vector, $\phi(\cdot,\xi)$ is the second stage objective function, and $Y(\xi,x)$ is the second stage constraint system depending on the first stage decision vector x. We assume that the random vector $\boldsymbol{\xi}$ has a finite support, i.e. $\Xi=\{\xi^1,\ldots,\xi^N\}$, where each ξ^i for $i\in\{1,\ldots,N\}$ is referred to as a scenario. We can then rewrite (1) as

$$\min\left\{\sum_{i=1}^N f_i(x):\ x\in X\subseteq\{0,1\}^n\right\}, \tag{2}$$

where $f_i(x)=p_i f(x,\xi^i)$ and p_i is the probability mass associated with scenario i.

A popular approach for solving (2) is the so-called scenario or dual decomposition method. By making copies of the decision variables x, problem (2) can be reformulated as

$$\min\left\{\sum_{i=1}^N f_i(x^i):\ x^i\in X\ \forall\, i,\sum_{i=1}^N A_i x^i=h\right\}$$

where the equations $\sum_{i=1}^N A_i x^i=h$ enforce the nonanticipativity constraints $x^1=\cdots=x^N$. The Lagrangian dual problem by dualizing these nonanticipativity constraints take the form

$$\max_\lambda\left\{v(\lambda):=\sum_{i=1}^N\min\left\{f_i(x^i)+\lambda^\top A_i x^i:\ x^i\in X\right\}-\lambda^\top h\right\}, \tag{3}$$

where λ is the dual vector. For any λ the value $v(\lambda)$ provides a lower bound to (2) which can be evaluated by separately solving subproblems corresponding to each scenario. The best possible such bound is given by the optimal value of the dual problem (3) which is a large-scale and nonsmooth, albeit convex, optimization problem and is in general quite challenging. Even with an optimal dual solution, owing to the presence of integer decision variables, there typically remains a duality gap, and moreover a primal feasible solution (one that satisfies the original and the nonanticipativity constraints) is not readily available. Caroe and Schultz [5] proposed a branch and bound algorithm where the dual problem (3) is used to generate lower bounds, the integer feasible solutions to the scenario subproblems are "averaged" to generate a possibly fractional primal solutions, and the feasible region is successively partitioned by branching on the fractional variables to enforce integrality. Alonso-Ayuso et.al. [2] propose solving scenario subproblems (with $\lambda = 0$) with a branch-and-fix approach that coordinates the selection of the branching nodes and branching variables in order to enforce the nonanticipativity constraints. A number of heuristics [6, 9, 15] have also been designed based on the scenario-wise decomposability of the dual problem (3). In these approaches the dual problem is augmented by an additional penalty to achieve consensus among the scenario subproblem solutions, and the penalty parameters are iteratively updated to try to achieve primal feasible solutions. No convergence guarantees are available for such approaches, but they are reported to have good performance in various applications.

In this paper we propose a scenario decomposition algorithm for (2) that proceeds by solving scenario subproblems to generate candidate solutions and lower bounds, evaluating candidate solutions to get upper bounds, and cutting off candidate solutions from the scenario subproblems to get improved lower bounds and new candidate solutions. The scheme is applicable for quite general problem structures and, since unlike existing exact scenario decomposition methods requires little coordination among scenario subproblems, can be implemented easily in a distributed framework. The remainder of the paper is organized as follows: in Section 2 we describe the proposed scenario decomposition method along with some implementation issues, in Section 3 we discuss some justification for exploring solutions to scenario subproblems as candidate solutions for the overall problem, and finally in Section 4 we present some computational results using a distributed implementation of the proposed method on standard two-stage stochastic integer programming and nonlinear stochastic integer programming test problems.

2 A Scenario Decomposition Algorithm

We propose a simple scenario decomposition algorithm that exploits the scenario-wise decomposability of (3) and the 0-1 nature of the variables. The algorithm explores solutions to the scenario subproblems as candidate primal feasible solutions to the overall problem. The explored solutions are then cut-off from future consideration in all subproblems. This allows for an improvement in the lower bound by restricting the feasible region and eventually closing the duality gap.

The overall scheme is outlined in Algorithm 1. In the lower bounding step (lines 3-12), the scheme uses the decomposable dual problem (3) to generate lower bounds, and a set of candidate solutions. Note that we have not exactly specified how to update the dual solutions λ since the general scheme is independent of this. In fact, the scheme is valid without updating λ at all. In the upper bounding step (lines 13-22) each of the candidate solutions is evaluated and the best of these provide an upper bound and becomes the incumbent solution x^*. A key feature of the scheme is that the evaluated set of solutions is discarded from the set of feasible solutions (note the set $X \setminus S$ in line 7). Because of the 0-1 nature of the solutions this can be easily accomplished by adding "integer cuts" of the form

$$\sum_{j:\widehat{x}_j=1} (1 - x_j) + \sum_{j:\widehat{x}_j=0} x_j \geq 1 \;\; \forall\, \widehat{x} \in S$$

to the original constraints X of the problem. Recently stronger formulations for discarding a given set of 0-1 solutions have been proposed which can also be used in this context [3].

Proposition 1 *Assuming that solving a scenario subproblem (step 7) and evaluating the objective function (step 17) requires finite time, Algorithm 1 (scenario decomposition) terminates in a finite number of iterations returning an optimal solution or detecting infeasibility.*

Algorithm 1 Scenario Decomposition

$UB \leftarrow +\infty$, $LB \leftarrow -\infty$, $S = \emptyset$, $x^* \leftarrow \emptyset$
while $UB > LB$ **do**
  `Lower bounding:`
  $\lambda \leftarrow 0$
  while some termination criteria is not met **do**
    for $i = 1$ to N **do**
      solve $\min\{f_i(x) + \lambda^\top A_i x : \ x \in X \setminus S\}$
      let v_i be the optimal value and x^i be an optimal solution
    end for
    update λ
  end while
  $LB \leftarrow \sum_{i=1}^N v_i - \lambda^\top h$, $\widehat{S} = \cup_{i=1}^N \{x^i\}$ and $S \leftarrow S \cup \widehat{S}$
  `Upper bounding:`
  for $x \in \widehat{S}$ **do**
    $u \leftarrow 0$
    for $i = 1$ to N **do**
      compute $f_i(x)$ and set $u \leftarrow u + f_i(x)$
    end for
    if $UB > u$ **then**
      $UB \leftarrow u$ and $x^* \leftarrow x$
    end if
  end for
end while

Proof: If the problem is infeasible the lower bound is set to $+\infty$ and the algorithm terminates after the first iteration. Otherwise, finite termination of the algorithm follows from the fact that the feasible region gets strictly smaller in each iteration and so the lower bound is nondecreasing (the lower bound is $+\infty$ when the feasible region becomes empty). Moreover since no solution is ever discarded without evaluation, it follows from the finiteness of the solution set that the algorithm returns an optimal solution in a finite number of iterations. □

The proposed scenario decomposition algorithm presumes that we have exact oracles to solve scenario subproblems of the form:

$$\min\{f_i(x) + \lambda^\top A_i x : \ x \in X \setminus S\}$$

for any λ and S, and to evaluate the objective function $f_i(x)$ for a given solution. These are similar to the assumptions made for other decomposition schemes based on the dual problem (3). For two-stage stochastic integer programs these steps require solving many single scenario integer programs and can present significant bottleneck. Of course instead of exact computations, we can compute the scenario subproblems and objective function to some pre-specified tolerance, use the best lower bound in the lower bounding step and the best upper bound in the upper bounding step, and terminate the algorithm after the lower and upper bounds are within tolerance. Also, note that in the upper bounding step when evaluating a solution $\widehat{x}$ we do not need to consider the scenarios for which this solution was optimal in the lower bounding step since the corresponding objective function values are already available. Significant computational benefits can be obtained by using relaxations of the scenario subproblems. Let $\underline{f}_i(x)$ be a relaxation of $f_i(x)$, i.e. $\underline{f}_i(x) \leq f_i(x)$ for all $x \in X$. For two-stage stochastic integer programs such a relaxation could be obtained by relaxing the integrality requirements on the second-stage variables. It is valid to use $\underline{f}_i(x)$ in the lower bounding step. The upper bounding step can be modified as follows: for a given candidate solution $\widehat{x}$ first evaluate $\underline{u} := \sum_{i=1}^N \underline{f}_i(\widehat{x})$ and if $\underline{u} \geq UB$ then we do not need to further consider $\widehat{x}$ since a better solution is already known; on the other hand if $\underline{u} < UB$ then we need to evaluate $u = \sum_{i=1}^N f_i(\widehat{x})$ as usual before

$\widehat{x}$ can be cut-off from further consideration. We could also evaluate and discard only a small subset of the solutions found rather than considering all the scenario subproblem solutions $\widehat{S}$. This would reduce the effort within a main iteration, perhaps at the expense increasing the number of main iterations.

An attractive feature of the proposed scheme is that there is very little interaction between the scenario subproblems in the lower and upper bounding steps. Accordingly the scheme can be easily implemented in a distributed framework. Each processor can be assigned a subset of scenarios and would essentially run through Algorithm 1 independently except for broadcasting and collecting solutions and objective values after solving scenario subproblems in step 7 and after evaluating objective functions in step 17. Note that if a subgradient algorithm (cf. [4]) is used to update the dual solutions λ where the update only depends on the scenario subproblem solutions and objective values, then each processor can locally update their copies of the dual solutions after exchanging solutions to the scenario subproblems in step 7, and these local dual solutions will be identical. In Section 4 we provide computational results using such a distributed implementation.

3 Optimality of scenario solutions

The algorithm proposed in the previous section explores solutions to the scenario subproblems as candidates for solutions to the overall problem. This is a departure from other methods based on solving the dual problem (3) where candidate solutions are generated by aggregating solutions to the scenario subproblems. In this section we attempt to provide some rationale why the solutions to the scenario subproblems themselves can be good solutions to the overall problem in the case the problem of interest is a sample average approximation problem.

Consider a 0-1 stochastic program

$$(P): \ \min \{\mathbb{E}[f(x, \boldsymbol{\xi})] : \ x \in X \subseteq \{0,1\}^n\},$$

and its sample average approximation

$$(SAA_N): \ \min \left\{ (1/N) \sum_{i=1}^{N} f(x, \xi^i) : \ x \in X \subseteq \{0,1\}^n \right\},$$

corresponding to an iid sample $\{\xi^i\}_{i=1}^N$. The scenario subproblem corresponding to the i-th scenario/sample is

$$(P_i): \ \min \left\{ f(x, \xi^i) : \ x \in X \subseteq \{0,1\}^n \right\}.$$

Let S_N be the set of optimal solutions of (SAA_N), and S_i be the set of optimal solutions of (P_i) for $i = 1, \ldots, N$. Note that these are random sets since they depend on the sample drawn. We investigate the following question:

> What is the probability that the set of solutions to one of the scenario problems (P_i) contain a solution to (SAA_N)?

More precisely we would like to estimate $\Pr[S_N \cap (\bigcup_{i=1}^N S_i) \neq \emptyset]$ and understand its dependence on X and N.

Trivially, if $|X| \leq 2$ and $N \geq 2$ then the above probability is 1. On the other hand, it is not hard to construct examples where this probability is arbitrarily small. For example, suppose $X = \{x^1, x^2, x^3\}$, $\boldsymbol{\xi}$ has two realizations ξ^1 and ξ^2 with equal probability, and the values of $f(x, \xi)$ are as in Table 1. Then for N reasonably large $S_N = \{x^1\}$ with very high probability while $\bigcup_{i=1}^N S_i = \{x^2, x^3\}$ with probability one.

In the following we show that if the collection of random objective functions $\{f(x, \boldsymbol{\xi})\}_{x \in X}$ of (P) is jointly normal with a nonnegative and strictly diagonally dominant covariance matrix then

$$\lim_{N \to \infty} \Pr \left[S_N \cap (\bigcup_{i=1}^{N} S_i) \neq \emptyset \right] = 1$$

Solution	ξ^1	ξ^2
x^1	0	0
x^2	-1	2
x^3	2	-1

Table 1: Values of $f(x, \xi)$ for an example

exponentially fast. Before proceeding with a proof of the above result we make some remarks. The condition of nonnegative covariance matrix implies that the objective functions at different solutions are nonnegatively correlated, and therefore, perhaps it is not very surprising that a scenario solution is likely to be optimal for the sample average problem. Recall that a matrix is strictly diagonally dominant if the diagonal element in a row is greater than the sum of the absolute values of all other elements in that row. This condition can be ensured by adding independent zero-mean normal noise to the objective function value for each solution without changing the optimal solution or optimal value of (P). We will need the following Lemma.

Lemma 1 *Let W be a $K+1$ dimensional Gaussian random vector, $W \sim N(\mu, \Sigma)$ with the following properties:*

1. *For some $\epsilon \geq 0$, $\mu_0 \leq \mu_k + \epsilon$ for all $k = 1, \ldots, K$ and*
2. *the covariance matrix Σ is nonnegative and strictly diagonally dominant.*

Condition 2 implies that there exists δ such that $0 < \delta^2 := \min_{k \neq l}\{\sigma_k^2 - \sigma_{kl}, \sigma_l^2 - \sigma_{kl}\}$. Then

$$\Pr[W_0 \leq W_k \ \forall k = 1, \ldots, K] \geq (\Phi(-\sqrt{2}\epsilon/\delta))^K,$$

where Φ is the standard normal cdf.

Proof: Since $\mu_0 \leq \mu_k + \epsilon$ for all $k = 1, \ldots, K$, we have

$$\Pr[W_0 \leq W_k \ \forall k = 1, \ldots, K] \quad \geq \quad \Pr[(W_0 - \mu_0) - (W_k - \mu_k) \leq -\epsilon \ \forall k = 1, \ldots, K].$$

Let $U_k := (W_0 - \mu_0) - (W_k - \mu_k)$ for $k = 1, \ldots, K$, and note that U is a K dimensional Gaussian random vector with $\mathbb{E}[U_k] = 0$, $\mathbb{E}[U_k^2] = \sigma_0^2 + \sigma_k^2 - 2\sigma_{0k} \geq 2\delta^2$, and $\mathbb{E}[U_k U_l] = \sigma_0^2 - \sigma_{0l} - \sigma_{0k} + \sigma_{kl} \geq 0$ for all $k, l = 1, \ldots, K$. Consider another K dimensional Gaussian random vector V with $\mathbb{E}[V_k] = 0$, $\mathbb{E}[V_k^2] = \mathbb{E}[U_k^2]$ and $\mathbb{E}[V_k V_l] = 0$ for all $k, l = 1, \ldots, K$. Since $\mathbb{E}[V_k V_l] \leq \mathbb{E}[U_k U_l]$ for all $k, l = 1, \ldots, K$, by Slepian's inequality [14]

$$\Pr[W_0 \leq W_k \ \forall k] \geq \Pr[U_k \leq -\epsilon \ \forall k] \geq \Pr[V_k \leq -\epsilon \ \forall k] = \prod_{k=1}^{K} \Pr[V_k \leq -\epsilon],$$

where the last identity follows from independence. Since $\mathbb{E}[V_k^2] \geq 2\delta^2$. Hence for any k,

$$\Pr[V_k \leq -\epsilon] = \Pr\left[V_k / \sqrt{\mathbb{E}[V_k^2]} \leq -\epsilon / \sqrt{\mathbb{E}[V_k^2]}\right] \geq \Pr\left[V_k / \sqrt{\mathbb{E}[V_k^2]} \leq -\sqrt{2}\epsilon/\delta\right] = \Phi(-\sqrt{2}\epsilon/\delta)$$

and the result follows. □

Proposition 2 *If the collection of random objective functions $\{f(x, \boldsymbol{\xi})\}_{x \in X}$ of (P) is jointly normal with a nonnegative and strictly diagonally dominant covariance matrix then*

$$\lim_{N \to \infty} \Pr\left[S_N \cap (\bigcup_{i=1}^{N} S_i) \neq \emptyset\right] = 1$$

exponentially fast.

Proof: Pick $\epsilon \in (0,1)$ and let S_*^ϵ be the set of ϵ-optimal solutions to (P), i.e. $S_*^\epsilon = \{x \in X : \mathbb{E}[f(x,\boldsymbol{\xi})] \leq \mathbb{E}[f(x,\boldsymbol{\xi})] + \epsilon \; \forall y \in X\}$. Note that,

$$\Pr\left[S_N \cap (\bigcup_{i=1}^N S_i) \neq \emptyset\right] \geq \Pr\left[[S_*^\epsilon \subseteq (\bigcup_{i=1}^N S_i)] \wedge [S_N \subseteq S_*^\epsilon]\right] \geq \Pr[S_*^\epsilon \subseteq (\bigcup_{i=1}^N S_i)] + \Pr[S_N \subseteq S_*^\epsilon] - 1,$$

where the last inequality is using Boole's inequality. From SAA theory [8] we already know that

$$\Pr[S_N \subseteq S_*^\epsilon] \geq (1 - |X| e^{-N\frac{\epsilon^2}{4\sigma_{\max}^2}}).$$

where $\sigma_{\max}^2 = \max_{x\in X}[\mathbb{V}[f(x,\boldsymbol{\xi})]$. Consider $x^* \in S_*^\epsilon$, then

$$\begin{aligned}
\Pr[S_*^\epsilon \subseteq (\textstyle\bigcup_{i=1}^N S_i) \neq \emptyset] &\geq \Pr[x^* \in (\textstyle\bigcup_{i=1}^N S_i)] \\
&= 1 - \textstyle\prod_{i=1}^N \Pr[x^* \notin S_i] \\
&= 1 - \textstyle\prod_{i=1}^N (1 - \Pr[x^* \in S_i]) \\
&= 1 - \textstyle\prod_{i=1}^N (1 - \Pr[f(x^*,\boldsymbol{\xi}) \leq f(y,\boldsymbol{\xi}) \; \forall \, y \in X \setminus \{x^*\}]).
\end{aligned}$$

Applying Lemma 1 with $K = |X \setminus \{x^*\}|$,$W_0 = f(x^*,\boldsymbol{\xi})$ and $W_k = f(y,\boldsymbol{\xi})$ for $y \in X \setminus \{x^*\}$, we get

$$\Pr\left[f(x^*,\boldsymbol{\xi}) \leq f(y,\boldsymbol{\xi}) \; \forall \, y \in X \setminus \{x^*\}\right] \geq (\Phi(-\sqrt{2}\epsilon/2\delta))^{|X|},$$

for some $\delta > 0$ depending on the covariance matrix of $\{f(x,\boldsymbol{\xi})\}_{x\in X}$. Thus

$$\Pr\left[S_N \cap (\bigcup_{i=1}^N S_i) \neq \emptyset\right] \geq (1 - (1 - (\Phi(-\sqrt{2}\epsilon/\delta))^{|X|})^N) - |X| e^{-N\frac{\epsilon^2}{4\sigma_{\max}^2}}. \tag{4}$$

Since $\Phi(-\sqrt{2}\epsilon/\delta) > 0$ for $\delta > 0$, taking limits with respect to N on both sides of (4) we get the desired result. □

We close this section with some remarks on Proposition 2. Note that when ϵ is very small relative to δ, $\Phi(-\sqrt{2}\epsilon/\delta) \approx 1/2$ and so the right hand side of (4) is approximately $(1 - (1 - (1/2)^{|X|})^N) - |X| e^{-NC}$ where C is a small constant. This indicates that smaller the solution set,$|X|$, relative to the sample size N, higher is the probability that one of the scenario solutions is optimal. If $|X|$ is large relative to N, then the bound in (4) is very loose, even negative. Consequently inequality (4) does not lead to any meaningful finite sample size estimates, but implies asymptotic behavior with respect to N for fixed X. Proposition 2 can be extended to more general distributions under different assumptions. For example, if the random vector $\{f(x^*,\boldsymbol{\xi}) - f(y,\boldsymbol{\xi})\}_{y\in X\setminus\{x^*\}}$ (where x^* is an ϵ-optimal solution of (P)) has a density and contains a ball centered at 0 of radius at least ϵ in its support, then there exists $p > 0$ such that $\Pr\left[f(x^*,\boldsymbol{\xi}) \leq f(y,\boldsymbol{\xi}) \; \forall \, y \in X \setminus \{x^*\}\right] \geq p$. We can then replace $(\Phi(-\sqrt{2}\epsilon/\delta))^{|X|}$ with p in (4) and the conclusion of Proposition 2 holds.

4 Computational Results

In this section we report some computational results using a distributed implementation of the proposed scenario decomposition to solve a class of two-stage stochastic integer programs and a class of nonlinear stochastic programs. Our implementation uses CPLEX 12.5 as the MIP solver (in single thread mode) and openMPI [7] for parallelization. We use a barrier synchronization step after solving the scenario subproblems and evaluating objective function across the processors before coordinating the scenario solutions and optimal values. Preliminary experiments with the subgradient method to update the dual solutions λ did not provide significant improvement in the lower bounds. Consequently, in the current implementation we use $\lambda = 0$ and do not update the dual solutions. We use an absolute optimality tolerance of 10^{-6} in the overall algorithm and in solving the MIP subproblems, and impose a time limit of 5000 seconds. All computations are done using the `boyle` cluster in the ISyE High Performance Computing Facility (`http://www.isye.gatech.edu/computers/hpc/`). These constitute a set of 16 identical machines each with 8 Intel Xeon 2.66GHz chips and 8 GB RAM running Linux. We limit our parallelization to 32 processors since this is the current limit on the cluster.

4.1 Two-stage stochastic integer programming instances

Our first set of results correspond to the `sslp` problem instances available in the SIPLIB library at `http://www2.isye.gatech.edu/~sahmed/siplib/`. These are a set of 10 two-stage stochastic integer programming instances from a telecommunications application [11]. The deterministic equivalent of the largest of these problems has over 1000000 binary variables. In our implementation we use continuous relaxation of the second stage variables in the lower bounding step as explained in Section 2 .

Table 2 presents the computational results. The first column is the name of the instance in the form `sslp_`n_1`_`n_2`_`N where n_1 is the number of first stage variables (all binary), n_2 is the number of second stage variables (mixed), and N is the number of scenarios. Columns 2-6 present the optimal value, the number of main iterations, total number of solutions explored, and the total solution time using 1 and 32 processors (except for the three `sslp_15_45` instances where the number of scenarios are less than 32; in this case the number of processors used is equal to the number of scenarios). Each reported solution time is the average of 3 independent runs to account for the load variability in the computing cluster. In each of the 10 instances an optimal solution is found in the first iteration, i.e. one of the solutions to the scenario subproblems was optimal. In Figure 1 we present speedup of the proposed distributed scenario decomposition algorithm with respect to the number of processors for the instance `sslp_10_50_500`. The appearance of super-linear speedup is due to the fact that the run times are averages of 3 independent runs, and perhaps also due to the cache effects. The parallel solution times are significantly smaller and serial solution times are competitive in comparison to the times reported in the literature for these instances [10, 11, 12, 13]. In particular, comparing with the parallel implementation of [10] for solving the dual problem (without recovering primal feasible solutions) we note that [10] reports 2900+ seconds on 1 processor and 800+ seconds on 32 processors for `sslp_10_50_500`.

				Time (secs)	
Problem	Optimal Value	Iterations	Solutions	1 processor	32 processors
`sslp_5_25_50`	-121.600	2	16	2.3	0.7
`sslp_5_25_100`	-127.370	2	17	6.0	1.0
`sslp_10_50_50`	-364.640	4	123	65.7	6.0
`sslp_10_50_100`	-354.190	4	187	153.3	11.0
`sslp_10_50_500`	-349.136	3	241	1296.0	46.3
`sslp_10_50_1000`	-351.711	2	191	2691.0	60.7
`sslp_10_50_2000`	-347.262	2	226	4952.0	143.3
`sslp_15_45_5`	-262.400	6	27	4.3	2.0*
`sslp_15_45_10`	-260.500	13	116	35.3	7.0*
`sslp_15_45_15`	-253.600	17	233	92.7	13.3*

Table 2: Solutions times for the `sslp` instances (*the number of processors is equal to the number of scenarios.)

4.2 Nonlinear stochastic integer programming instances

Our second set of instances is a class of expected utility knapsack problems from [1]. These problems are of the form

$$\min \left\{ -(1/N) \sum_{i=1}^{N} \mathcal{U}(\sum_{j=1}^{n} v_{ij} x_j) : \ \sum_{j=1}^{n} a_j x_j \leq 1, \ x_j \in \{0,1\}^n \right\},$$

where $\mathcal{U}(t) = 1 - \exp(-t/c)$, i.e. a negative exponential utility function with risk aversion coefficient c. We generated instances with number of variables $n \in \{25, 50, 100\}$ and number of scenarios $N \in \{100, 500, 1000\}$. The coefficients v_{ij} and a_j for each instance were generated according to [1] and $c = 4$. The data for all instances are available in the SIPLIB library at `http://www2.isye.gatech.edu/~sahmed/siplib/`.

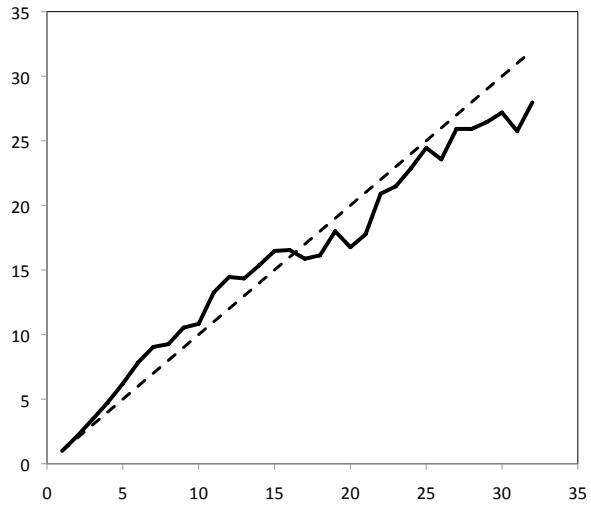

Figure 1: Speedup for `sslp_10_50_500`.

Note that for these problem, due to the monotonicity of the utility function, solving the i-th scenario subproblem in the lower bounding step (when $\lambda = 0$) is equivalent to solving the (linear) knapsack problem

$$\min \left\{ \sum_{j=1}^{n} v_{ij} x_j : \ \sum_{j=1}^{n} a_j x_j \leq 1, \ x_j \in \{0,1\}^n \right\}.$$

The lower bound is then obtained by averaging the negative of the utility function evaluated at the optimal values of such knapsack problems corresponding to each scenario. The upper bounding step constitute simply evaluating the scenario solutions.

Table 3 presents the computational results. The first column is the name of the instance in the form `exputil_`n`_`N where n is the number of variables and N is the number of scenarios. As before, columns 2-6 present the optimal value, the number of main iterations, total number of solutions explored, and the total solution time or the % optimality gap in case the 5000 sec time limit is exceeded. Each reported solution time is the average of 3 independent runs to account for the load variability in the computing cluster. Once again, in all of the instances an optimal solution is found in the first iteration.

Problem	Optimal Value	Iterations	Solutions	Time (secs) / gap (%)	
				1 processor	32 processors
`exputil_25_100`	-0.242304	2	49	4.7	1.7
`exputil_25_500`	-0.246211	2	68	24.0	6.3
`exputil_25_1000`	-0.242750	2	261	122.0	15.7
`exputil_50_100`	-0.246343	2	78	3.3	2.3
`exputil_50_500`	-0.247751	3	462	100.0	20.7
`exputil_50_1000`	-0.247643	3	323	225.3	41.7
`exputil_100_100`	-0.247542	22	2072	4443.0	690.7
`exputil_100_500`	-0.248989	>22	>10096	0.025%	0.009%
`exputil_100_1000`	-0.249317	9	5418	0.004%	721.0

Table 3: Solution times for the `exputil` instances

Acknowledgements

This research has been supported in part by the National Science Foundation (Grant # 1129871) and the Air Force Office of Scientific Research (Grant #FA9550-12-1-0154). The author thanks Dave Goldsman and Alexander Shapiro for valuable discussions.

References

[1] Shabbir Ahmed and Alper Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128(1-2):149–169, 2011.

[2] Antonio Alonso-Ayuso, Laureano F. Escudero, and M. Teresa Ortunso. Bfc, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 01 programs. *European Journal of Operational Research*, 151(3):503 – 519, 2003.

[3] Gustavo Angulo, Shabbir Ahmed, Santanu S. Dey, and Volker Kaibel. Forbidding vertices. *Working paper*, 2013.

[4] Dmitri Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[5] Claus C. Caroe and Ruediger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999.

[6] Teodor Gabriel Crainic, Xiaorui Fu, Michel Gendreau, Walter Rei, and Stein W. Wallace. Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58(2):114–124, 2011.

[7] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, 2004.

[8] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[9] Arne Lokketangen and David L. Woodruff. Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming. *Journal of Heuristics*, 2(2):111–128, 1996.

[10] Miles Lubin, Kipp Martin, Cosmin Petraa, and Burhaneddin Sandikci. On parallelizing dual decomposition in stochastic integer programming. Technical Report ANL/MCS-P3037-0912, Argonne National Labs, 2012.

[11] Lewis Ntaimo and Suvrajeet Sen. The million-variable "march" for stochastic combinatorial optimization. *J. of Global Optimization*, 32(3):385–400, 2005.

[12] Lewis Ntaimo and Suvrajeet Sen. A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, 40(3):299–319, 2008.

[13] Lewis Ntaimo and Matthew W. Tanner. Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. *Journal of Global Optimization*, 41(3):365–384, 2008.

[14] David Slepian. The one-sided barrier problem for gaussian noise. *Bell System Technical Journal*, pages 463–501, 1962.

[15] Jean-Paul Watson and David Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, pages 1–16, 2010.