RUTCOR
RESEARCH
REPORT

THE PROBABILISTIC SET COVERING PROBLEM

Patrizia Beraldi [a] Andrzej Ruszczyński [b]

RRR 8-99, JUNE 1999, REVISED OCTOBER 2000

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a] Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS), Italy
[b] RUTCOR - Rutgers Center for Operations Research, 640 Bartholomew Road, Piscataway, NJ 08854-8003

Rutcor Research Report
RRR 8-99, June 1999, revised October 2000

THE PROBABILISTIC SET COVERING PROBLEM

Patrizia Beraldi Andrzej Ruszczyński

Abstract. In a probabilistic set covering problem the right hand side is a random binary vector and the covering constraint has to be satisfied with some prescribed probability. We analyse the structure of the set of probabilistically efficient points of binary random vectors, develop methods for their enumeration and propose specialized branch and bound algorithms for probabilistic set covering problems.

1 Introduction

The set covering problem is one of the fundamental models of integer programming. It has a deceptively simple formulation:

$$\begin{aligned} \min\ & c^T x \\ & Tx \geq \mathbb{1}, \\ & x \in \{0,1\}^n, \end{aligned} \tag{1}$$

where T is a 0–1 matrix of dimension $m \times n$, c is an integer n-vector, and $\mathbb{1} = [1 \dots 1]^T$. Despite of that, set covering problems are very difficult and are known to belong to the class of NP-hard problems.

Set covering problems arise in manifold applications, such as scheduling, manufacturing, service planning, logical data analysis. Interested readers are referred to the survey by Balas [1] and to the annotated bibliography by Ceria, Nobili and Sassano [15]. There is also an extensive literature on numerical methods for solving such problems, both exact ([2], [4], [8], [28]) and approximate ([5], [7], [11], [14], [23], [24], [26]). An experimental comparison of the main methods is reported in the survey paper by Caprara, Fischetti, and Toth [12].

Our objective is to analyse a *probabilistic set covering problem*, in which the right hand side of (1) is replaced by a binary random vector ξ. Clearly, requiring that $Tx \geq \xi$ *for all* possible realizations of ξ would lead back to the deterministic formulation (1). Instead of that, we require that the constraints $Tx \geq \xi$ are satisfied with probability at least p, where $p \in (0,1)$ is some prespecified reliability level.

The probabilistic set covering problem is therefore formulated as follows:

$$\min\ c^T x \tag{2}$$

$$\mathbb{P}\{Tx \geq \xi\} \geq p, \tag{3}$$

$$x \in \{0,1\}^n. \tag{4}$$

The symbol $\mathbb{P}$ denotes probability. It should be stressed that we do not assume independence of the components ξ_i of ξ, $i = 1, \dots, m$, and that the constraint (3) is a *joint probabilistic constraint* in the terminology of [30].

Many deterministic set covering models have their stochastic counterparts.

Example 1.1 Suppose that we have n potential locations of service centers, and m possible locations of service requests. Let

$$T_{ij} = \begin{cases} 1 & \text{if a request at } i \text{ can be served by center located at } j, \\ 0 & \text{otherwise.} \end{cases}$$

Our binary decision variables, x_i, represent the selection of service center locations: $x_i = 1$ if a center is located at i, otherwise $x_i = 0$.

Let ξ be an m-dimensional binary random vector representing service requests that occur: $\xi_i = 1$ if there is a service request at location i; otherwise $\xi_i = 0$. Then inequality (3) can

be interpreted as the requirement that the centers located can serve the requests with the probability at least p. Many variations are possible here, with several matrices T_k for different service quality levels, and several probability thresholds p_k.

Example 1.2 Consider a graph with node set $\mathcal{N}$ and arc set $\mathcal{A}$. Our "adversary" chooses at random one or many paths in the graph, from some set of paths Π. We want to select a subset of arcs $\mathcal{D} \subseteq \mathcal{A}$, such that the probability that each adversaries' path has at least one arc in $\mathcal{D}$ (is "intercepted") equals p or more.

We define the binary random vector ξ to represent our adversaries' choice: for each path $\pi \in \Pi$ we have $\xi_\pi = 1$ if path π is selected, and $\xi_\pi = 0$ otherwise.

Let T be the arc–path incidence matrix:

$$T_{\pi a} = \begin{cases} 1 & \text{if path } \pi \text{ contains arc } a, \\ 0 & \text{otherwise.} \end{cases}$$

Our decision variables $x_a \in \{0,1\}$ model the choice of arcs: $x_a = 1$ if $a \in \mathcal{D}$, and $x_a = 0$ otherwise. With these definitions inequality (3) represents the requirement on the probability of interception.

Remark 1.1 The *expected value problem*, understood as the deterministic model in which random variables are replaced by their expectations, takes on the form

$$\begin{aligned} \min\ & c^T x \\ & Tx \geq \mathbb{E}\xi, \\ & x \in \{0,1\}^n. \end{aligned}$$

If $\mathbb{E}\xi_i > 0$, $i = 1, \ldots, m$, this problem is equivalent to (1), that is, to the *worst case problem*, in which the constraint has to be satisfied for *all* realizations of ξ. Therefore, the expected value approach is of no interest here.

Remark 1.2 If the components of ξ are independent, the deterministic equivalent of (2)–(4) can be developed as follows. The probabilistic constraint (3) is equivalent to the existence of a vector $z \in \{0,1\}^m$ such that

$$Tx \geq z$$

and

$$\mathbb{P}\{\xi \leq z\} \geq p.$$

Using the independence of the components of ξ, and taking the logarithm of both sides, we can write the last inequality as

$$\sum_{i=1}^{m} \ln(\mathbb{P}\{\xi_i \leq z_i\}) \geq \ln(p).$$

Let $q_i = \mathbb{P}\{\xi_i = 0\}$, $i = 1, \ldots, m$. Since $\ln(\mathbb{P}\{\xi_i \leq z_i\}) = (1 - z_i)\ln(q_i)$, we arrive at the following re-formulation of (2)–(4)

$$\begin{aligned} \min\ & c^T x \\ & Tx \geq z, \\ & \sum_{i=1}^{m} (1 - z_i)\ln(q_i) \geq \ln(p), \\ & x \in \{0,1\}^n, \quad z \in \{0,1\}^m. \end{aligned}$$

However, if the components of ξ are dependent, we cannot carry out the crucial transformation of the probabilistic constraint into a linear constraint.

Our motivation for this research is twofold. First, the probabilistic set covering problem is interesting in its own and has a number of potential applications. Secondly, we intend to use it as a convenient laboratory model for developing new concepts and algorithms which will be useful for a broader class of problems: stochastic programming problems with probabilistic constraints involving both integer decision variables and integer random variables. The literature about these problems is very scarce, and main developments are still ahead of us. We can mention here the works [25] and [22] on routing problems with probabilistic constraints (see also [9]), and few papers that address probabilistically constrained problems with discrete random variables, but continuos decision variables ([21], [29], [31], [32]).

In the next section we shall recall the notion of a p-efficient point of a discrete distribution and we shall discuss its role in the analysis of problem (2)–(4). Section 3 is devoted to procedures for enumerating p-efficient points of binary random vectors. In section 4 we shall discuss branch and bound methods for solving the probabilistic set covering problem. Finally, in section 5 we shall present some numerical results.

We use $\mathbb{Z}^m$ to denote the set of integer vectors of dimension m. The symbol $|\cdot|$ denotes the ℓ_1 norm of an m-vector, that is, $|v| = \sum_{i=1}^{m} |v_i|$. We use conv$A$ to denote the convex hull of the set A.

2 p-Efficient points

Let $F : \mathbb{R}^m \to [0,1]$ denote the probability distribution function of ξ, that is, $F(v) = \mathbb{P}\{\xi \leq v\}$. To analyse the structure of the feasible set of problem (2)–(4) we define the set:

$$\mathcal{Z}_p = \{z \in \mathbb{R}^m : F(z) \geq p\}. \tag{5}$$

All solution strategies proposed in this paper are based on the concept of *p-efficient* point of a probability distribution function. This concept has been introduced by Prékopa in [29] and extensively analysed in [21]. Here we restrict this notion to binary random vectors.

Definition 2.1 *A point $v \in \{0,1\}^m$ is called a* p-efficient *point (PEP) of the probability distribution F, if $F(v) \geq p$ and there is no binary $y \leq v$, $y \neq v$ such that $F(y) \geq p$.*

The number of p-efficient points of any discrete random vector is finite [21]; for a binary random vector this property is trivial.

Let $p \in (0,1)$ and let $v^1, \ldots, v^N$ be all p-efficient points of ξ. Then

$$\mathcal{Z}_p = \bigcup_{i=1}^{N} (\{v^i\} + \mathbb{R}^m_+). \tag{6}$$

We can thus replace (3) by the *disjunctive constraint*:

$$Tx \geq v^i \quad \text{for at least one } i \in \{1, \ldots, N\}.$$

So, there are two sources of non-convexity in the probabilistic set covering problem: one introduced by the integrality condition on the decision variables, and the other one introduced by the probabilistic constraint. The following result shows that the non-convexity introduced by the disjunctive constraint is absorbed by the integrality condition.

Lemma 2.1 $\mathcal{Z}_p \cap \mathbb{Z}^m = \operatorname{conv}\mathcal{Z}_p \cap \mathbb{Z}^m$.

Proof. It is sufficient to show that every integer vector $z \in \operatorname{conv}\mathcal{Z}_p$ is an element of $\mathcal{Z}_p$. Let

$$z \geq \sum_{j \in J} \lambda_j v^j,$$

where the set $J \subseteq \{1, \ldots, N\}$ is such that $\lambda_j \in (0,1)$, $j \in J$. Since z is integer and the v^j's are binary,

$$z \geq \max\{v^j,\ j \in J\},$$

so $z \in \mathcal{Z}_p$, as required. □

Let us now pass to the derivation of bounds for p-efficient points. They will use the conditional marginal distributions

$$F_i(v_i \mid \xi \leq w) = \mathbb{P}\{\xi_i \leq v_i \mid \xi \leq w\}, \quad i = 1, \ldots, m.$$

Lemma 2.2 *Let $p \in (0,1)$ and let $w \in \{0,1\}^m$ be such that $F(w) \geq p$. Define l_i to be the $\frac{p}{F(w)}$-efficient point of the conditional marginal $F_i(v_i \mid \xi \leq w)$, that is*

$$l_i(w) = \begin{cases} 1 & \text{if } F_i(0 \mid \xi \leq w) < p/F(w), \\ 0 & \text{if } F_i(0 \mid \xi \leq w) \geq p/F(w) \end{cases} \qquad i = 1, \ldots, m. \tag{7}$$

Then

(i) for every p-efficient point $v \leq w$ we have $v \geq l(w)$;

(ii) if $z \leq w$ then $l(z) \geq l(w)$;

(iii) w is p-efficient if and only if $l(w) = w$.

Proof. For any vector $v \leq w$ and for any i we have

$$F(v) = F(w)\mathbb{P}\{\xi \leq v \mid \xi \leq w\} \leq F(w)\mathbb{P}\{\xi_i \leq v_i \mid \xi \leq w\} = F(w)F_i(v_i \mid \xi \leq w).$$

Therefore, if v is p efficient, we must have

$$F_i(v_i \mid \xi \leq w) \geq p/F(w),$$

so, $v_i = 1$, if $F_i(0 \mid \xi \leq w) < p/F(w)$. This proves (i).

Let us prove (ii). If $z \leq w$,

$$\begin{aligned} F_i(0 \mid \xi \leq z)F(z) &= \mathbb{P}\{\xi_i = 0 \mid \xi \leq z\}\mathbb{P}\{\xi \leq z\} \\ &= \mathbb{P}\{\xi_i = 0, \xi \leq z\} \\ &\leq \mathbb{P}\{\xi_i = 0, \xi \leq w\} \\ &= \mathbb{P}\{\xi_i = 0 \mid \xi \leq w\}\mathbb{P}\{\xi \leq w\} \\ &= F_i(0 \mid \xi \leq w)F(w). \end{aligned}$$

Therefore $l(z) \geq l(w)$, as required.

To prove (iii) suppose that $l_i(w) < w_i$ for some i, that is, $w_i = 1$ and $l_i(w) = 0$. Define $\bar{w} = (w_1, \ldots, w_{i-1}, 0, w_{i+1}, \ldots, w_m)$. Then

$$F(\bar{w}) = F(w)F_i(0 \mid \xi \leq w) \geq p,$$

so w cannot be p-efficient. □

We shall use Lemma 2.2 to trim the search tree in methods for enumerating p-efficient points and in branch and bound methods for solving (2)–(4).

3 Enumeration of p-efficient points

3.1 Forward enumeration

In [31], Prékopa et al. proposed a conceptual algorithm for the enumeration of the PEP for the general case of discrete random variables. It uses nesting with respect to the dimension of the random vector. In the case of binary random variables, the enumeration of PEP can be carried out in a more efficient way.

Throughout, a binary vector $v \in B^m$ is said to belong to level j if $|v| = j$.

On the basis of the definition of PEP, it is evident that a natural strategy to adopt for their determination is based on a "level" search. We may start from the lowest level and move from one level to the next until no more PEP can be found.

This basic scheme can be improved by using two main considerations. The first one follows from Lemma 2.2. By setting $w = \mathbb{1}$, we can define the point $l = l(\mathbb{1})$ as follows:

$$l_i = \begin{cases} 1 & \text{if } F_i(0) < p, \\ 0 & \text{if } F_i(0) \geq p \end{cases} \qquad i = 1, \ldots, m. \tag{8}$$

Here, F_i is the ith marginal distribution of ξ (without any conditions). By Lemma 2.2, the vector l constitutes the lower bound for all p-efficient points of ξ. Basing on that, we can fix to 1 some components of the potential PEP and restrict our search to a space of lower dimension. In effect, the starting level will be $j + 1$ where $j = |l|$.

The second consideration follows from the concept of *dominance*.

Let v be a PEP of level j. We say that it dominates a point w of level k if $v \leq w$. Thus a point w of level k can be PEP only if it is not dominated by any PEP belonging to the previous levels.

It is evident that each point of level j dominates $\binom{m-j}{k-j}$ points of level k, and each point of level k is dominated by $\binom{k}{j}$ points of level j.

Let us now consider a certain level k. It can be partitioned into three different subsets: dominated points, PEP, nondominated and not PEP. Among all these points, only those belonging to the third subset represent potential candidates for further exploration. In particular, each candidate at level k will be the father of $m - k$ children, out of which some will be either dominated, or PEP, or will become candidates for the next level. It is important to stress that each child has many fathers (k in this case) and, consequently, it is necessary to introduce some mechanism preventing duplication of children which have already been generated ("prime" children). The enumeration process terminates when the number of candidates at the current level is zero.

The algorithm presented below formalizes these ideas.

1. **Initialization**
   Determine the lower bound vector l. Set:
   $k = |l|$ level counter,
   $S^k = \emptyset$ set of PEP at level k,
   $C^k = \{l\}$ set of candidate points at level k,
   $P^k = \emptyset$ set of "prime" children of points of level k,
   $J = \emptyset$ set of all PEP found fo far.

2. **Enumeration**
   For each $v \in C^k$ generate all its "prime" children and add them to P^k.

3. **Dominance Test**
   For each $v \in P^k$, verify if there exists a point in J dominating it. If this is the case, eliminate v; otherwise, compute the value of the probability distribution function $F(v)$. If $F(v) \geq p$, add v to S^{k+1}, otherwise add v to C^{k+1}.

4. **Termination Test**
   Set $J = J \cup S^{k+1}$. If $C^{k+1} = \emptyset$, stop; otherwise, increase k by 1 and go to Step 2.

To avoid duplication, with each node v we associate the position $i(v)$ at which v differs from its father (the last position set to 1). To generate "prime" children, we consider only $i > i(v)$ as potential positions to be set to 1.

The determination of the PEP through the proposed algorithm can be seen as a breadth-first search procedure on the tree of all binary vectors of dimension $m - |l|$ ($|l|$ positions have been already fixed to 1). The root of the tree is the node defined by the vector l. It is clear that if m is large and $|l|$ is not close to m we may encounter serious difficulties in this type of enumeration.

3.2 Backward enumeration

An alternative approach to the determination of the PEP is based on a "backward" search. We start from the node of maximum level (the vector $\mathbb{1}$) and we move to lower levels of the binary tree. In the backward search the determination of the PEP is based on the application of Lemma 2.2.

Let us consider a point w at level $k = m - j$, where $j \geq 1$. By Lemma 2.2, all p-efficient points $v \leq w$ are dominated by the vector $l(w)$ given by (7). We define the following sets:

$$I(w) = \{i \in \{1, \ldots, m\} \ : \ l_i(w) = 1\},$$
$$J(w) = \{i \in \{1, \ldots, m\} \ : \ w_i = 0\}.$$

Clearly, the components $i \in I(w)$ can be permanently fixed to 1 at all candidates $v \leq w$, while the components $i \in J(w)$ are already fixed to 0. Since w has been obtained by fixing to zero one component in a vector z at the higher level, Lemma 2.2(b) implies that $I(z) \subseteq I(w)$. Therefore, only the components $l_i(w)$ for $i \notin I(z) \cup J(w)$ need be computed.

If $l(w) \neq w$, some components are still free, that is the set $\{1, \ldots, m\} \setminus I(w) \setminus J(w) \neq \emptyset$. Only these components will be used to branch by setting one of them to 0.

Since a predecessor of w is also a predecessor of other nodes at level k, the same node could be generated many times. In order to avoid this, some mechanism that guarantees the generation of the "prime" predecessors only has to be introduced. Similarly to the forward method, with each node w we associate the last position $i(w)$ at which w has a zero. Only $i > i(w)$ are considered as candidates for branching (setting $w_i = 0$). Thus, we need only to compute $l_i(w)$ for

$$i \in \{i(w) + 1, \ldots, m\} \setminus I(w) \setminus J(w).$$

The generation procedure terminates when all the potential PEP w at a given level satisfy the condition $l(w) = w$.

The "backward" algorithm can be formalized as follows:

1. **Initialization**
   Determine the vector l. Set:
   $k = m$ level counter,
   $S^k = \emptyset$ set of PEP at level k,
   $C^k = \{\mathbb{1}\}$ set of candidate points at level k,
   $P^k = \emptyset$ set of "prime" predecessors of points at level k,
   $J = \emptyset$ set of all PEP.

2. **Enumeration**
   For each $v \in C^k$ generate all its "prime" predecessors and add them to P^k.

3. **Efficiency Test**
   For each $v \in P^k$, compute $l(v)$. If $l(v) = v$ then add v to S^{k-1}; otherwise, add v to C^{k-1}.

4. **Termination Test**
   Set $J = J \cup S^{k-1}$. If $C^{k-1} = \emptyset$, stop; otherwise, decrease k by 1 and go to Step 2.

The main advantage of this method is the quick improvement of the lower bounds $l(w)$ in the course of computation. The more components are set to 0 by branching, the more other components will be fixed to 1 by lower bounds. This facilitates the trimming of the search tree.

4 Solution methods

In order to solve the probabilistic set covering problem, we can adopt two different strategies that we refer to as *complete* and *hybrid*.

The complete strategy consists of two distinct phases. In the first one we enumerate all PEP, whereas in the second phase we determine, by using some tailored solution method, the optimal solution of the problem.

Generating all the PEP is computationally expensive, because their number can be very large even for moderate values of m. The hybrid solution approaches try to avoid the complete enumeration of the PEP and to generate them only when required. The key idea is to alternate the phase of enumeration and the phase of solution of the corresponding problems until some termination condition is reached.

It is evident that a crucial role in the hybrid solution methods will be played by the upper and lower bound values. Indeed, on the basis of the comparison between the lower bound value associated with a potential PEP, and the value of the best known feasible solution, we can decide either to eliminate the node, or to label it as a candidate for further exploration.

As pointed out in section 3, the determination of PEP can be carried out by visiting the tree of all binary vectors starting either from the node l of the lowest bound, or from the node $\mathbb{1}$ of maximum level.

By using these two proposed enumeration schemes, we may develop two hybrid solution methods: *forward* and *backward*.

4.1 Complete methods

Once all PEP $v^1, \ldots, v^N$ are known, the straightforward way to solve the probabilistic set covering problem, is to solve the problem

$$\begin{aligned} \min\ & c^T x \\ & Tx \geq v^k, \\ & x \in \{0,1\}^n \end{aligned} \tag{9}$$

for each $k = 1, \ldots N$. The optimal solution of the original problem will be the best one among the N optimal solutions determined. Each such problem is a deterministic set covering problem in which the rows of the matrix T that correspond to the 0 entries of the PEP can be removed.

It is well known that the set covering problem is NP-hard, thus an efficient solution strategy should try to avoid the solution (as integer) of as many problems as possible. This can be accomplished by using a "bounding–pruning" scheme. For each problem we may derive a lower bound on its optimal value by solving its relaxation (e.g. linear or lagrangian). On the basis of a comparison with some known feasible solution, we can decide either to prune the problem or to hold it as a potential candidate for further exploration.

Critical to this strategy is the determination of a "starter", that is the first problem whose integer solution (either optimal or feasible) will be used as an upper bound, z_I. Heuristic procedures for generating the upper bound will be discussed in section 4.4.

In the complete method we maintain a list L of all PEP ordered by level. At each iteration, we select a PEP v^k from L and we solve the relaxation of the corresponding problem (9). As in the standard branch and bound algorithm (here we use only the bounding part), we compare the round-up optimal value of this problem, z_k, with the upper bound z_I. The following cases can happen.

- The solution is not integer and $z_k \geq z_I$; in this case we prune the problem.
- The solution is not integer, but $z_k < z_I$; in this case we add the problem to the list M of potential candidates to be solved as integer.
- The solution is integer, but $z_k < z_I$. Then we update the value of z_I and eliminate from the list M all problems with larger values of the objective function.

Once the list L is empty, we proceed with the determination of the integer solution starting from the problem with the smallest lower bound value. Each time a better value for z_I is found, all the problems $j \in M$ with round-up value $z_j \geq z_I$ are eliminated. The procedure terminates when M is empty.

4.2 The forward branch and bound method

The forward branch and bound method combines the forward enumeration procedure with a "bounding–pruning" scheme. To initialize the method we need a feasible solution which

will serve as the first upper bound z_I. This can be computed by solving (9) with a first PEP t as right-hand side.

Suppose that the forward enumeration procedure of section 3.1 is at level k in the tree of binary vectors. The non-dominated points of the level can either belong to the set S^k of PEP, or to the set C^k of potential candidates to be further explored.

For each node $v \in S^k \cup C^k$, we solve a relaxation of the problem:

$$\begin{aligned} \min\ & c^T x \\ & Tx \geq v, \\ & x \in \{0,1\}^n. \end{aligned} \tag{10}$$

Let $z(v)$ be the round-up value of the objective function. It is evident that this value represents a lower bound not only on the optimal solution of problem (10), but also on optimal solutions of all problems corresponding to nodes generated from v (that is, for nodes $w \geq v$). Thus, on the basis of the comparison between $z(v)$ and the best upper bound, z_I, we can decide either to prune the node (if $z(v) \geq z_I$) or to proceed further (if $z(v) < z_I$). In particular, if the last condition occurs and v is a PEP ($v \in S^k$) it may generate a better upper bound. If the solution happens to be integer, we update z_I; otherwise, we add problem (10) to the list M of problems to be solved as integer.

The condition $z(v) \geq z_I$ can also be used to prune other nodes at higher levels. Although we prevent the duplication of immediate successors, we cannot avoid the situation that a successor of the current node has a predecessor at earlier levels which have already been discarded. In order to avoid this redundant generation (and solution of the corresponding relaxed problems), we can label v as a "pseudo" PEP, and use it in the dominance test to prune additional nodes.

The forward solution method can be formalized as follows.

1. **Initialization**
   Determine the vector l and the upper bound z_I. Set:
   $k = |l|$ level counter,
   $S^k = \emptyset$ set of PEP at level k,
   $C^k = \{l\}$ set of candidate points at level k,
   $P^k = \emptyset$ set of "prime" children of points at level k,
   $L^k = \emptyset$ set of "potential" candidates at level k,
   $R = \emptyset$ set of "pseudo" PEP,
   $M = \emptyset$ list of the problems to be solved as integer,
   $J = \{t\}$ set of all PEP.

2. **Enumeration**
   For each $v \in C^k$, generate all its "prime" children and add them to P^k.

3. **Dominance Test**
   For each $v \in P^k$, verify if there exists any point in $J \cup R$ dominating it. If this is the

case, discard v; otherwise, compute at v the probability distribution function $F(v)$. If $F(v) \geq p$, then add v to S^{k+1}; otherwise, add v to L^{k+1}.

4. **Termination Test**
   Set $J = J \cup S^{k+1}$. If $L^{k+1} = \emptyset$ and $S^{k+1} = \emptyset$, Stop; otherwise, increase k by 1.

5. **Solution**
   For each $v \in S^k \cup L^k$, solve a relaxation of problem (10). Let $z(v)$ be the objective function value.

   If $z(v) \geq z_I$, then

   (i) if $v \in L^k$ add v to R;

   (ii) if $v \in S^k$ discard the point.

   If $z(v) < z_I$, then

   (i) if $v \in L^k$ then add v to C^k;

   (ii) if $v \in S^k$ and the solution is fractional then add problem (10) to M; if the solution is integer, update the upper bound $z_I = z(v)$, and eliminate from M all problems with a worse lower bound value.

   If $C^k = \emptyset$, stop; otherwise, go to Step 2.

When the algorithm terminates, the list M contains all the most promising problems to solve as integer. As in the case of the complete strategy, we determine the optimal solution starting from the problem with the smallest lower bound value. Each time a better z_I is obtained, we prune some problems until M is empty.

4.3 The backward branch and bound method

In the backward method the determination of the lower bound values is based on Lemma 2.2.

Let us consider a node w at level $m - j$. In order to determine a lower bound on the optimal solution for all the problems corresponding to potential p-efficient predecessors of w, we solve a relaxation of the following problem:

$$\begin{aligned} \min\ & c^T x \\ & Tx \geq l(w), \\ & x \in \{0,1\}^n, \end{aligned} \tag{11}$$

where the vector $l(w)$ is computed by (7). As in the forward method, we compare the round-up of the optimal objective value $z(l(w))$ of a relaxation of (11) and the best known feasible solution z_I, and we either eliminate the node or continue exploration. In particular, if $z(l(w)) \geq z_I$, then node w can be discarded. Furthermore, in order to avoid generation

of poor candidates, we can store the vector $l(w)$ and use it in an efficiency test to prune all $v \geq l(w)$. If $z(l(w)) < z_I$, then w is a promising candidate. In addition, if $l(w) = w$, we add problem (11) to the list M of problems to be solved as integer, if the solution is fractional; otherwise we update z_I and prune.

The backward method can be formalized as follows.

1. **Initialization**
   Determine the vector l, and the upper bound z_I. Set:
   $k = m$ the level counter,
   $S^k = \emptyset$ set of PEP at level k,
   $C^k = \emptyset$ set of candidate points at level k,
   $P^k = \emptyset$ set of "prime" predecessors of points at level k,
   $L^k = \emptyset$ set of "potential" candidates at level k,
   $M = \emptyset$ list of the problems to solve as integer,
   $R = \emptyset$ set of eliminated points,
   $J = \{t\}$ set of all PEP found.

2. **Enumeration**
   For each $v \in C^k$, generate its "prime" predecessors and add them to P^k.

3. **Efficiency Test**
   For each $v \in P^k$, verify if there exists any point w in R such that $w \leq v$. If this is the case, discard v; otherwise, compute $l(v)$. If $l(v) = v$ add v to S^{k-1}; otherwise, add v to L^{k-1}.

4. **Termination Test**
   If $S^{k-1} = \emptyset$ and $L^{k-1} = \emptyset$, stop; otherwise, decrease k by 1.

5. **Solution**
   For each $v \in S^k \cup L^k$ solve a relaxation of problem (11). Let $z(l(v))$ be the objective function value.

   If $z(l(v)) \geq z_I$, then

   (i) if $v \in L^k$ add $l(v)$ to R,

   (ii) if $v \in S^k$ discard the point.

   If $z(l(v)) < z_I$, then

   (i) if $v \in L^k$ add v to C^k;

   (ii) if $v \in S^k$ add problem (11) to M, if the solution is fractional; if the solution is integer, update $z_I = z(l(v))$, and eliminate from M all problems with a worse lower bound value.

   If $C^k = \emptyset$, stop; otherwise, go to Step 2.

As in the forward method, at termination the list M contains all problems to be solved as integer.

4.4 Greedy heuristics

All solution methods share the problem of determining an initial upper bound. Such a problem can be addressed by using different strategies.

An adaptation of Chvátal's greedy heuristic method [17] for deterministic set covering problems belongs to this class. We find the variable x_{j_1} for which the ratio of its cost coefficient c_{j_1} to the number of nonzeros $|T_{j_1}|$ in its column of T is minimal, and we set $x_{j_1} = 1$. At step k, when variables $x_{j_1}, \ldots, x_{j_{k-1}}$ are already set to 1, we check whether $F(Tx) \geq p$. If so, we stop; otherwise, we select the next variable x_{j_k} for which the ratio of c_{j_k} to the number of nonzeros in its column T_{j_k} in the rows which are not yet covered is minimal, etc. At termination, we calculate $v = Tx$ and solve to optimality the corresponding set covering problem.

It is evident that the upper bound value found in this way may be rather weak, because $v = Tx$ may be much larger than a PEP.

An improvement can be obtained by searching for a p-efficient point of the lowest possible level. In the complete method we can simply select a point v^{i^*} at random from the set of PEP having the least norm. In the backward enumeration method, we proceed in a depth first fashion as long as there is a possibility to go to a lower level. The last PEP v^{i^*} found in the branch explored in this way is entered to the set covering problem (9). The integer solution of this problem will be our first upper bound. We shall call this approach the simple PEP-based heuristic.

A stronger bound can be obtained by using the vector of Lagrange multipliers u corresponding to the deterministic problem (1).

In the complete methods, having enumerated all p-efficient points $v^1, \ldots, v^N$, we choose v^{i^*} for which the dual bound $u^T v^i$ is minimal. Then we solve as integer the set covering problem (9).

In the backward branch and bound method, we proceed in a depth-first fashion as follows. At step k, given v^k such that $F(v^k) \geq p$, we calculate the lower bound $l(v^k)$ by (7). Then, among the coordinates i such that $l_i(v^k) < v_i^k$ we find i_k such that

$$u_{i_k} = \max\{u_i : l_i(v^k) = 0\}.$$

Then we set

$$v_i^{k+1} = \begin{cases} 0 & i = i_k, \\ v_i^k & i \neq i_k, \end{cases}$$

and the iteration continues. The algorithm stops when $l(v^k) = v^k$, that is, when v^k is a p-efficient point (Lemma 2.2). At this point we solve the corresponding deterministic problem to optimality. We shall call this method dual PEP-based heuristic.

5 Computational Experiments

This section is devoted to the presentation and discussion of the computational experiments. In the first part we shall focus our attention on the enumeration methods: we analyse the main computational issues and present the numerical results. The second part is dedicated to the solution methods: crucial factors affecting the performance of the methods are discussed and the numerical results of both the complete and hybrid solution strategies are presented.

5.1 Computational issues for enumeration strategies

In order to evaluate the performance of enumeration and solution methods, we consider a binary random vector ξ whose components ξ_i, $i = 1, \ldots, m$, are partitioned into independent subvectors $\xi^1, \ldots, \xi^L$ of dependent random variables. This assumption makes the model rather flexible and encompasses the cases of all independent and all dependent random variables.

Clearly, computation of the probability distribution function at a given point $v \in \{0,1\}^m$ is an indispensable part of the procedure. We do not need to store the huge array with the values of F at each binary point; it is sufficient to store lower dimensional marginal distributions $F_1, \ldots, F_L$ for each of L independent groups of dependent components. Then $F(v) = F_1(v^1) \cdot \cdots \cdot F_L(v^L)$, where $v^1, \ldots, v^L$ are the corresponding subvectors of v. To efficiently store the s-dimensional marginal probability distribution function we treat the binary argument as a binary representation of an integer, so only a real array of dimension 2^s of function values needs to be stored.

As far as the forward generation of binary points is concerned, a reasonable way to proceed is to store only the positions of the binary point at which 1 appears. So, for points of level 1 we store one position, for points of level 2 two positions, and so on. To generate points of a given level k starting from points of the previous level $k-1$ the following ordering scheme avoids duplication. Let v be a point of level $k-1$ with positions $(i_1, \ldots, i_{k-1})$ fixed to 1. If $i_{k-1} < m$, starting from v we generate $s = m - i_{k-1}$ points w with the last 1 at $i_k = i_{k-1} + j$, for $j = 1, \ldots s$. In this way, we avoid the generation of the same point at a given level, starting from different points of the previous level.

For backward methods the approach is similar. We only store positions at which 0 appears. For a point v of level $m - k + 1$ with positions $(i_1, \ldots, i_{k-1})$ fixed to 0, we generate $s = m - i_{k-1}$ points w with the last 0 at $i_k = i_{k-1} + j$, for $j = 1, \ldots s$.

5.2 Numerical results of the enumeration methods

The test problems used in our computational experiments have been generated by specifying, in the general model introduced above, the type of dependency among the random components of a given group.

Let us denote by $X_1 \ldots, X_s$ the dependent variables belonging to a subvector of size s. We have considered two different types of dependency.

The first type of dependency model is the following. Given independent Bernoulli random variables $Y_1, \ldots Y_s$ with parameters $\alpha_1, \ldots \alpha_s$, we define the variables $X_1 \ldots X_s$ as

$$X_i = Y_i \vee Y_{i(\mathrm{mod}\ s)+1}, \quad i = 1, \ldots, s,$$

where $\vee$ and `mod` denote the union and *modulo* operators, respectively. In a such a scheme two consecutive random variables have a common source of dependency. The scheme, referred in the following as "circular", can be depicted as in Figure 1.

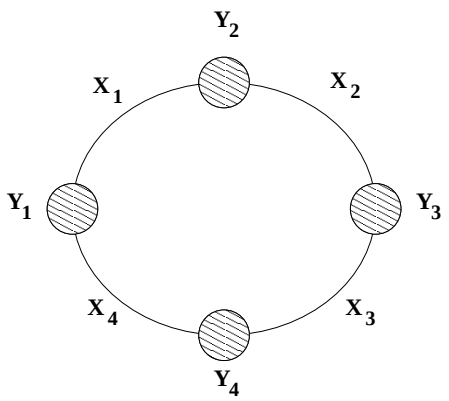

Figure 1: The circular dependency scheme.

In the second case given independent Poisson random variables $Y_1, \ldots Y_s$ with parameters $\lambda_1, \ldots, \lambda_s$, we first define the integer vector $V_1 \ldots V_s$ as follows:

$$V_i = Y_i + Y_{s+1}, \quad i = 1, \ldots s.$$

The distribution of $(X_1, \ldots, X_s)$ is the conditional distribution of $(V_1, \ldots, V_s)$ under the condition that $V_i \leq 1$, $i = 1, \ldots, s$. This is a rather involved scheme of dependency (see [30]). As an illustration, we report the computation of the mass function for a subvector of size 2.

Let us denote by $\lambda_1, \lambda_2, \lambda_3$ the parameters of three independent Poisson random variables Y_1, Y_2, Y_3. The dependent variables V_1, V_2 are defined as follows:

$$V_1 = Y_1 + Y_3,$$
$$V_2 = Y_2 + Y_3.$$

Let k_1 and k_2 be two non-negative integer values. The following relation holds:

$$\begin{aligned}
\mathbb{P}\{V_1 = k_1, V_2 = k_2\} &= \sum_{k=0}^{\min(k_1,k_2)} \mathbb{P}\{Y_1 + Y_3 = k_1, Y_2 + Y_3 = k_2 \mid Y_3 = k\} \cdot \mathbb{P}\{Y_3 = k\} \\
&= \sum_{k=0}^{\min(k_1,k_2)} \mathbb{P}\{Y_1 = k_1 - k, Y_2 = k_2 - k\} \cdot \mathbb{P}\{Y_3 = k\} \\
&= e^{-(\lambda_1+\lambda_2+\lambda_3)} \sum_{k=0}^{\min(k_1,k_2)} \left(\frac{\lambda_3}{\lambda_1\lambda_2}\right)^k \lambda_1^{k_1}\lambda_2^{k_2}.
\end{aligned}$$

By replacing k_1, k_2 with the values 0, 1 and normalizing we get the mass distribution function of X_1, X_2. In the described model there exists a common source of dependency for all components of the random variable within a subgroup. The model, referred in the following as "star", can be depicted as in Figure 2.

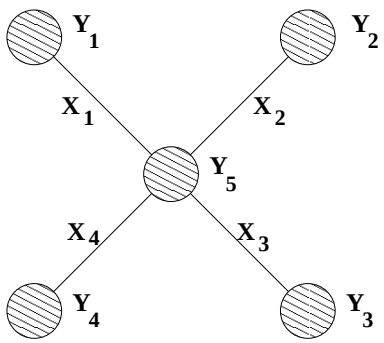

Figure 2: The star dependency scheme.

For completeness, we have also considered the case of independent random variables. In particular, we have assumed that each component ξ_i, $i = 1, \ldots, m$, can take the value 0 with probability $q_i = q_0^{1/i}$, where the value q_0 can be defined by the user.

We do not intend to use independence in our methods, by applying the transformations mentioned in Remark 1.2. The objective of this example is to see how general methods behave if the components of the random vector happen to be independent.

It is also not our intention to analyse the effect of different dependence structures on the performance of the methods. The only objective of these constructions is to generate non-trivial examples. Nevertheless, the design of meaningful test problems is a crucial issue: to our best knowledge, no test involving dependent discrete random variables has been ever presented in literature.

In order to design the test problems, we have considered, for each dependency model, different sizes of dependent groups. In the case of independency, we have as many groups of size 1 as the dimension m of the random vector.

We have generated three different classes of problems: *small*, *medium*, *large*. The size of the search space has been set to 50, 100 and 200, respectively. The characteristics of the test problems are summarized in Table 1, where we report the type of model, the group size and the number of groups.

Table 1: Characteristics of the Test Problems

Problem	**Model**	**Group's size**	**Number of groups**
Test1	Star	5	10
Test2	Circular	5	10
Test3	Star	10	5
Test4	Circular	10	5
Test5	Independent	1	50
Test6	Star	5	20
Test7	Circular	5	20
Test8	Star	10	10
Test9	Circular	10	10
Test10	Independent	1	100
Test11	Star	5	40
Test12	Circular	5	40
Test13	Star	10	20
Test14	Circular	10	20
Test15	Independent	1	200

All computational experiments have been carried out on a single processor R10K at 195 MHz of the supercomputer SGI Origin 2000, which has 4 nodes, with 4 Mb cache memory and 128 Mb RAM each. All codes have been implemented in the C language. The `cc` compiler with the default optimization level has been used.

We have enumerated the PEP for two different values of the probability level p. For each class of problems, Tables 2–4 report the number of PEP and the time (in seconds) required by the forward and backward enumeration procedures. We note that the number of PEP depends on choice of the distribution parameters, i.e. the values of α_i and λ_i in the circular and star dependency schemes, respectively, and the value of q_0 in the case of independent random variables. The choice of these parameters depends on the considered applications and it is left to the user. In our experiments the values of α_i and λ_i have been chosen in the range $[0.009, 0.99]$. In order to reflect the freedom in the user's choice, we have defined our tests in such a way that different groups may have different values of the parameters selected from the defined ranges. These values are reported in Tables 9–12. For each dependency scheme, the test problems are grouped according to the size s of each group. Note that in the

case of star dependency, we have to include an additional random variable, with parameter λ_{s+1}, denoting the center of the star.

In the case of independency the experiments have been performed by setting q_0 to 0.1 for all the tests.

Table 2: Results of the Enumeration Procedures for the Small Test Problems

Problem	**p**	**PEP**	**Forward**	**Backward**
Test1	0.95	74	1.01	1.06
Test1	0.90	1016	130.89	100.29
Test2	0.95	52	0.70	0.72
Test2	0.90	779	100.76	80.48.89
Test3	0.95	156	2.40	2.45
Test3	0.90	2490	296.15	210.99
Test4	0.95	107	2.00	2.17
Test4	0.90	1732	269.78	178.89
Test5	0.95	6	0.01	0.02
Test5	0.90	56	0.68	0.80

The numerical results shows that the number of PEP increases as p decreases. This is confirmed by our intuition: the higher the reliability imposed, the smaller the number of feasible "configurations". Other considerations concern the impact of the structure of the problem on the number of PEP. For a given size of the search space, we note that, at least for the test considered here, the larger the group's size the higher the number of p-efficient points. Nevertheless, the size of the search space influences the number of PEP: for a given group's size the larger the number of groups the higher the number of PEP. This behavior is even more relevant in the case of independent random variables. Here by increasing the size of the search space from 100 to 200 we pass (for $p = 0.95$) from 139 to 179564 PEP.

The computational comparison of the two enumeration strategies reveals the advantage of the backward enumeration procedure over the forward one. This can be explained by observing that the vast majority of PEP belongs to high levels (with large numbers of 1s). The backward strategy enumerates these points at early iterations, which substantially reduces the number of candidates at later stages. This does not apply to the forward strategy. Since we start from the lowest level (actually, the level m -$|l|$), our search through initial levels may be fruitless. We note that for Test15 with $p = 0.90$ both the procedures fail to enumerate all the PEP because of problems related to memory allocation (represented by "-" in Table 4).

Table 3: Results of the Enumeration Procedures for the Medium Test Problems

Problem	p	PEP	Forward	Backward
Test6	0.95	198	2.45	2.40
Test6	0.90	5289	192.08	137.20
Test7	0.95	144	3.82	3.30
Test7	0.90	5426	590.81	330.90
Test8	0.95	202	2.54	2.51
Test8	0.90	7923	585.65	368.20
Test9	0.95	182	2.89	2.70
Test9	0.90	12809	803.71	423.0
Test10	0.95	139	1.85	1.80
Test10	0.90	24253	1500.39	824.60

Table 4: Results of the Enumeration Procedures for the Large Test Problems

Problem	p	PEP	Forward	Backward
Test11	0.95	240	6.01	5.90
Test11	0.90	8466	622.70	479.20
Test12	0.95	208	7.00	5.56
Test12	0.90	13397	1071.53	729.00
Test13	0.95	274	7.84	7.20
Test13	0.90	9872	1136.87	752.90
Test14	0.95	282	10.56	8.7
Test14	0.90	14789	1610.38	914.99
Test15	0.95	179564	-	11132.90
Test15	0.90	-	-	-

5.3 Computational results of the solution strategies

In order to evaluate the performance of the solution procedures we have considered two test problems from the Beasley's OR library [6], namely `scp41` and `scp42`. These are deterministic set covering problems with $m = 200$ rows and $n = 1000$ columns. The probabilistic instances have been defined combining the deterministic problems with Test11–Test15 in Table 1, thus obtaining 10 different test problems. Other numerical results, collected on randomly generated test problems of smaller size, are reported in [3].

All the codes developed for the solution of the probabilistic set covering problem use the state-of-the-art LP solver CPLEX 6.5 [20]. The choice of this software, although probably not the most efficient for the solution of deterministic set covering instances (see [12] for an experimental comparison among of the main methods), is motivated by its high flexibility. Indeed, it contains callable libraries which allow to perform several operations (such as sensitivity analysis, determination of the dual multipliers, etc.) used in our algorithmic schemes.

As in any branch and bound scheme, the performance of the solution procedures is influenced by the initial upper bound value. It is obtained by solving an instance of the set covering problem having as the right-hand side the first PEP. We shall present and compare the results collected by using both the simple and dual heuristic strategies proposed in section 4.4 to address this issue. The upper bound value is used to decide to either eliminate or maintain a given candidate problem for further exploration. The "elimination" test is based on the use of a lower bound on the optimal solution of the candidate problem; the bound which is obtained by solving a relaxation of the problem.

The choice of the type of relaxation represents another crucial issue in the solution method. We have used a two stage relaxation. At first, we compute a dual bound using the Lagrange multipliers u obtained by the linear relaxation of the set covering problem (1). Its round-up value is used to perform the first selection: we solve the linear relaxation only for problems with the dual bound better than the best solution found so far. We note that the solution of the linear relaxations can be efficiently carried out by the dual simplex method starting from the optimal basis of the previous problem. The use of the two-stage relaxation allows to achieve better performance especially in the hybrid solution methods. In this case, the number of relaxations to be solved is larger than those handled in the complete strategies because both PEP and potential candidates have to considered.

The use of preprocessing or presolving techniques represents another crucial issue in improving the performance of the solution methods. Their aim is to shrink the size of the problem with the resulting benefit of cutting the solution time. This is even more relevant for the probabilistic version of the set covering problem, because several instances of the deterministic problem have to be solved. The literature dealing with preprocessing techniques for the deterministic set covering problem is quite rich (see [10] and the references herein for a detailed survey).

Particularly relevant are techniques aiming at reduction of dominated/duplicated columns and rows. We have implemented both techniques (for columns and for rows) as described by Beasley in [4]. The column reduction technique has been applied to the original deterministic

set covering problems `scp41`, `scp42` used in our tests. The reduction in the number of the columns is remarkable: the number of deleted columns is 827 and 781 for the two problems, respectively. On the other hand, the row reduction technique appears to be completely ineffective, at least for the two test problems considered here.

Reduced cost fixing is another useful preprocessing technique. It consists in fixing to zero the value of each variable whose reduced cost is greater than the difference between current upper and lower bounds on the optimal value of the problem (see, for example, [4]). In the case of the probabilistic set covering problem this idea can be applied at two different levels: in the solution of a given instance of the set covering problem and within the whole solution procedure. The first level is not the main concern here: we insert the set covering instance in a black box and we get the solution. The second level is more interesting. Within the hybrid solution method we can maintain a "global" gap defined as difference of the current incumbent value and a global lower bound computed on the basis of the bounds associated to the potential candidates at a given level. The use of the global gap allows to deal with a unique matrix constraint during the computation. The implementation of the reduced cost fixing requires a computational cost which is not rewarded, at least for the test problems considered, by the reduction in the problem's size produced by the use of this technique. The numerical results presented below have been collected including only column reduction and the other techniques offered by CPLEX.

Our final remark concerns the enumeration strategies used in the solution methods. On the basis of the computational results presented in the previous subsection, it is evident that the backward enumeration method outperforms, at least on the test problems considered here, the forward one. Thus in the following we shall present the results obtained by using the backward strategy.

5.3.1 Numerical results of the complete solution methods

Tables 5–6 report the computational results collected by using the complete method with the simple and dual heuristics for determining the first PEP. Here z_I denotes the initial upper bound value, PEP the total number of p-efficient points, LP the number of linear relaxations and z^* the optimal objective value of the probabilistic set covering problem. The last column reports the solution time measured in seconds. The test have been performed for two values of the probability level p. We note that the number of integer problems solved has not been reported because for both `scp41` and `scp42` the linear relaxations provide an integer solution.

The computational results show that the complete method with dual heuristic outperforms, at least on the tests considered here, the version with the simple heuristic. Given the nature of the method, both the simple and the dual versions enumerate all the PEP. However, the number of solved problems is smaller in the dual version. The advantage related to the solution of a lower number of problems is more relevant as the solution time per problem increases. Thus better performance can be achieved for more computationally demanding problems.

Table 5: Numerical results of the complete strategy with simple heuristic

Problem	**p**	z_I	**PEP**	**LP**	z^*	**Time**
Test11.1	0.95	428	240	9	419	6.89
Test11.1	0.90	424	8466	258	413	502.66
Test12.1	0.95	429	208	42	416	9.52
Test12.1	0.90	424	13397	128	410	740.71
Test13.1	0.95	427	274	41	416	10.97
Test13.1	0.90	420	9872	80	402	760.29
Test14.1	0.95	418	282	10	410	9.78
Test14.1	0.90	410	14789	492	404	958.46
Test15.1	0.95	425	179564	250	357	11160.41
Test15.1	0.90	-	-	-	-	-
Test11.2	0.95	516	240	34	510	9.14
Test11.2	0.90	510	8466	959	484	578.49
Test12.2	0.95	515	208	11	503	6.73
Test12.2	0.90	513	13397	339	494	759.69
Test13.2	0.95	492	274	31	487	10.17
Test13.2	0.90	489	9872	150	447	765.68
Test14.2	0.95	507	282	33	497	11.85
Test14.2	0.90	507	14789	531	479	962.87
Test15.2	0.95	498	179654	1212	443	11254.13
Test15.2	0.90	-	-	-	-	-

5.3.2 Numerical results of the hybrid solution methods

Tables 7–8 report the numerical results collected by using the hybrid method with the simple and dual heuristics. Here PEP refers to the number of the p-efficient points enumerated by the algorithm. This number is always smaller than the total number of PEP enumerated by the complete methods. For some test problems, the reduction can be huge. For example, for Test11.1 with a probability level p of 0.90 the hybrid method with simple heuristic enumerates only 11% of the total number of PEP. The advantage of a reduced enumeration is more relevant as the number of PEP increases. As the extreme case, we cite Test15 with $p = 0.90$. This problem can be successfully solved only by the hybrid method, whereas the complete algorithm fails because of memory requirements.

We note that the number of enumerated points is influenced by the initial upper bound value: the better the value, the smaller the portion of the graph explored during the computation. For example, for Test 13.2 with $p = 0.90$ the number of PEP decreases from 1745

Table 6: Numerical results of the complete strategy with dual heuristic

Problem	**p**	z_I	**PEP**	**LP**	z^*	**Time**
Test11.1	0.95	419	240	7	419	6.71
Test11.1	0.90	414	8466	170	413	495.59
Test12.1	0.95	420	208	32	416	8.72
Test12.1	0.90	413	13397	83	410	736.65
Test13.1	0.95	416	274	11	416	8.12
Test13.1	0.90	402	9872	7	402	752.98
Test14.1	0.95	418	282	10	410	9.78
Test14.1	0.90	410	14789	492	404	958.46
Test15.1	0.95	418	179564	188	357	11153.59
Test15.1	0.90	-	-	-	-	-
Test11.2	0.95	510	240	19	510	7.79
Test11.2	0.90	484	8466	80	484	487.88
Test12.2	0.95	505	208	7	503	6.30
Test12.2	0.90	499	13397	252	494	751.86
Test13.2	0.95	492	274	31	487	10.17
Test13.2	0.90	449	9872	24	447	755.24
Test14.2	0.95	497	282	15	497	10.23
Test14.2	0.90	485	14789	322	479	944.15
Test15.2	0.95	451	179654	746	443	11207.54
Test15.2	0.90	-	-	-	-	-

to 740.

Table 7: Numerical results of the hybrid strategy with simple heuristic

Problem	**p**	z_I	**PEP**	**LP**	z^*	**Time**
Test11.1	0.95	425	209	15	419	6.75
Test11.1	0.90	425	935	74	413	60.87
Test12.1	0.95	429	165	34	416	7.76
Test12.1	0.90	424	6387	653	410	448.52
Test13.1	0.95	427	79	8	416	3.99
Test13.1	0.90	427	463	23	402	37.90
Test14.1	0.95	422	252	10	410	8.83
Test14.1	0.90	422	5284	502	404	380.76
Test15.1	0.95	427	140274	564	357	8780.02
Test15.1	0.90	382	529814	2769	343	33153.07
Test11.2	0.95	516	221	22	510	7.68
Test11.2	0.90	516	1881	136	484	123.41
Test12.2	0.95	515	106	6	503	4.02
Test12.2	0.90	511	6249	214	494	400.20
Test13.2	0.95	510	217	29	487	8.65
Test13.2	0.90	514	1745	215	447	155.64
Test14.2	0.95	516	297	44	497	13.32
Test14.2	0.90	516	13267	542	479	911.31
Test15.2	0.95	471	130049	967	443	8170.48
Test15.2	0.90	458	389886	3712	431	24581.25

On the basis of these computational results we can draw the following empirical conclusions. The hybrid solution methods seem to outperform the complete ones. The main bottleneck of the complete methods is related to the enumeration of the whole set of the PEP. At least for the test problems considered here, enumeration can require even more than 90% of the entire solution time. This percentage may be smaller when considering more computationally demanding problems. Anyway, the savings introduced by improving the solution phase seem to be limited. The number of PEP enumerated by the hybrid methods is much smaller than the number generated by the complete ones. Furthermore, this number can be reduced if a good initial upper bound value is available. The dual heuristic method appears to be a good choice here.

Table 8: Numerical results of the hybrid strategy with dual heuristic

Problem	**p**	z_I	**PEP**	**LP**	z^*	**Time**
Test11.1	0.95	419	198	6	419	5.67
Test11.1	0.90	425	935	74	413	60.87
Test12.1	0.95	418	155	30	416	7.33
Test12.1	0.90	413	5192	650	410	383.02
Test13.1	0.95	417	65	6	416	2.99
Test13.1	0.90	411	425	6	402	33.87
Test14.1	0.95	418	250	7	410	8.63
Test14.1	0.90	422	1294	502	404	380.76
Test15.1	0.95	380	110121	158	357	6945.08
Test15.1	0.90	372	256581	2185	343	16248.31
Test11.2	0.95	516	221	22	510	7.68
Test11.2	0.90	506	821	76	484	55.61
Test12.2	0.95	505	88	5	503	3.12
Test12.2	0.90	498	5826	208	494	372.28
Test13.2	0.95	488	213	24	487	8.34
Test13.2	0.90	479	740	52	447	62.58
Test14.2	0.95	501	258	34	497	11.77
Test14.2	0.90	515	1321	504	479	904.28
Test15.2	0.95	450	100107	856	443	6300.35
Test15.2	0.90	458	389886	3712	431	24581.25

6 Conclusions

Probabilistic set covering problems possess interesting structural properties that can be efficiently used in the solution methods. The key notion is that of a p-efficient point: a vector that can be substituted for the right hand side of the set covering problem to obtain the solution of the probabilistic problem. There are many p-efficient points, and all solution methods considered here are based on their complete or partial enumeration.

We have considered two classes of enumeration and solution methods: forward and backward. The computational results indicate that the backward methods, using lower bounds derived from conditional marginals, have smaller numbers of LPs and IPs solved, points generated, and a shorter solution time, than forward methods.

We have also proposed and compared a number of heuristic methods for generating initial upper bounds. Among them, the dual heuristic, which aims at finding the p-efficient point with the lowest dual lower bound appears very attractive.

Several recent achievements in stochastic integer programming offer promising directions of research for our problem. One of them is the application of Gröbner bases of polynomial ideals [19]. This is a method from computational algebra that has been applied in [18] for the solution of integer problems with a varying right hand side. The integer program is translated into a subalgebra membership problem in a ring of polynomials, and a specialized algorithm can be developed for solving this question. The Gröbner basis of a certain polynomial ideal enters as the essential ingredient into that procedure. The key feature of this approach is that for various right hand sides in the initial integer program always the same Gröbner basis applies. Consequently, once the basis is found, solving the integer program for another right hand side amounts just to another generalized division of multivariate polynomials. Clearly, finding the Gröbner basis is difficult, but so is our problem, and this possibility definitely deserves further studies.

Table 9: Distribution Parameters of Test1,Test6,Test11

Group	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
1	0.010	0.010	0.090	0.700	0.090	0.070
2	0.020	0.070	0.050	0.070	0.090	0.070
3	0.070	0.070	0.070	0.090	0.010	0.010
4	0.010	0.020	0.010	0.070	0.020	0.010
5	0.900	0.010	0.040	0.010	0.010	0.010
6	0.010	0.180	0.190	0.080	0.010	0.010
7	0.070	0.500	0.020	0.550	0.080	0.020
8	0.670	0.500	0.120	0.040	0.080	0.020
9	0.080	0.720	0.440	0.650	0.060	0.110
10	0.091	0.110	0.110	0.110	0.110	0.120
1	0.810	0.010	0.100	0.051	0.010	0.010
2	0.200	0.090	0.010	0.700	0.300	0.300
3	0.010	0.810	0.900	0.810	0.050	0.001
4	0.009	0.890	0.710	0.020	0.061	0.061
5	0.100	0.010	0.100	0.020	0.300	0.500
6	0.200	0.100	0.110	0.110	0.010	0.810
7	0.910	0.910	0.010	0.090	0.090	0.090
8	0.005	0.005	0.805	0.090	0.007	0.900
9	0.020	0.020	0.910	0.910	0.010	0.010
10	0.900	0.140	0.150	0.150	0.010	0.700
11	0.600	0.010	0.040	0.610	0.071	0.911
12	0.018	0.010	0.010	0.010	0.010	0.615
13	0.011	0.010	0.080	0.770	0.009	0.011
14	0.011	0.910	0.011	0.011	0.011	0.089
15	0.010	0.710	0.700	0.010	0.050	0.001
16	0.909	0.890	0.010	0.020	0.860	0.860
17	0.100	0.010	0.100	0.020	0.300	0.500
18	0.500	0.200	0.100	0.110	0.810	0.810
19	0.010	0.910	0.010	0.810	0.090	0.090
20	0.090	0.005	0.005	0.990	0.007	0.907
1	0.810	0.010	0.100	0.051	0.010	0.810
2	0.200	0.090	0.010	0.700	0.300	0.300
3	0.810	0.010	0.100	0.950	0.010	0.810
4	0.200	0.890	0.010	0.700	0.300	0.010
5	0.010	0.810	0.900	0.810	0.850	0.100
6	0.800	0.890	0.710	0.020	0.060	0.060
7	0.810	0.810	0.900	0.810	0.050	0.001
8	0.809	0.890	0.710	0.020	0.061	0.960
9	0.100	0.010	0.110	0.020	0.800	0.500
10	0.200	0.100	0.110	0.110	0.810	0.810
11	0.100	0.910	0.100	0.020	0.300	0.800
12	0.210	0.100	0.110	0.110	0.010	0.010
13	0.910	0.910	0.010	0.890	0.990	0.890
14	0.900	0.005	0.805	0.990	0.007	0.900
15	0.910	0.910	0.010	0.890	0.890	0.890
16	0.005	0.005	0.800	0.090	0.007	0.900
17	0.890	0.020	0.910	0.910	0.010	0.502
18	0.500	0.200	0.100	0.110	0.810	0.710
19	0.900	0.140	0.150	0.150	0.010	0.700
20	0.020	0.020	0.810	0.910	0.010	0.010
21	0.900	0.140	0.150	0.150	0.010	0.700
22	0.600	0.010	0.840	0.610	0.870	0.910
23	0.018	0.010	0.011	0.011	0.910	0.610
24	0.600	0.010	0.041	0.611	0.970	0.910
25	0.018	0.810	0.011	0.011	0.011	0.615
26	0.011	0.010	0.089	0.770	0.009	0.011
27	0.011	0.910	0.011	0.811	0.811	0.189
28	0.011	0.010	0.089	0.770	0.009	0.011
29	0.011	0.910	0.011	0.011	0.911	0.189
30	0.010	0.710	0.730	0.810	0.050	0.101
31	0.909	0.890	0.810	0.920	0.861	0.861
32	0.810	0.710	0.700	0.810	0.050	0.001
33	0.900	0.890	0.010	0.020	0.861	0.061
34	0.100	0.010	0.100	0.020	0.301	0.500
35	0.500	0.200	0.100	0.110	0.810	0.810
36	0.100	0.010	0.100	0.020	0.305	0.500
37	0.500	0.200	0.100	0.110	0.810	0.810
38	0.810	0.910	0.010	0.810	0.090	0.890
39	0.890	0.005	0.005	0.990	0.007	0.907
40	0.900	0.910	0.010	0.810	0.090	0.090

Table 10: Distribution Parameters of Test2,Test7,Test12

Group	α_1	α_2	α_3	α_4	α_5
1	0.010	0.030	0.060	0.050	0.010
2	0.010	0.070	0.090	0.010	0.030
3	0.010	0.007	0.010	0.005	0.010
4	0.090	0.008	0.010	0.020	0.060
5	0.050	0.010	0.020	0.021	0.030
6	0.010	0.007	0.010	0.005	0.010
7	0.090	0.008	0.010	0.020	0.060
8	0.050	0.010	0.020	0.020	0.030
9	0.050	0.020	0.010	0.110	0.030
10	0.020	0.070	0.010	0.080	0.020
1	0.010	0.010	0.080	0.051	0.010
2	0.202	0.002	0.002	0.102	0.002
3	0.080	0.080	0.010	0.810	0.080
4	0.810	0.050	0.001	0.009	0.891
5	0.710	0.020	0.061	0.061	0.101
6	0.010	0.100	0.020	0.300	0.502
7	0.080	0.100	0.081	0.100	0.011
8	0.810	0.010	0.910	0.010	0.090
9	0.090	0.090	0.005	0.005	0.805
10	0.090	0.007	0.900	0.020	0.020
11	0.910	0.910	0.010	0.010	0.902
12	0.140	0.151	0.150	0.010	0.702
13	0.601	0.011	0.041	0.611	0.071
14	0.911	0.018	0.011	0.011	0.011
15	0.011	0.615	0.011	0.011	0.089
16	0.770	0.009	0.011	0.011	0.911
17	0.011	0.011	0.011	0.089	0.010
18	0.710	0.700	0.010	0.050	0.001
19	0.909	0.891	0.010	0.020	0.861
20	0.861	0.100	0.012	0.103	0.020
1	0.010	0.410	0.421	0.051	0.010
2	0.811	0.200	0.090	0.810	0.702
3	0.301	0.300	0.810	0.010	0.100
4	0.951	0.010	0.810	0.200	0.890
5	0.010	0.700	0.304	0.010	0.010
6	0.010	0.080	0.810	0.051	0.101
7	0.009	0.090	0.710	0.020	0.061
8	0.061	0.010	0.010	0.901	0.010
9	0.050	0.801	0.009	0.890	0.810
10	0.820	0.261	0.861	0.100	0.810
11	0.101	0.021	0.800	0.501	0.200
12	0.010	0.010	0.010	0.010	0.010
13	0.100	0.910	0.100	0.021	0.300
14	0.801	0.200	0.100	0.110	0.110
15	0.010	0.010	0.910	0.910	0.010
16	0.890	0.990	0.890	0.905	0.005
17	0.805	0.990	0.007	0.901	0.911
18	0.910	0.010	0.890	0.890	0.890
19	0.005	0.805	0.805	0.091	0.007
20	0.901	0.890	0.021	0.911	0.910
21	0.010	0.710	0.901	0.140	0.151
22	0.150	0.010	0.701	0.020	0.021
23	0.810	0.910	0.011	0.010	0.900
24	0.140	0.150	0.150	0.011	0.700
25	0.601	0.011	0.841	0.611	0.871
26	0.911	0.018	0.011	0.011	0.011
27	0.911	0.615	0.601	0.011	0.041
28	0.611	0.971	0.911	0.018	0.811
29	0.011	0.011	0.011	0.615	0.011
30	0.011	0.089	0.770	0.009	0.011
31	0.011	0.911	0.011	0.811	0.811
32	0.189	0.011	0.011	0.089	0.770
33	0.709	0.711	0.011	0.911	0.011
34	0.011	0.911	0.189	0.010	0.710
35	0.700	0.810	0.050	0.101	0.909
36	0.890	0.810	0.921	0.861	0.861
37	0.810	0.710	0.700	0.810	0.050
38	0.001	0.909	0.890	0.010	0.020
39	0.861	0.061	0.100	0.010	0.100
40	0.020	0.300	0.501	0.500	0.200

Table 11: Distribution Parameters of Test3,Test8,Test13

Group	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}	λ_{11}
1	0.010	0.010	0.090	0.700	0.090	0.020	0.070	0.050	0.070	0.090	0.010
2	0.070	0.070	0.070	0.090	0.010	0.010	0.020	0.010	0.070	0.020	0.010
3	0.900	0.010	0.040	0.010	0.010	0.010	0.180	0.190	0.080	0.010	0.010
4	0.070	0.500	0.020	0.550	0.080	0.670	0.500	0.120	0.040	0.080	0.020
5	0.080	0.720	0.440	0.650	0.060	0.090	0.110	0.110	0.110	0.110	0.110
1	0.100	0.201	0.100	0.051	0.010	0.010	0.202	0.090	0.010	0.300	0.200
2	0.010	0.100	0.050	0.010	0.050	0.010	0.090	0.900	0.710	0.020	0.090
3	0.100	0.010	0.100	0.020	0.300	0.500	0.020	0.100	0.110	0.010	0.010
4	0.010	0.910	0.010	0.090	0.190	0.090	0.005	0.050	0.090	0.007	0.900
5	0.200	0.020	0.010	0.010	0.100	0.010	0.900	0.040	0.150	0.010	0.700
6	0.601	0.011	0.041	0.611	0.071	0.011	0.018	0.011	0.011	0.011	0.615
7	0.011	0.011	0.089	0.770	0.009	0.011	0.011	0.011	0.011	0.011	0.89
8	0.010	0.040	0.700	0.010	0.050	0.001	0.009	0.890	0.010	0.020	0.061
9	0.100	0.010	0.100	0.020	0.300	0.500	0.500	0.200	0.100	0.110	0.810
10	0.010	0.910	0.010	0.810	0.090	0.090	0.090	0.005	0.005	0.090	0.007
1	0.801	0.910	0.100	0.051	0.010	0.810	0.200	0.090	0.010	0.700	0.901
2	0.300	0.810	0.810	0.100	0.951	0.010	0.810	0.200	0.890	0.810	0.702
3	0.300	0.910	0.910	0.810	0.900	0.810	0.850	0.101	0.809	0.990	0.710
4	0.820	0.061	0.061	0.810	0.810	0.900	0.810	0.050	0.801	0.809	0.890
5	0.710	0.020	0.061	0.961	0.100	0.010	0.100	0.020	0.800	0.500	0.201
6	0.100	0.110	0.110	0.810	0.810	0.100	0.910	0.100	0.020	0.302	0.800
7	0.201	0.100	0.110	0.110	0.010	0.010	0.910	0.910	0.010	0.890	0.990
8	0.890	0.905	0.005	0.805	0.990	0.007	0.900	0.910	0.910	0.010	0.890
9	0.890	0.890	0.805	0.005	0.805	0.890	0.807	0.900	0.890	0.820	0.910
10	0.910	0.810	0.710	0.900	0.140	0.150	0.150	0.810	0.702	0.820	0.820
11	0.810	0.910	0.010	0.010	0.901	0.140	0.150	0.150	0.810	0.701	0.601
12	0.911	0.841	0.611	0.871	0.911	0.018	0.711	0.011	0.711	0.911	0.615
13	0.601	0.011	0.041	0.611	0.971	0.911	0.018	0.811	0.011	0.011	0.011
14	0.615	0.011	0.011	0.089	0.770	0.009	0.011	0.011	0.911	0.011	0.811
15	0.811	0.189	0.011	0.011	0.089	0.770	0.009	0.011	0.011	0.911	0.911
16	0.911	0.911	0.189	0.010	0.710	0.700	0.810	0.050	0.101	0.909	0.890
17	0.810	0.920	0.861	0.861	0.810	0.710	0.701	0.810	0.050	0.001	0.909
18	0.890	0.010	0.021	0.861	0.061	0.100	0.010	0.100	0.021	0.300	0.500
19	0.500	0.201	0.100	0.110	0.811	0.810	0.100	0.010	0.100	0.021	0.300
20	0.500	0.500	0.200	0.100	0.110	0.810	0.810	0.810	0.912	0.010	0.810

Table 12: Distribution Parameters of Test4,Test9,Test14

Group	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
1	0.010	0.010	0.100	0.050	0.010	0.010	0.200	0.090	0.010	0.030
2	0.010	0.070	0.010	0.050	0.001	0.009	0.090	0.010	0.020	0.060
3	0.100	0.010	0.10	0.020	0.030	0.050	0.020	0.010	0.110	0.010
4	0.910	0.910	0.010	0.090	0.090	0.090	0.005	0.090	0.007	0.090
5	0.002	0.010	0.010	0.010	0.010	0.020	0.014	0.015	0.010	0.010
1	0.010	0.010	0.100	0.051	0.010	0.010	0.201	0.090	0.010	0.300
2	0.300	0.010	0.810	0.702	0.010	0.850	0.801	0.009	0.090	0.710
3	0.020	0.061	0.101	0.010	0.100	0.020	0.300	0.502	0.200	0.100
4	0.110	0.010	0.810	0.910	0.910	0.012	0.890	0.090	0.091	0.005
5	0.005	0.090	0.007	0.901	0.020	0.021	0.010	0.010	0.810	0.810
6	0.900	0.140	0.150	0.010	0.702	0.601	0.011	0.041	0.611	0.071
7	0.011	0.018	0.011	0.011	0.011	0.615	0.011	0.011	0.089	0.770
8	0.009	0.011	0.011	0.011	0.011	0.011	0.089	0.010	0.710	0.700
9	0.010	0.050	0.001	0.009	0.890	0.010	0.021	0.861	0.100	0.010
10	0.100	0.020	0.301	0.500	0.501	0.200	0.100	0.110	0.810	0.010
1	0.010	0.010	0.100	0.051	0.010	0.010	0.200	0.090	0.010	0.700
2	0.900	0.301	0.010	0.010	0.102	0.951	0.010	0.810	0.290	0.890
3	0.810	0.701	0.300	0.010	0.010	0.011	0.900	0.010	0.050	0.001
4	0.809	0.990	0.010	0.020	0.061	0.061	0.810	0.810	0.901	0.810
5	0.050	0.001	0.009	0.090	0.710	0.020	0.061	0.961	0.100	0.010
6	0.100	0.020	0.801	0.501	0.201	0.100	0.110	0.110	0.810	0.810
7	0.020	0.910	0.100	0.020	0.300	0.801	0.200	0.010	0.010	0.010
8	0.010	0.010	0.910	0.910	0.010	0.890	0.990	0.890	0.905	0.005
9	0.805	0.990	0.007	0.900	0.910	0.910	0.010	0.890	0.890	0.890
10	0.005	0.005	0.805	0.890	0.807	0.901	0.891	0.820	0.010	0.010
11	0.810	0.710	0.900	0.141	0.151	0.150	0.812	0.701	0.820	0.820
12	0.810	0.910	0.010	0.010	0.900	0.140	0.150	0.150	0.810	0.700
13	0.601	0.911	0.841	0.611	0.871	0.911	0.018	0.711	0.011	0.711
14	0.911	0.615	0.601	0.011	0.041	0.611	0.971	0.911	0.018	0.811
15	0.011	0.011	0.011	0.615	0.011	0.011	0.089	0.770	0.009	0.011
16	0.011	0.911	0.011	0.811	0.811	0.189	0.011	0.011	0.089	0.770
17	0.009	0.011	0.011	0.911	0.911	0.911	0.911	0.189	0.010	0.710
18	0.700	0.810	0.050	0.101	0.909	0.890	0.810	0.921	0.861	0.861
19	0.810	0.710	0.700	0.810	0.050	0.001	0.909	0.890	0.010	0.020
20	0.861	0.061	0.100	0.010	0.101	0.020	0.301	0.500	0.500	0.020

References

[1] E. Balas, A Class of Location, Distribution and Scheduling Problems: Modeling and Solution Methods. in P. Gray and L. Yuanzhang (eds.), *Proceedings of the Chinese-U.S. Symposium on System Analysis*, J. Wiley and Sons (1983).

[2] E. Balas, M.C. Carrera, A Dynamic Subgradient-Based Branch-and-Bound Procedure for Set Covering. *Operations Research* 44 (1996), 875–890.

[3] P. Beraldi, Programmazione Stocastica Intera sotto Vincoli Probabilistici, PhD Thesis, University of Calabria, (1999).

[4] J.E. Beasley, An Algorithm for Set Covering Problems. *European Journal of Operational Research*, 31 (1987), 85–93.

[5] J.E. Beasley, A Lagrangian Heuristic for Set Covering Problems. *Naval Research Logistics*, 37 (1990), 151–164.

[6] J.E. Beasley, OR–Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Sociecty* 41 (1990), 1069–1072.

[7] J.E. Beasley, P.C. Chu, A Genetic Algorithm for the Set Covering Problem. *European Journal of Operational Reserach* 94 (1996), 392–404.

[8] J.E. Beasley, K. Jörnesten, Enhancing an Algorithm for Set Covering Problems. *European Journal of Operational Research* 58 (1992), 293–300.

[9] J.R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer-Verlag, New York, NY, 1997.

[10] R. Borndörfer, Aspects of Set Packing, Partitioning and Covering, Shaker Verlag, Aachen, (1998).

[11] A. Caprara, M. Fischetti, P. Toth, A Heuristic Method for the Set Covering Problem. *Operations Research*, Vol. 47 (1999), 730–743.

[12] A. Caprara, M. Fischetti, P. Toth, Algorithms for Set Covering Problem. Working Paper, DEIS, University of Bologna, (1998), to appear on *Annals of Operations Research*.

[13] A. Caprara, M. Fischetti, P. Toth, D. Vigo, P.L. Guida, Algorithms for Railway Crew Management. *Mathematical Programming* 79 (1997), 125–141.

[14] S. Ceria, P. Nobili, A. Sassano, A Lagrangian-Based Heuristic for Large-Scale Set Covering Problems. *Mathematical Programming* 81 (1998), 215-228.

[15] S. Ceria, P. Nobili, A. Sassano, Set Covering Problem. in M. Dell'Amico, F. Maffioli and A. Sassano (eds.) *Annotated Bibliographies in Combinatorial Optimization*, J. Wiley and Sons (1997), 415–428.

[16] H.D. Chu, E. Gelman, E.L. Johnson, Solving Large Scale Crew Scheduling Problems. *European Journal of Operational Research* 97 (1997), 260–268.

[17] V. Chvátal, A Greedy Heuristic for the Set Covering Problem. *Mathematics of Operations Research* 4 (1979), 233–235.

[18] P. Conti, C. Traverso, Buchberger Algorithm and Integer Programming. *Proceeding AAECC-9* (New Orleans), Lecture Notes in Computer Science 539, (1991) 130-139, Berlin.

[19] D. Cox, J. Little, D. O'Shea, Ideals, Varieties and Algorithms. Springer-Verlag, New York (1992).

[20] CPLEX Optitmization, Inc., Incline Village, NV. *ILOG CPLEX 6.5: User's Manual*, 1999.

[21] D. Dentcheva, A. Prékopa, A. Ruszczyński, Concavity and Efficient Points of Discrete Distributions in Probabilistic Programming, accepted for publication in *Mathematical Programming*.

[22] M. Gendreau, G. Laporte and R. Séguin, Stochastic Vehicle Routing, *European Journal of Operational Research*, 88 (1996) 3–12.

[23] S. Haddadi, Simple Lagrangian Heuristic for the Set Covering Problem. *European Journal of Operational Research* 97 (1997), 200–204.

[24] L.W. Jacobs, M.J. Brusco, A Local Search Heuristic for Large Set Covering Problems. *Naval Research Logistics*, 52 (1995), 1129-1140.

[25] G. Laporte, F.V. Louveaux and H. Mercure, Models and Exact Solutions for a Class of Stochastic Location–Routing Problems, *European Journal of Operational Research*, 39 (1989), 71-78.

[26] L.A.N. Lorena, F.B. Lopes, A Surrogate Heuristic for Set Covering Problems. *European Journal of Operational Research* 79 (1994), 138–150.

[27] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization. Wiley-Interscience New York (1990).

[28] P. Nobili, A. Sassano, A Separation Routine for Set Covering Polytope. in E. Balas, G. Cornuejols, R. Kannan (eds.), *Integer Programming and Combinatorial Optimization*, Proceedings of the 2nd IPCO Conference, Carnegie-Mellon University Press (1992).

[29] A. Prékopa, 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 (1990), 441–461.

[30] A. Prékopa, Stochastic Programming. Kluwer Scientific Publisher, Boston (1995).

[31] A. Prékopa, B. Vizvári, T. Badics, Programming Under Probabilistic Constraint with Discrete Random Variable. in F. Giannesi et al. (eds.), *New Trends in Mathematical Programming*, Kluwer Academic Publisher (1998), 235–255.

[32] S. Sen, Relaxations for the Probabilistically Constrained Programs with Discrete Random Variables. *Operations Research Letters* 11 (1992), 81–86.