Ann Oper Res (2010) 177: 127–137
DOI 10.1007/s10479-009-0670-9

An exact approach for solving integer problems under probabilistic constraints with random technology matrix

Patrizia Beraldi · Maria Elena Bruni

Published online: 12 November 2009

Abstract This paper addresses integer programming problems under probabilistic constraints involving discrete distributions. Such problems can be reformulated as large scale integer problems with knapsack constraints. For their solution we propose a specialized Branch and Bound approach where the feasible solutions of the knapsack constraint are used as partitioning rules of the feasible domain. The numerical experience carried out on a set covering problem with random covering matrix shows the validity of the solution approach and the efficiency of the implemented algorithm.

Keywords Stochastic integer programming · Probabilistic constraints · Branch and bound method

1 Introduction

Integer programming under probabilistic constraints represents one of the most challenging areas of modern stochastic programming. Its relevance is due to the possibility of bringing into the integer programming framework, the fundamental issues of reliability and risk which are of special concern in real life applications.

Let us denote by $(\Omega, \Im, \mathbb{P})$ a probability space. Later on we shall focus our attention on the following class of stochastic programming problems (IPPC, for short):

$$\min\ z = c^T x \tag{1}$$

$$\mathbb{P}\{T(\omega)x \geq h(\omega)\} \geq \alpha \tag{2}$$

$$x \in X \tag{3}$$

Here c is a n-dimensional vector, $X = \{Ax \geq b, x \in Z_+^n\}$ is shorthand for the usual deterministic constraints, where A is a $q \times n$ matrix and b is q-dimensional vector. In addition,

P. Beraldi (✉) · M.E. Bruni
Dipartimento di Elettronica, Informatica e Sistemistica, Università degli Studi della Calabria,
87030 Rende, CS, Italy
e-mail: beraldi@deis.unical.it

the decision variables x are assumed to be purely integer. The mixed-integer case can be handled in a straightforward way without any added conceptual difficulty.

Constraints (2) are joint probabilistic constraints which impose the satisfaction of the stochastic constraints $T(\omega)x \geq h(\omega)$ within a prescribed reliability level $\alpha \in (0, 1)$. We assume that ω follows a discrete distribution with finite support $\Omega = (\omega^1, \ldots, \omega^S)$. In this case Ω is called scenario set. We denote by p^s the probability of realization of the s-scenario ω^s. We observe that discrete distributions arise frequently in applications, either directly, or as empirical approximations of the underlying distribution (see, for example, Prékopa 1995).

It is evident that, linking together integrality and reliability, model (1)–(3) provides a versatile paradigm suitable to model applications occurring in almost all fields of management as well as in engineering disciplines. We mention, for example, a classical application arising in medical logistics and concerning the problem of locating and dimensioning emergency medical services so as to guarantee a reliable level of service on a given geographical region and minimize the overall costs (Beraldi et al. 2004). We also cite applications in financial risk management (Gaivoronski and Pflug 2005), in routing design (Gendreau et al. 1996), in facility location (Owen and Daskin 1998), in power industry management (Prékopa 1995).

The main difficulty in dealing with problem (1)–(3) derives from the non-convex nature of the problem. Convexity is regained only if the distribution function of the random parameters satisfies some rather strong conditions, such as, for example, the log-concavity property for continuous distributions (Prékopa 1993). In the case of discrete distributions, the feasible region defined by (2) is non-convex even for continuous decision variables. In our case, the integer restrictions add another source of non convexity. This double level of difficulty poses severe computational difficulties.

Programming under probabilistic constraints involving continuous distributions has been the subject of intensive research, beginning with the seminal work of Charnes and Cooper (1959). The interested readers are referred to Prékopa (1995) and references therein. Much less attention has received the case of discrete distributions (Dentcheva et al. 2000, 2002; Sen 1992). All the cited contributions rely on the derivation of deterministic equivalent formulations of problem (1)–(3). When uncertainty affects only the right hand side of (2) and the technology matrix is deterministic, such reformulations may be obtained by means of the α-efficient points of the distribution function. These are defined as the minimal points of the set of realizations ω^s for which $\mathbb{P}\{\omega \leq \omega^s\} \geq \alpha$ (Prékopa 1990). In Dentcheva et al. (2000) the authors proposed an algorithm that iteratively updates the set of relevant α-efficient points to generate tight lower and upper bounds for the problem. Branch and Bound methods based on partial and complete enumeration of the α-efficient points have been proposed in Beraldi and Ruszczyński (2002a) for the probabilistic set covering problem and in Beraldi and Ruszczyński (2002b) for general integer problems. In Sen (1992), the author suggested convex relaxation of probabilistically constrained problems using disjunctive programming techniques. More recently, in Cheon et al. (2006) the authors proposed a branch and bound algorithm based on the partitioning of the non-convex feasible region and on the use of bounds to fathom inferior partition elements. The basic algorithm is enhanced by domain reduction and cutting plane strategies.

Unlike most of the papers mentioned above, our contribution addresses the more general case where uncertainty affects both sides of (2). To the best of our knowledge, the only exception that dealt with a similar case is due to Ruszczyński (2002), who proposed a specialized branch and cut algorithm for the problem instance with continuous decision variables.

The remainder of the paper is organized as follows. In Sect. 2 we introduce the deterministic equivalent formulation of the problem. Section 3 is devoted to the presentation of

the solution method in its basic version. The relevant issues to exploit in order to define an enhanced approach are illustrated in Sect. 4. Finally, in Sect. 5 we present and discuss preliminary computational results collected on different instances of a set covering problem with random technology matrix.

2 The deterministic equivalent formulation

All the methods proposed in the literature for solving probabilistically constrained problems involving discrete distributions are based on the derivation of deterministic equivalent formulations of the original problem. When uncertainty affects both sides of the probabilistic constraints, the reformulation may be obtained by introducing a binary knapsack constraint. In what follows, we show how the reformulation can be derived in our case.

Let us denote by $\mathcal{S} = \{1, \ldots, S\}$ the index set for scenarios and by (T^s, h^s), with T^s a $m \times n$ matrix and h^s a m-dimensional vector, the uncertain problem parameters associated with scenario ω^s occurring with probability p^s.

Furthermore, for each $s \in \mathcal{S}$ let us denote by K^s the corresponding feasible set:

$$K^s = \{x | T^s x \geq h^s\} \tag{4}$$

For each s, K^s is assumed to be non-empty and compact. Problem (1)–(3) can be rewritten as:

$$\min\ z = c^T x \tag{5}$$

$$x \in X \cap K(\alpha) \tag{6}$$

where

$$K(\alpha) = \bigcup_{I \in \Delta} \bigcap_{s \in I} K^s \tag{7}$$

and

$$\Delta = \left\{ I | I \subseteq \{1, \ldots, S\}, \sum_{s \in I} p_s \geq \alpha \right\} \tag{8}$$

The disjunctive reformulation of $K(\alpha)$ clearly shows the non-convex nature of the problem. By adopting a standard technique used in disjunctive programming (Balas 1985), $K(\alpha)$ can be rewritten by using binary variables. In particular, for each scenario s, we may introduce a vector $M^s \in R^m$ such that $T^s x + M^s >= h^s$, for all $x \in X$. In addition, we introduce a vector y of binary variables whose components y_s, $s \in \mathcal{S}$ take value 0 if the corresponding set of constraints K^s has to satisfied and 1 otherwise. Thus, problem (5)–(6) can be equivalently rewritten as:

$$\min\ z = c^T x \tag{9}$$

$$T^s x + M^s y_s \geq h^s \quad s = 1, \ldots, S \tag{10}$$

$$\sum_{s=1}^{S} p^s y_s \leq (1 - \alpha) \tag{11}$$

$$y_s \in \{0, 1\} \quad s = 1, \ldots, S \tag{12}$$

$$x \in X \tag{13}$$

We observe that (11)–(12) define a binary knapsack constraint ensuring that the violation of the stochastic constraints is limited to $(1-\alpha)$.

In the reformulation introduced above, the number of constraints is replicated in the number of scenarios. Thus, even for small size problems, (9)–(13) results in an integer problem of very large size. As a consequence, exploiting the problem structure represents the only alternative to design effective solution algorithms.

3 The solution approach

In spite of their relevance, from both methodological and applicative standpoint, IPPC problems have received scant attention by the scientific community. To the best of our knowledge, the only contribution that dealt with a similar case is due to Ruszczyński (2002), who considered the version of the problem with nonlinear convex function, but continuous rather than integer decision variables. Ruszczyński proposed a Branch and Cut method based on the definition of a partial order relation "$\preceq$" on Ω. This relation is mathematically translated by introducing into model (9)–(13) precedence constraints. The proposed method uses valid inequalities obtained by a specialized lifting procedure for the precedence constraints.

Our method relies on a different idea: instead of attacking the full formulation of the integer problem it treats the binary knapsack separatelely using its feasible solutions as partitioning rules of the feasible domain within a Branch and Bound scheme.

Let us denote by y^c a feasible solution of (11) with $c \in \mathcal{C} = \{1, \ldots, C\}$. Then, the set $K(\alpha)$ can be rewritten as:

$$K(\alpha) = \bigcup_{c \in \mathcal{C}} \mathcal{K}(c)$$

where, for each $c \in \mathcal{C}$

$$\mathcal{K}(c) = \bigcap_{\{s \mid y_s^c = 0\}} K^s$$

It can be trivially observed that if all y^c, with $c \in \mathcal{C}$ were known, then the optimal solution of (9)–(13) would be obtained as $\min_{c \in \mathcal{C}} z^c$ where

$$z^c = \min\ c^T x$$
$$x \in X \cap \mathcal{K}(c)$$

This straightforward approach would benefit by the reduction of the cardinality $\mathcal{C}$ and, thus, of the number of integer problems to solve. To this aim some dominance criterion might be applied.

Definition 3.1 Given two feasible solutions y^1 and y^2 of (11) we say that y^1 dominates y^2 if $y^1 \geq y^2$, where the inequality "$\geq$" should be understood componentwise.

It is evident that the explicit enumeration even of the dominant solutions only, would result computationally intensive limiting the applicability of the method to very small instances.

The proposed approach uses an implicit enumeration and relies on the definition of a search tree. In particular, branches refer to the decisions of fixing a given component s

of the vector y at the value 0. This operation results in requiring the satisfaction of the corresponding set of scenario constraints K^s. Nodes refer to the corresponding subproblems. It is worthwhile noting that the search tree should be defined in such a way to guarantee that subproblems are *all* different. This can be accomplished by adopting the following branching rule.

At the top of the search tree (level 0) no fixing decision is performed, i.e. all the components of y have value 1. Since at each branch only one component is fixed, nodes at level 1 refer to $|S|$ disjunctive subproblems, each defined on one different scenario feasible domain. More generically, let us consider a node l at level j of the tree and let us denote by $\Gamma(l)$ the path joining the root of the tree to l. In particular, let us denote by $l_1, \ldots, l_j$ the different arcs and, thus, the set of indices of the vector y that have been set to 0 along $\Gamma(l)$. Starting from this node, we perform $||S| - l_j|$ branches by setting, in each branch k, the component y_k to 0, with $k = l_j + 1, \ldots, S$.

Node l inherits all the restrictions defined by the arcs (branches) of the path $\Gamma(l)$. Then, node l at level j will refer to the subproblem defined on the feasible set $\bigcap_{s=l_1}^{l_j} K^s$. In the following, the term node and corresponding subproblem will be used indistinctly.

We observe that the feasible set associated with a given node can be defined "incrementally", starting from the feasible domain of its predecessor and adding the scenario constraints implied by the last branch. This will be very useful in the solution phase allowing to adopt some warm starting procedure. In particular, in our method we consider a linear relaxation and, thus, the solution of a given subproblem can be obtained starting from the optimal solution of its predecessor rather than from scratch.

It is important to point out that some subproblems of the tree might be associated with infeasible solutions of (11). Nevertheless, the definition of the search tree assures that the solution of a subproblem provides a lower bound on the optimal solution for all the subproblems generated starting from it. As in any Branch and Bound scheme, lower bounds on the solution of subproblems can be useful in eliminating not promising partitions of the feasible domain. In our method another pruning criteria is represented by the satisfaction of (11). In fact, since by construction all the feasible solutions generated starting from the current node are dominated by that node, they can not provide a better solution.

We observe that in our scheme the satisfaction of (11) at a given node does not assure that the corresponding solution y is not dominated by another one determined at a previous level. However, the node corresponding to a dominated solution would yield a worse solution and it would be discarded during the exploration of the search tree.

In the following, the basic scheme of the proposed approach is reported:

Step 0 (Initialization). Compute an initial upper bound UB and set:
$M = \{\}$ set of subproblems to solve as integer;
$L = \{0\}$ set of active nodes, where 0 denotes the root of the tree.

Step 1 (Termination). Check L. If it is empty, STOP. Otherwise, extract a node l and go to *Step 2*.

Step 2 (Solution). Solve a linear relaxation of the subproblem l. Let x^l be the solution and $z(x^l)$ the corresponding objective function value. If $z(x^l) \geq UB$ then prune the node. If $z(x^l) < UB$ check the feasibility (11). If $\sum_{s \in \mathcal{S} - \Gamma(l)} p^s > (1 - \alpha)$ go to *Step 3* otherwise check if x^l has integer components. If so update the incumbent value, otherwise add node l to M.

Step 3 (Branching). Let l_j be the last arc of $\Gamma(l)$. Perform $(|S| - l_j)$ branches and add the generated subproblems into L. Go to *Step 1*.

When the algorithm terminates M contains all the subproblems to solve as integer. The solution process is carried out starting from the problem with the lowest bound value and whenever the incumbent value is updated, other problems are removed from M until the list is empty.

Theorem 3.1 *If the original problem is feasible, then the proposed algorithm finds the optimal solution in a finite number of iterations.*

Proof Since the number of nodes generated and explored in the search tree is finite, the algorithm terminates after finitely many steps. The optimality of the solution follows from the validity of the lower and upper bounds used. □

We observe that the proposed method has general validity and can be applied to solve problems with continuous decision variables and nonlinear convex functions.

4 Enhancements and refinements

The performance of the proposed solution method depends on different issues. In what follows, we suggest some enhancements and refinements that can be adopted to improve efficiency.

First of all, we mention the determination of the upper bound value UB used to initialize the algorithm. Let us denote by $\hat{y}$ a feasible solution of (11). Then, UB can be computed by solving the problem:

$$\min\ c^T x \tag{14}$$

$$T^s x \geq h^s \quad s \in \mathcal{S} \text{ with } \hat{y}_s = 0 \tag{15}$$

$$x \in X \tag{16}$$

Depending on the criteria used to determine $\hat{y}$, different strategies for computing UB may be defined. Here we propose an approach that takes into account the weights associated with the different scenarios. In particular, $\hat{y}$ is determined by solving the following knapsack problem:

$$\max \sum_{s=1}^{S} \gamma^s y_s \tag{17}$$

$$\sum_{s=1}^{S} p^s y_s \leq (1-\alpha) \tag{18}$$

$$y_s \in \{0,1\} \quad s = 1,\ldots,S \tag{19}$$

The value γ_s are computed on the basis of the dual variables associated with the different scenarios. These latter are determined by solving the version of the problem obtained by (9)–(13) dropping (11). The logic behind this strategy is evident: look for a solution $\hat{y}$ which corresponds to the subset of scenarios of minimal weight.

Another critical issue in the definition of an efficient approach is the preprocessing of the scenario set. It is worthwhile noting that on the basis of the comparison of the probability

values p^s, $s \in \mathcal{S}$, and the reliability level α, the value that some variables y_s will take in the optimal solution can be established a priori. In particular, if $p^s > (1-\alpha)$, then y_s can be set to 0. In the following, we shall assume that constraints corresponding to such scenarios are included in X and that Ω is accordingly reduced. Further reduction can be obtained by defining on Ω a partial order relation as suggested in Ruszczyński (2002). In particular, Ruszczyński showed that the order $i \preceq j$ defined as:

$$i \preceq j \quad \Leftrightarrow \quad h^i - T^i x \leq h^j - T^j x \quad \forall\, x \in X \tag{20}$$

is a "consistent" order for problem (9)–(13) and proved that adding the constraints

$$y_i \leq y_j \quad \forall i, j \in \{1, \ldots, S\} \text{ such that } i \preceq j \tag{21}$$

does not cut off optimal solutions.

The precedence constraints defined by the partial order may be useful in the reduction of the search tree.

Let us consider a node l of the search tree for which $\sum_{s \in \mathcal{S} - \Gamma(l)} p^s > (1-\alpha)$ and $z(x^l) < UB$. Before proceeding to the branching phase, we may check if the corresponding solution y satisfies the precedence constraints. If their satisfaction requires to fix the value of other components of y to 0, then we check if following this operation (11) is satisfied. If so we may prune node l, otherwise we proceed to branch starting from the new solution.

The advantage of the proposed solution approach relies on the possibility of exploiting lower bound values provided by the solution of the linear relaxation of subproblems defined during the exploration of the search tree. We remind that these subproblems might be associated with infeasible solutions of (11). In effect, the proposed method in its basic version might enumerate many of these solutions. In order to improve efficiency, we may limit the exploration only to "promising" infeasible solutions. This can be accomplished by modifying the branching rule and solving the linear relaxation of subproblems which satisfy some criteria. In particular, because of the branching mechanism illustrated in previous section, it is evident that it is not worthwhile to generate branches on $k = l_j + 1, \ldots, |S|$ if

$$\sum_{s=1, s \notin \Gamma(l)}^{k} p^s > (1-\alpha) \tag{22}$$

since they will produce infeasible solutions of (11). In addition, we may limit the solution phase only to subproblems for which the violation of (11) is limited by θ, where $\theta = \max_{\{s \in S \;\; | p^s \leq (1-\alpha)\}} \{p^s\}$.

5 Numerical illustration

In order to test the proposed solution approach we have considered a specific problem arising in the location of emergency services.

We consider a given geographical territory and we assume that the service request is concentrated in a finite set I of demand points (e.g. municipality or county). We also consider a given finite set J of potential sites where service facilities may be located. A candidate location j can provide service to a demand point i (i.e. i can be covered by j) only if the traveling times d_{ij}, between i and j, is within a given threshold value V. On the basis of the restricted traveling distance, it is possible to define the covering matrix T, whose generic

Table 1 Characteristics of the test problems

Test problem	$\|I\|$	$\|J\|$	$\|S\|$	*Dens*
Test 1	25	250	20	4.4%
Test 2	25	250	30	4.4%
Test 3	25	250	50	4.4%
Test 4	50	500	20	17.7 %
Test 5	50	500	30	7.2 %
Test 6	50	500	50	6.0 %

component t_{ij} is equal to 1 if $d_{ij} \leq V$ and 0 otherwise. Let us denote by c_j the cost incurred for installing the service facility at location j. The problem is to decide where to locate emergency services so to cover all the demand points minimizing the total cost. This problem can be formulated by a classical set covering model (Balas 1983), where the decision variables x_j, with $j = 1, \ldots, J$, are equal to 1 if the service facility is located at j and 0 otherwise.

It is evident that in real applications the traveling times d_{ij} can not be considered deterministic since their values may vary because of traffic conditions, speed ambulances, time of day, climate conditions, and land and road type. Thus, a more accurate model should take explicitly into account uncertainty by including a random covering matrix rather than a deterministic one. This leads to the definition of a stochastic version of the set covering problem. In particular, in a reliability perspective the problem consists in deciding the optimal location of the emergency services in such a way to cover all the demand points with a given reliability level α.

The basis of our preliminary experiments is represented by six randomly generated instances obtained by varying the number of demand points $|I|$, of potential locations $|J|$, and scenarios $|S|$. We observe that the deterministic equivalent formulation presents $[|I| \times |S| + 1]$ constraints and $[|J| + |S|]$ binary variables and is characterized by a coverage matrix T composed by $|S|$ submatrices T^s each of size $|I| \times |J|$.

Let *Dens* denote the number of nonzero entries of matrix T. Table 1 reports the characteristics of the test problems.

To provide the basis for the experiments, randomly generated test problems were created. Each entry of matrix T^s corresponds to a realization of a random variable with Bernoulli distribution. We observe that scenarios probabilities are determined in a general way allowing us to consider the more general case of dependent random variables. If the individual entries of T are independently drawn from the given probability distribution, then the scenario probabilities can be easily calculated. In order to determine the scenarios probabilities, we have randomly generated a set Π of numbers between 0 and 1 and we have extracted a subset of cardinality $|S|$ from Π whose elements obey to the second probability axiom. The full problems data are reported in Beraldi and Bruni (2005).

Each instance has been solved for three different reliability levels, 0.85, 0.90, 0.95. The proposed algorithm has been implemented in the $C++$ programming language and uses the LINDO API callable library to solve linear and integer programming problems at various steps of the method.

Table 2 reports the numerical results. They have been collected by implementing the solution approach with the enhancements introduced in Sect. 4. We present a comparison of our method with the general-purpose LINGO solver. In particular, in Table 2 we report, for each instance, the CPU seconds required to solve the deterministic equivalent reformulation

Table 2 Computational results

Test problem	α	LINGO		Our method	
		Time	*Nodes*	*Time*	*Nodes*
Test 1	0.85	6	10155	3	30
	0.90	11	33606	6	27
	0.95	1	1641	1	11
Test 2	0.85	73	1440	24	12
	0.90	52	79925	5	39
	0.95	32	44864	2	14
Test 3	0.85	25	13728	8	46
	0.90	31	7304	2	7
	0.95	19	7667	9	32
Test 4	0.85	62	49404	39	42
	0.90	44	43596	14	14
	0.95	174	161574	5	16
Test 5	0.85	> 5000	> 200000	406	104
	0.90	> 5000	> 200000	338	52
	0.95	> 5000	> 200000	1374	23
Test 6	0.85	> 5000	> 200000	149	124
	0.90	396	26794	25	7
	0.95	3752	722240	967	136

(see column *Time*) and the number of iterations performed (see column *Nodes*). For the sake of clarity we recall that although the relaxed problems are in both cases linear problems, the branch is accomplished on a different basis.

On the basis of the numerical results obtained, we can point out the following:

- For all the test problems the proposed algorithm offers significant advantage over the standard solver. This latter was not able to solve some instances within the imposed time limit of 5000 seconds. This behavior can be explained by observing that our method fully exploits the structure of the considered problem.
- We have observed that a key issue in enhancing the performance of the method is the determination of a good initial upper bound. However the time spent by the algorithm to evaluate an upper bound is not high if compared with the total time spent by the standard solver LINGO.
- The procedure used to construct an initial solution performs very well providing initial values that, in many of the test problems, coincide with the optimal solution (see Fig. 1 which shows the initial upper bound value and the optimal objective function value for the larger test problems). Nevertheless, we observe that in order to compare the standard solver with our algorithm on the same basis, the same initial upper bound values have been used.
- As known, in the worst case exact algorithms for solving the knapsack problem have exponential running time: it is important to point out that notwithstanding this very high computational complexity, our algorithm is very efficient in practice.

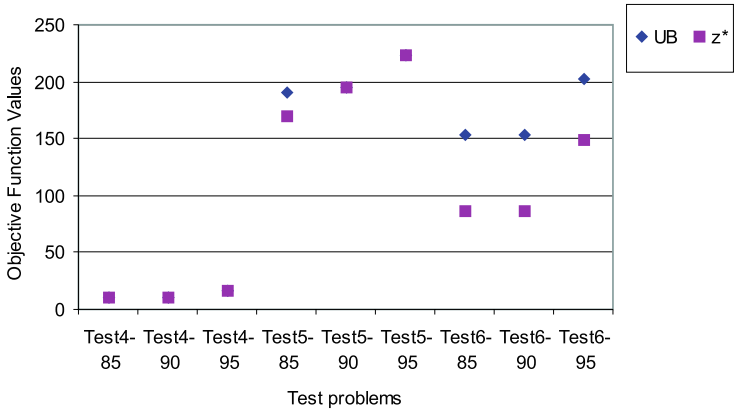

Fig. 1 Initial upper bound versus optimal solution

- The successful application of our branch and bound procedure also depends on the branch generating scheme: in fact it suffices to solve only non dominated subproblems to solve to optimality the problem. Because the cardinality of such subproblems is in general smaller than the cardinality of the whole search space, the resulting reduction allows us to tackle problems with high dimension.
- As expected, the computing time of the algorithm increases with the number of nodes for the demand points and the locations As evident from the table, LINGO has difficulty in finding an optimal solution (within the allowed time limit) when $|I| \times |J| \geq 25000$. Our algorithm seems to be able to handle such problems.
- Notice that the algorithm is quite insensitive to the scenario growth although the dimension of the problem increases rapidly when $|S|$ increases. When $|S|$ increases the CPU time initially increases until it reaches some peak and then decreases.
- The number of nodes examined by our method is significantly less than the nodes examined by LINGO. Furthermore, each LINGO node is more computational expensive than our node in which only a subset of scenarios are considered leading to a linear problem of smaller dimension.
- LINGO reported comparable solution time for one problem. However we should note that this problem instance is very easy to solve. We regard this indicator as somewhat misleading, since LINGO fails to efficiently solve more larger and complicated instances.

6 Conclusion

In this paper we presented a new solution framework for stochastic optimization problems with probabilistic constraints. In particular the solution of a general class of model not previously treated in literature has been efficiently addressed. It represents a direct generalization of the probabilistic models where uncertainty affects both sides of the stochastic constraints. Most practical problems would seem to fall into this category. The problem is a non-convex integer program for which we developed a specialized Branch and Bound algorithm. The analysis of the preliminary computational experiments indicate the proposed approach as a computationally efficient methodology for obtaining global optimal solution to realistic sized probabilistically constrained models with random technology matrix. In addition, we

developed fairly tight upper bounds on the optimal solution with no additional burden. We feel that our algorithm is a viable step toward developing an adequate methodology for dealing with such complex models.

References

Balas, E. (1983). A class of location, distribution and scheduling problems: modeling and solution methods. In P. Gray & L. Yuanzhang (Eds.), *Proceedings of the Chinese—US symposium on systems analysis*. New York: Wiley.

Balas, E. (1985). Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic and Discrete Methods*, *6*, 466–486.

Beraldi, P., & Bruni, M. E. (2005). *An exact approach for integer problems under probabilistic constraints with random technology matrix*. ParcoLab Technical Report, Unical, Italy.

Beraldi, P., & Ruszczyński, A. (2002a). The probabilistic set covering problem. *Operations Research*, *50*, 956–967.

Beraldi, P., & Ruszczyński, A. (2002b). A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software*, *17*, 359–382.

Beraldi, P., Bruni, M. E., & Conforti, D. (2004). Designing robust emergency medical service via stochastic programming. *European Journal of Operational Research*, *158*, 183–193.

Charnes, A., & Cooper, W. W. (1959). Chance-constrained programming. *Management Science*, *6*, 73–89.

Cheon, M.-S., Ahmed, S., & Al-Khayyal, F. (2006). A branch-reduce-cut algorithm for the global optimization of probabilistically constrained linear programs. *Mathematical Programming B*, *108*(2), 617–634.

Dentcheva, D., Prékopa, A., & Ruszczyński, A. (2000). Concavity and efficient points of discrete distributions in probabilistic programming. *Mathematical Programming*, *89*, 55–77.

Dentcheva, D., Prékopa, A., & Ruszczyński, A. (2002). Bounds for probabilistic integer programming problems. *Discrete Applied Mathematics*, *124*, 5–65.

Gaivoronski, A., & Pflug, G. (2005). Value-at-risk in portfolio optimization: properties and computational approach. *The Journal of Risk*, *7*, 1–339.

Gendreau, M., Laporte, G., & Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, *88*, 3–12.

LINDO Systems Inc. LINDO API 2.0 and LINGO 8.0 http://www-lind.com/.

Owen, S. H., & Daskin, M. S. (1998). Strategic facility location: a review. *European Journal of Operational Research*, *111*, 423–447.

Prékopa, A. (1990). Dual method for the solution of one-stage stochastic programming problem with random RHS obeying a discrete probability distribution. *ZOR—Methods and Models of Operations Research*, *34*, 441–461.

Prékopa, A. (1993). Contributions to the theory of stochastic programming. *Mathematical Programming*, *4*, 202–221.

Prékopa, A. (1995). *Stochastic programming*. Boston: Kluwer Scientific.

Ruszczyński, A. (2002). Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming Ser. A*, *93*, 195–215.

Sen, S. (1992). Relaxation for probabilistically constrained programs with discrete random variables. *Operations Research Letters*, *11*, 81–86.