Improving Risk Management

RIESGOS-CM

Análisis, Gestión y Aplicaciones

P2009/ESP-1685

Technical Report 2010.09

Stochastic Set PStochastic Set Packing Problemacking Problem

Laureano F. Escudero, Mercedes Landete, Antonio M. Rodríguez-Chía

http://www.analisisderiesgos.org

Stochastic Set Packing Problem

Laureano F. Escudero[a] Mercedes Landete[b]

Antonio M. Rodríguez-Chía[c]

[a]Dpto. Estadística e Investigación Operativa. Univ. Rey Juan Carlos, Spain

[b]Centro de Investigación Operativa. Univ. Miguel Hernández of Elche, Spain

[c]Dpto. Estadística e Investigación Operativa. Univ. Cádiz, Spain

June 18, 2010

Abstract

In this paper a stochastic version of the Set Packing Problem, SPP, is studied via scenario analysis. We consider a one-stage recourse approach to deal with the uncertainty in the coefficients. This consists of maximizing in the stochastic SPP a composite function of the expected value minus the weighted risk of obtaining a scenario whose objective function value is worse than a given threshold. The splitting variable representation is decomposed by dualizing the nonanticipativity constraints that link the deterministic SPP with a 0-1 knapsack problem for each scenario under consideration. As a result a (structured) larger pure 0-1 model is created. We present several procedures for obtaining good feasible solutions, as well as a preprocessing approach for fixing variables. The Lagrange multipliers updating is performed by using the Volume Algorithm. Computational experience is reported for a broad variety of instances, which shows that the new approach usually outperforms a state-of-the-art optimization engine, producing a comparable optimality gap with smaller (several orders of magnitude) computing time.

Keywords: Assignment, combinatorial optimization, set packing, stochastic 0-1 programming, simple recourse, Lagrangian decomposition, Volume Algorithm.

1 Introduction

Frequently, a number of problems that we find in the real-world present element which escape the control of the decision-maker. The uncertainty often originates from the impossibility of choosing one scenario where the problems occur, and then, it is unclear which objective function should be optimized. However, some information could become available by estimating a weight (probability) for each of the potential scenarios. This circumstance allows us to use Stochastic Integer Programming, SIP, for solving one-stage problems (also known as simple recourse problems), see [7].

Indeed, these types of problems, where different scenarios may occur, are formulated in SIP through a *Deterministic Equivalent Model*, DEM. This is a deterministic problem whose optimal solutions either provide an averaged objective value among all the possible scenarios or "good" solutions for the original problem because, for instance, they reduce the probability of the occurrence of non-desired scenarios. In the first case, most of the approaches consider as objective function the expected value alone and in the second one, some of the approaches try to avoid the non-desired scenarios by using some rejecting risk measure. In this sense, it has been widely studied mean-risk measures by considering semi-deviations [37], reject probabilities (so-called excess probabilities for minimization problems) [47] and conditional value-at-risk [41, 48] as risk measure-based functions to optimize.

In the literature we can find different solution procedures for these models; for instance, Benders decomposition [2, 4, 7, 11, 19, 18, 20, 29, 30, 44, 51], Lagrangian decomposition [12, 17, 23, 26, 28, 43, 46, 47, 48, 54], disjunctive decomposition [35, 50], stochastic branch-and-cut [36, 49, 55] or branch-and-fix coordination [1] approaches. See also [20, 45, 52] for solving the two-stage problem having mixed-integer first- and second-stage variables.

In this paper, we will analyze a stochastic version of the Set Packing Problem whose deterministic formulation is given by:

$$\begin{aligned} \text{(SPP)} \quad \max \quad & cx \\ \text{s.t.} \quad & Ax \leq 1, \\ & x \in \{0,1\}^m. \end{aligned}$$

where $c \in \mathbb{R}^m_+$ and $A = (a_{ij})_{i \in I, j \in J}$ with $a_{ij} \in \{0,1\}$ $\forall (i,j) \in I \times J = \{1,\ldots,n\} \times \{1,\ldots,m\}$.

This problem has numerous applications and its polyhedra structure has been broadly studied during the last 40 years, see e.g., [34]. We can find in the literature different approaches to solve either the general problem (see [9, 38]) or particular and important versions such as assignment problems, combinatorial auction problems [13], or location problems (see [8] for the Simple Plant Location Problem, [10] for the Hub Location Problem or [53] for the Reliability Location Problem) among others. However, in spite of what has been extensively studied for the Set Packing problem, we only find in the literature some attempts to deal with a stochastic version of this problem where A is a random matrix, (see e.g., [16, 22, 40]).

Unlike the SPP where the coefficients of the objective function was assumed to be known, we were motivated by real problems where these coefficients are uncertain. Consider, for example, the case of a farmer who specializes in raising corn. He has three different types of corn seed and three pieces of land with the same size. Based on past experience, the farmer knows that the mean yield of this three pieces of land can be different by two main reasons: the type of seed corn and the weather conditions. In particular, each type of corn has a mean yield depending on the landscape where it has been allocated. Moreover, this mean yield could be below average for dry years, on average for normal years and above average for raining years. The farmer has a minimum requirement to cattle feed, such that, any production in excess of the feeding requirement would be sold, otherwise the farmer must buy corn. Therefore, since weather conditions cannot be accurately predicted six months ahead, the farmer must make up his mind without perfect information. In this case, the constraints of the problem are given by a classical assignment problem (three types of corn seed allocated to three pieces of land) and the problem can be stated in three different scenarios (with dry, normal or raining weather). In addition, since the purchase prices are rather expensive, the goal of the farmer is to obtain the maximum benefit but avoiding solutions with very low production in some scenarios (although, in others these solutions may provide the best production), i.e., the farmer rejects risks (see Section 2.3 and 4 for an illustration of the formulation of this kind of problems).

In this paper we present an exact algorithm for the (one stage) stochastic Set Packing problem, SSPP, where a mean–risk function is considered as the objective function, in particular, mean$-\beta$ risk for some positive risk measure and some $\beta > 0$. Some other approaches, say, type min-max or max-min to reduce the probability of the occurrence of non-desired scenarios, such as the minimization of the greatest regret robust solution value (i.e, the greatest difference between the optimal solution value of the scenarios and the provided solution value over the scenarios) or the maximization of the solution value for the worst scenario, require specific equations across the scenario constraint systems. Instead of those approaches, we only require an independent new constraint for each scenario, allowing an easier decomposition of the problem.

There are not many approaches for solving large-scale stochastic combinatorial problems via scenario analysis. See in [21] the presentation of an exact algorithm for medium sized multistage mixed 0-1 problems. See also in [31] a heuristic algorithm for SIP solving, where a subset of scenarios is updated at each iteration of the algorithm. See in [14] approximation algorithms polynomially solvable for stochastic job problems, where the uncertainty lies in the processing of each job and the goal is to compute a policy whose expected value is provably close to that of an optimal adaptive policy. [16] studies stochastic variants of the 0-1 dimensional knapsack problem to pack a maximum-value collection of items with stochastic vector-valued sizes; see there additional applications for the maximum clique, stable set, matching, b-matching and others. [5] presents a heuristic procedure for stochastic integer problems under probability constraints and [15, 33] make use of the scenario analysis for dealing with the stochastic knapsack problem or the stochastic machine replacement problem. None of these approaches explicitly consider the Set Packing problem under uncertainty in the coefficients of the objective function.

The main ingredients of the algorithm for SSPP that we present here are: (a) designing a one-stage DEM for the mean–risk SSPP; (b) exploiting the special structure of the DEM of the mean–risk stochastic integer program for fixing variables in the preprocessing; (c) presenting the *splitting variable* representation of the DEM; (d) obtaining good lower bounds of the optimal solution for mean–risk SSPP; and (e) obtaining tight upper bounds on the optimal solution, by using the Volume Algorithm [3] to update

the multipliers in the Lagrangian decomposition of the SSPP.

The remainder of the paper is organized as follows. Section 2 deals with some properties of the SSPP model and three procedures for obtaining feasible solutions (i.e., lower bounds for the optimal solution value). At the end of the section, we give an illustration of the kind of problems for which the method of the paper provide solutions. Section 3 presents the Lagrangian decomposition. Section 4 reports the computational experience that we have obtained. Section 5 concludes.

2 The model and its properties

Given a set of scenarios Ω, we will consider two types of Set Packing problems. The first one, the deterministic model, looks for an optimal solution at each scenario $\omega \in \Omega$:

$$\begin{aligned} (\text{SPP}_\omega) \quad \max \quad & \sum_{j \in J} c_j^\omega x_j \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j \leq 1, \quad \forall i \in I \\ & x_j \in \{0,1\}, \quad \forall j \in J, \end{aligned}$$

where $c^\omega = (c_1^\omega, \ldots, c_m^\omega) \in \mathbb{R}_+^{\mathbf{m}}$ is the vector of benefits associated with the scenario ω, $A = (a_{ij})_{i \in I, j \in J}$ is a 0-1 matrix and x is the scenario based vector of variables. In the example of the introduction, c^ω vector represents the mean yield for the different weather conditions (alternatively, it might be the benefits of bids in auctions problems) and the constraints represents the assignment problem that ensure that each type of corn seed should be allocated to just one piece of land.

In the second one, the stochastic model, it is the weighted sum of the objective functions over all scenarios that is maximized.

$$\begin{aligned} (\text{SPP}_\sum) \quad \max \quad & \sum_{\omega \in \Omega} \sum_{j \in J} w^\omega c_j^\omega x_j \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j \leq 1, \quad \forall i \in I \\ & x_j \in \{0,1\}, \quad \forall j \in J, \end{aligned}$$

where w^ω gives the weight that the modeler associates to the scenario ω.

In this paper, we will study the particular stochastic version that consists of the maximization of the objective function expected value minus the weighted risk of having a scenario with an objective function value smaller than a given threshold, by the inclusion of an additional reject risk constraint for any $\omega \in \Omega$:

$$\begin{aligned}
\text{(SSPP)} \quad \max \quad & \sum_{\omega\in\Omega}\sum_{j\in J} w^\omega c_j^\omega x_j - \beta \sum_{\omega\in\Omega} w^\omega \delta^\omega \\
\text{s.t.} \quad & \sum_{j\in J} a_{ij} x_j \leq 1, \quad \forall i \in I \\
& \sum_{j\in J} c_j^\omega x_j \geq \phi - \phi\delta^\omega, \quad \forall \omega \in \Omega \\
& x_j, \delta^\omega \in \{0,1\}, \quad \forall j \in J, \omega \in \Omega,
\end{aligned}$$

where β is the penalty associated with the scenarios whose objective value is smaller than a given value ϕ. Note that δ^ω is defined so that it takes the value 1 if $c^\omega \cdot x < \phi$ and the value 0 otherwise, see [45]. In other words, the definition of δ is just what is required to find a feasible solution of SSPP from a feasible solution of $\text{SPP}_\sum$. This model represents real situations where a penalty is applied if the objective value of a scenario is not greater than or equal to a given threshold. In the example of the farmer in the introduction, this threshold can be given by the requirement to cattle feed.

In the following, we shall denote by $v(\text{P})$ to the optimum value of problem P. Also, we shall assume that if $\nu_1, \ldots, \nu_k$ are a well-defined set of variables, then ν is the vector $(\nu_1, \ldots, \nu_k)$.

2.1 Relationship between $v(\text{SPP}_\omega)$, ϕ and β

It is worth while mentioning that depending on the relationship between the values $v(\text{SPP}_\omega)$, ϕ and β, the process of obtaining an optimal solution of SSPP could have very different levels of difficulty.

The case where β is small enough, the optimal solution of $(\text{SPP}_\sum)$, in general, provides a good lower bound of the problem. Indeed, since the penalty of not satisfying the risk constraints is quite small, the optimal solution of $(\text{SPP}_\sum)$ could be quite close to the optimal solution of SSPP. In the case where β is large enough, a good lower bound can be provided with those solutions such that the number of scenarios in which

$v(\text{SPP}_\omega) \geq \phi$ is as large as possible.

Additionally, we are able to fix $\delta^\omega = 1$ for any $\omega \in \Omega$ such that $v(\text{SPP}_\omega) < \phi$, therefore, if ϕ is big enough we can fix δ^ω for many $\omega \in \Omega$ and the resulting problem can be solved in a easier way. In other words, difficult problems are those with $\phi < \min_{\omega\in\Omega} v(\text{SPP}_\omega)$.

2.2 Lower bounds of v(SSPP)

SSPP is often too difficult to be solved directly. A lower bound for the value of the objective function in an optimal solution of SSPP allow to measure the gap when an upper bound for the same value is known and thus the quality of an interval containing the optimal value. If this lower bound comes from a feasible solution, it also helps to reduce computation time of any algorithm which needs a feasible seed for starting. Since one of the goals of this paper is to perform a Lagrangian algorithm for solving the SSPP and any Lagrangian algorithm gives upper bounds of the optimal value, then we would like to have good lower bounds of v(SSPP).

In this section three different procedures for obtaining lower bounds of v(SSPP) are described. In Section 4 the efficiency of these lower bounds is illustrated.

Procedure I. Solving the linear relaxation of SSPP provides a lower bound of v(SSPP) in a very easy way. In this procedure, we analyze a stronger formulation of this linear relaxation, by solving an integer linear programming, that obviously allows us to obtain better lower bounds. In particular, the procedure can be described through the following steps.

Step 1 For each $\omega \in \Omega$ solve

$$\begin{aligned} \min \quad & \sum_{j\in J} c_j^\omega x_j \\ \text{s.t.} \quad & \sum_{j\in J} a_{ij} x_j = 1, \quad \forall i \in I' \\ & x_j \in \{0,1\}, \quad \forall j \in J. \end{aligned}$$

where I' is a subset of I for which the above problem has a feasible solution. Let $c^\omega_{\min}$ be the optimal value for each $\omega \in \Omega$.

Step 2 In SSPP replace

$$\sum_{j\in J} c_j^{\omega} x_j \geq \phi - \phi\delta^{\omega}, \quad \forall \omega \in \Omega,$$

with the new ones

$$\sum_{j\in J} c_j^{\omega} x_j \geq \phi - \delta^{\omega}(\phi - c_{\min}^{\omega}), \quad \forall \omega \in \Omega,$$

and solve the corresponding linear relaxation.

One way of computing I' is to start with $I' = I$: if the problem in Step 1 has a feasible solution, stop; else set $I' = I' \setminus \{i\}$ for an $i \in I'$. We repeat until the problem in Step 1 has a feasible solution. At the end, $I' \neq \emptyset$ because $a_{ij} \in \{0, 1\}$.

Note that the solution of the LP relaxation of the revised formulation is a lower bound of v(SSPP). Moreover, the efficiency of this procedure is due to the fact that the lower bounds can be computed in a very easy way because only an integer linear programming, which is an assignment problem, is needed to be solved. In addition, the resulting stronger linear relaxation of SSPP can be solved very fast by any commercial solver.

Procedure II. This procedure provides a lower bound by finding a feasible solution of SSPP from an optimal solution of $\text{SPP}_{\sum}$. In particular, the procedure can be described through the following steps.

Step 1 Solve $\text{SPP}_{\sum}$. Let x be an optimal solution of $\text{SPP}_{\sum}$.

Step 2 Set

$$\delta^{\omega} = \begin{cases} 1, & \text{if } \sum_{j\in J} c_j^{\omega} x_j < \phi \\ 0, & \text{otherwise,} \end{cases}$$

for all $\omega \in \Omega$.

Step 3 Compute $\sum_{w\in\Omega} \sum_{j\in J} w^{\omega} c_j^{\omega} x - \beta \sum_{\omega\in\Omega} w^{\omega}\delta^{\omega}$.

Note that (x, δ^{ω}) is a feasible solution of SSPP, then its objective value for the SSPP provides a lower bound for v(SSPP). This procedure is useful for updating lower bounds in Lagrangian iterative algorithms (see Section 3.1).

Procedure III. This procedure is special suitable for large values of β. The idea behind this approach is to obtain a feasible solution of SSPP with as many scenarios as possible providing values greater than ϕ.

Given a feasible solution of $\text{SPP}_{\sum}$, we choose those scenarios, $\Omega' \subseteq \Omega$, where $v(\text{SPP}_\omega)$ is a promising value for any $\omega \in \Omega'$. We have chosen $v(\text{SPP}_\omega) > 0.9\phi$ for each $\omega \in \Omega'$ as our criterium for promising.

Then, a good lower bound could follow from the following scalarized problem:

$$\begin{aligned} (\text{MOSPP}_\rho) \quad \max \quad & \sum_{\omega \in \Omega'} \rho^\omega \sum_{j \in J} w^\omega c_j^\omega x_j \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j \leq 1, \quad \forall i \in I \\ & x_j \in \{0,1\}, \quad \forall j \in J, \end{aligned}$$

for certain $\rho^\omega > 0$ with $\omega \in \Omega'$. Hence, Procedure III is defined by the following iterative scheme.

Input I, J, β, ϕ, Ω, $\text{LBitera}_{\max}$, w^ω, c^ω for any $\omega \in \Omega$.

Step 1 Initialize LB$= -\infty$, $\rho^\omega = 1$ for any $\omega \in \Omega$, LBiterations$= 0$.

Step 2 Solve $\text{SPP}_{\sum}$ and let x be an optimal solution. Set $\Omega' = \{\omega \in \Omega : c^\omega x > 0.9\phi\}$ and set $\rho^\omega = 0$ for any $\omega \in \Omega \setminus \Omega'$.

Step 3 Consider $A, B, C \subseteq \Omega'$, such that, $A = \{\omega \in \Omega' : 0.9\phi < c^\omega x \leq \phi\}$, $B = \{\omega \in \Omega' : \phi < c^\omega x \leq 1.1\phi\}$, $C = \{\omega \in \Omega' : c^\omega x > 1.1\phi\}$.

Step 4 Set $\rho^\omega = 1.7\rho^\omega$ for $\omega \in A$, $\rho^\omega = 0.9\rho^\omega$ for $\omega \in B$, and $\rho^\omega = 0.8\rho^\omega$ for $\omega \in C$.

Step 5 Solve MOSPP_ρ and let x be an optimal solution. Define $\delta^\omega = 1$ if $c^\omega x < \phi$ and 0 otherwise $\forall \omega \in \Omega$.

Step 6 If $\sum_{\omega \in \Omega} w^\omega c^\omega x - \beta \sum_{\omega \in \Omega} w^\omega \delta^\omega >$ LB, then LB $= \sum_{\omega \in \Omega} w^\omega c^\omega x - \beta \sum_{\omega \in \Omega} w^\omega \delta^\omega$.

Step 7 Set LBiterations $=$ LBiterations $+ 1$. If LBiterations $<$ LBitera_{max}, then go to Step 3. Else, stop.

This procedure does not consider the scenarios with objective value much smaller than ϕ, $\Omega \setminus \Omega'$, and it allocates: 1) big weights to the scenario $\omega \in \Omega'$, such

that, $v(\text{SPP}_\omega) < \phi$ but close enough to ϕ; and 2) intermediate weights to those scenarios, $\omega \in \Omega'$, such that, $v(\text{SPP}_\omega) \geq \phi$. As we mentioned, the idea behind this approach is to obtain a feasible solution of SSPP with as many scenarios as possible providing values greater than ϕ. Then, in such cases, the penalty β is avoided and hopefully a solution built under these conditions will provide a "good" objective value. In summary, this procedure gives the most importance to those scenarios that are close to having an objective value greater than or equal to ϕ, for this reason we assign to these scenarios a higher weight in the next iteration, in particular the value $1.7\rho^\omega$ have shown to be special suitable in our computational experience. Moreover, those scenarios that have already reached this value are given intermediate weights in the hope that in the next iteration they will keep its value over ϕ; in this case, for the next iteration we assign to scenarios of sets B and C the weights $0.9\rho^\omega$ and $0.8\rho^\omega$, respectively. Finally, in order to reach the goal, we sacrifice those scenarios which, in principle, are less likely to reach an objective value greater than ϕ, so we assign a null weight to the scenarios with optimal objective value lower than or equal to ϕ.

2.3 Relationship between the optimal solution to SSPP and related problems

The relationship presented in this section indicates the relevance of the integrality constraints of δ^ω-variables for all $\omega \in \Omega$. Indeed, the linear relaxation of the problem in this section has an optimal solution with $x_j \in \{0, 1\}\ \forall j \in J$, and $\delta^\omega \notin \{0, 1\}$ for some $\omega \in \Omega$. In principle, one could conjecture that if $\sum_{\omega\in\Omega} \delta^\omega = 0$ for an optimal solution of the SSPP then this value is also 0 for an optimal solution of its corresponding linear relaxation. However, in order to prove that this is not the case, consider the following example where the set packing constraints are given by those of an assignment problem, each scenario has the same associated probability, $|I| = |J| = 2$, $|\Omega| = 11$, $\beta = 0.9$, $\phi = 10$, i.e.

$$\max \quad \frac{1}{|\Omega|} \sum_{\omega\in\Omega} \sum_{i\in I} \sum_{j\in J} w^\omega c_{ij}^\omega x_{ij} - \frac{\beta}{|\Omega|} \sum_{\omega\in\Omega} \delta^\omega$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1, \quad \forall j \in J \\
& \sum_{j \in J} x_{ij} \leq 1, \quad \forall i \in I \\
& \sum_{i \in I} \sum_{j \in J} c_{ij}^{\omega} x_{ij} \geq \phi - \phi \delta^{\omega}, \quad \forall \omega \in \Omega \\
& x_{ij}, \delta^{\omega} \in \{0,1\}, \quad \forall i \in I, j \in J, \omega \in \Omega
\end{aligned}
$$

with

$$
c^1 = \begin{pmatrix} 11 & 5 \\ 5 & 11 \end{pmatrix} \quad c^{\omega} = \begin{pmatrix} 5 & 5 \\ 5 & 4 \end{pmatrix} \text{ for } w \in \{2, \ldots, 11\}.
$$

In this case, the optimal solution for the relaxed problem is $x_{11} = x_{22} = 1$, $x_{12} = x_{21} = 0$, $\delta^1 = 0$, $\delta^{\omega} = 0.1$ for any $\omega \in \{2, \ldots, |\Omega|\}$ with objective function value 10.1. The optimal solution of SSPP is $x_{11} = x_{22} = 0$, $x_{12} = x_{21} = 1$ and $\delta^{\omega} = 0$ for each $\omega \in \Omega$, obtaining an objective value of 10. However, the sum for the optimum value of variables δ^{ω} for $\omega \in \Omega$ for the corresponding LP relaxed optimal solution takes the value 1.

Additionally, it is worth while mentioning that there are some particular situations where we can obtain an optimal solution of SSPP in a easy way by solving in $\text{SPP}_{\sum}$. Indeed, whenever an optimal solution of $\text{SPP}_{\sum}$ satisfies that $\sum_{j \in J} c_j^{\omega} u_j \geq \phi$ for any $\omega \in \Omega$, that solution is also optimal for SSPP with $\delta^{\omega} = 0$ for any $\omega \in \Omega$.

3 Lagrangian decomposition

A commercial IP solver can be used to solve SSPP. But, in general, such an approach will yield excessive running times, even for moderately sized problems (see Section 4). This motivates the development of an alternative to solving SSPP directly that exploits the intrinsic framework of the model. Between the different alternatives that we have studied to solve SSPP, the most efficient has been based on the Lagrangian decomposition algorithm that we describe in the following.

With the aim of splitting the SSPP into several subproblems, we duplicate several variables. Setting $z_j = x_j$, $s_j^1 = x_j$, for all $j \in J$, $s_j^{\omega} = s_j^{\omega+1}$ $\forall j \in J$, $\omega \in \{1, \ldots, |\Omega| - 1\}$,

and writing the model in terms of the variables δ^ω, z and s^ω SSPP becomes:

$$\begin{aligned}
\max \quad & \sum_{\omega\in\Omega}\sum_{j\in J} w^\omega c_j^\omega z_j - \beta \sum_{\omega\in\Omega} w^\omega \delta^\omega \\
\text{s.t.} \quad & \sum_{j\in J} a_{ij} z_j \leq 1, \quad \forall i \in I && (1)\\
& z_j = s_j^1, \quad \forall j \in J && (2)\\
& s_j^\omega = s_j^{\omega+1}, \quad \forall j \in J, \omega \in \{1,\ldots,|\Omega|-1\} && (3)\\
& \sum_{j\in J} c_j^\omega s_j^\omega \geq \phi - \phi\delta^\omega, \quad \forall \omega \in \Omega && (4)\\
& z_j, s_j^\omega, \delta^\omega \in \{0,1\}, \quad \forall j \in J, \omega \in \Omega. && (5)
\end{aligned}$$

Dualizing the constraints (2) and (3) with the Lagrange multipliers vectors α and λ^ω, respectively, yields the Lagrangian Decomposition problem LD:

$$\begin{aligned}
(\text{LD}) \quad \max \quad & \sum_{\omega\in\Omega}\sum_{j\in J} w^\omega c_j^\omega z_j - \beta \sum_{\omega\in\Omega} w^\omega \delta^\omega + \sum_{j\in J} \alpha_j (z_j - s_j^1) + \sum_{j\in J}\sum_{\omega=1}^{|\Omega|-1} \lambda_j^\omega (s_j^\omega - s_j^{\omega+1}) \\
\text{s.t.} \quad & (1), (4), (5).
\end{aligned}$$

Therefore, the objective value of LD is the result of the addition of the Lagrangian subproblems P_0, $\text{P}_1, \ldots, \text{P}_{|\Omega|}$,

$$v(\text{LD}) = v(\text{P}_0) + \sum_{\omega\in\Omega} v(\text{P}_\omega),$$

where

$$\begin{aligned}
(\text{P}_0) \quad \max \quad & \sum_{j\in J} \left(\sum_{\omega\in\Omega} w^\omega c_j^\omega + \alpha_j \right) z_j \\
\text{s.t.} \quad & \sum_{j\in J} a_{ij} z_j \leq 1, \quad \forall i \in I \\
& z_j \in \{0,1\}, \quad \forall j \in J,
\end{aligned}$$

$$\begin{aligned}
(\text{P}_1) \quad \max \quad & \sum_{j\in J} (\lambda_j^1 - \alpha_j) s_j^1 - w^1 \beta \delta^1 \\
\text{s.t.} \quad & \sum_{j\in J} c_j^1 s_j^1 \geq \phi - \phi\delta^1, \\
& \delta^1, s_j^1 \in \{0,1\}, \quad \forall j \in J,
\end{aligned}$$

$$
\begin{aligned}
(\mathrm{P}_\omega) \quad \max \quad & \sum_{j\in J}(\lambda_j^\omega - \lambda_j^{\omega-1})s_j^\omega - w^\omega\beta\delta^\omega \\
\text{s.t.} \quad & \sum_{j\in J} c_j^\omega s_j^\omega \geq \phi - \phi\delta^\omega, \\
& \delta^\omega, s_j^\omega \in \{0,1\}, \quad \forall j \in J,
\end{aligned}
$$

for all $\omega \in \{2,\ldots,|\Omega|-1\}$ and

$$
\begin{aligned}
(\mathrm{P}_{|\Omega|}) \quad \max \quad & \sum_{j\in J}(-\lambda_j^{|\Omega|-1})s_j^{|\Omega|} - w^{|\Omega|}\beta\delta^{|\Omega|} \\
\text{s.t.} \quad & \sum_{j\in J} c_j^{|\Omega|} s_j^{|\Omega|} \geq \phi - \phi\delta^{|\Omega|}, \\
& \delta^{|\Omega|}, s_j^{|\Omega|} \in \{0,1\}, \quad \forall j \in J.
\end{aligned}
$$

Moreover, setting $\bar{s}_j^\omega = 1 - s_j^\omega \ \forall j \in J, \omega \in \Omega$ and $\bar{\delta}^\omega = 1 - \delta^\omega \ \forall \omega \in \Omega$, we can rewrite $\mathrm{P}_1,\ldots,\mathrm{P}_{|\Omega|}$ as the following 0-1 knapsack problems $\overline{\mathrm{P}}_1,\ldots,\overline{\mathrm{P}}_{|\Omega|}$:

$$
\begin{aligned}
(\overline{\mathrm{P}}_1) \quad \max \quad & \sum_{j\in J}(\alpha_j - \lambda_j^1)\bar{s}_j^1 + w^1\beta\bar{\delta}^1 + \sum_{j\in J}(\lambda_j^1 - \alpha_j) - w^1\beta \\
\text{s.t.} \quad & \sum_{j\in J} c_j^1 \bar{s}_j^1 + \phi\bar{\delta}^1 \leq \sum_{j\in J} c_j^1, \\
& \bar{\delta}^1, \bar{s}_j^1 \in \{0,1\}, \quad \forall j \in J,
\end{aligned}
$$

$$
\begin{aligned}
(\overline{\mathrm{P}}_\omega) \quad \max \quad & \sum_{j\in J}(\lambda_j^{\omega-1} - \lambda_j^\omega)\bar{s}_j^\omega + w^\omega\beta\bar{\delta}^\omega + \sum_{j\in J}(\lambda_j^\omega - \lambda_j^{\omega-1}) - w^\omega\beta \\
\text{s.t.} \quad & \sum_{j\in J} c_j^\omega \bar{s}_j^\omega + \phi\bar{\delta}^\omega \leq \sum_{j\in J} c_j^\omega, \\
& \bar{\delta}^\omega, \bar{s}_j^\omega \in \{0,1\}, \quad \forall j \in J,
\end{aligned}
$$

for all $\omega \in \{2,\ldots,|\Omega|-1\}$ and

$$
\begin{aligned}
(\overline{\mathrm{P}}_{|\Omega|}) \quad \max \quad & \sum_{j\in J}\lambda_j^{|\Omega|-1}\bar{s}_j^{|\Omega|} + w^{|\Omega|}\beta\bar{\delta}^{|\Omega|} - \sum_{j\in J}\lambda_j^{|\Omega|-1} - w^{|\Omega|}\beta \\
\text{s.t.} \quad & \sum_{j\in J} c_j^{|\Omega|} \bar{s}_j^{|\Omega|} + \phi\bar{\delta}^{|\Omega|} \leq \sum_{j\in J} c_j^{|\Omega|}, \\
& \bar{\delta}^{|\Omega|}, \bar{s}_j^{|\Omega|} \in \{0,1\}, \quad \forall j \in J.
\end{aligned}
$$

To be precise, $v(\mathrm{P}_\omega) = v(\overline{\mathrm{P}}_\omega)$ for all $\omega \in \{1,\ldots,|\Omega|\}$. The interest of these new subproblems lies in the possibility of optimally solving them without calling any IP solver; given the good results reported in [32] for solving knapsack problems, we propose the algorithm introduced there for solving our 0-1 knapsack problems.

3.1 Lower bound (LB) updating

Subsection 2.2 explains how a feasible solution for SSPP can be built from a feasible solution of $\text{SPP}_{\sum}$ or equivalently, for SPP_{ω} with $\omega \in \Omega$ (see Procedure II). Each Lagrangian iteration solves P_0, which is a set packing problem. Then, our algorithm will check at any iteration whether the feasible solution for SSPP obtained from the optimal solution of P_0 gives an optimal value greater than the initial LB obtained by the heuristic procedure described also in the Procedure III of Subsection 2.2; in this case, we will update the LB in the algorithm.

3.2 Fixing variables

Making use of the structure of our problem, we can develop two methods for fixing variables, the first one for the s-variables and the second for the δ-variables. Fixing variables have two benefits: it reduces the size of the problem and it helps to increase the lower bounds of $v(\text{SSPP})$ because the optimal value of its LP relaxation also increases. Since one of the goals of this work is to illustrate that we can obtain narrow intervals containing the optimal solution for a class of difficult stochastic programs in a reasonable time, the size of the problem and the quality of the lower bound strongly influences on the time.

The updating of the Lagrangian multipliers is an iterative method. We shall assume throughout this section that we are at iteration t of the Lagrangian multipliers updating procedure.

3.2.1 Fixing s-variables

For a better and intuitive understanding, we will consider the problems $\text{P}_0, \text{P}_1, \ldots, \text{P}_{|\Omega|}$ to describe this fixing variables procedure instead of their corresponding transformations, $\overline{\text{P}}_0, \overline{\text{P}}_1, \ldots, \overline{\text{P}}_{|\Omega|}$. Note that fixing the value of s_j^{ω} is equivalent to fixing the value of $\bar{s}_j^{\omega}$.

Let $v_t(\text{LD})$ and $v_t(\text{P}_0)$ be the optimal values of problems LD and P_0, respectively, at iteration t, and let z^t, $(s^{\omega,t}, \delta^{\omega,t})$ be the optimal solutions of the problems P_0, P_{ω} for $\omega \in \Omega$, respectively, at that iteration t. Moreover, let α_j^t, $\lambda_j^{\omega,t}$ be the Lagrange

multipliers at iteration t for $j \in J$ and $\omega \in \{1, \ldots, |\Omega|\}$. Using this notation we can obtain the following result:

Proposition 3.1 *Let $z_k^t = s_k^{1,t} = \ldots = s_k^{|\Omega|,t} = 0$ for some $k \in J$ at iteration t. Then,*

$$v_t(\mathrm{LD}) - \alpha_k^t - v_t(\mathrm{P}_0) + v(\mathrm{P}_=) \tag{6}$$

is an upper bound for SSPP with the additional constraint $x_k = 1$, where

$$\begin{aligned}
(\mathrm{P}_=) \quad \max \quad & \sum_{j \in J} \left(\sum_{\omega \in \Omega} w^\omega c_j^\omega + \alpha_j^t \right) z_j \\
\text{s.t.} \quad & \sum_{j \in J} a_{ij} z_j \leq 1 \quad \forall i \in I \\
& z_k = 1 \\
& z_j \in \{0, 1\} \quad \forall j \in J.
\end{aligned}$$

Proof:

Let $\hat{z}^t$ and $\hat{s}^{\omega,t}$ be defined by $\hat{z}_k^t = \hat{s}_k^{\omega,t} = 1$, $\hat{z}_j^t = z_j^t$ and $\hat{s}_j^{\omega,t} = s_j^{\omega,t}$ for any $j \in J \setminus \{k\}$. Since $(s_j^{\omega,t}, \delta^{\omega,t})$ is an optimal solution for P_ω with $\omega \in \Omega$, and then feasible, we have that $\sum_{j \in J} c_j^\omega s_j^\omega \geq \phi - \phi\delta^\omega$, for each $\omega \in \Omega$. Hence, since $\sum_{j \in J} c_j^\omega \hat{s}_j^\omega \geq \sum_{j \in J} c_j^\omega s_j^\omega$, $(\hat{s}_j^{\omega,t}, \delta^{\omega,t})$ is feasible for P_ω, i.e.,

$$\sum_{j \in J} c_j^\omega \hat{s}_j^\omega \geq \phi - \phi\delta^\omega, \qquad \forall \omega \in \Omega.$$

Thus, the sum of differences between the objective function of P_ω at the solutions $(\hat{s}_j^{\omega,t}, \delta^{\omega,t})$ and $(s_j^{\omega,t}, \delta^{\omega,t})$ for $\omega \in \Omega$ is given by

$$\lambda_k^{1,t} - \alpha_k^t - \lambda_k^{|\Omega|-1,t} + \sum_{\omega=2}^{|\Omega|-1} (\lambda_k^{\omega,t} - \lambda_k^{\omega-1,t}) = -\alpha_k^t.$$

Finally, we solve $\mathrm{P}_=$ because $\hat{z}$ is not always a feasible solution of $\mathrm{P}_=$. □

Thus, if value (6) is smaller than LB (Subsection 2.2), we can fix a set of variables to zero, i.e., we can add $z_j = s_j^1 = \ldots = s_j^{|\Omega|} = 0$ for all the subsequent problems.

3.2.2 Fixing δ-variables

If $\delta^{\omega,t} = 0$ for a certain $\omega \in \Omega$ at iteration t, then the value

$$v_t(\mathrm{LD}) - w^\omega \beta$$

is an upper bound for SSPP, where the additional constraint $\delta^\omega = 1$ is imposed. Consequently, if it is smaller than LB (Subsection 2.2), we can fix the variable to zero, i.e., $\delta^\omega = 0$.

See also the remark in Subsection 2.1 for fixing $\delta^\omega = 1$.

3.3 Lagrange multipliers updating

In this section we describe the methodology that we have followed for updating the Lagrange multipliers. In particular, we have considered the so-called Volume Algorithm proposed in [3], which is a modification of the traditional subgradient algorithm introduced in [25], see also [6], among others. The algorithm is designed to avoid the oscillatory behaviour of the traditional subgradient method for updating the Lagrange multipliers. Unlike the classical subgradient procedures, the Volume Algorithm only updates the multipliers when they correspond with an improvement of the incumbent solution of the Lagrangian problem (notice that the multipliers are updated at each iteration in the subgradient method), and the feasible solution is replaced by a convex combination of solutions obtained in previous iterations, instead of only considering the last one. For overviews on multiplier updating schemes, see [17, 24, 27, 42], among others.

Thus, at each iteration we compute

$$\alpha_j^t = \hat{\alpha}_j - \gamma^t(\tilde{z}_j^{t-1} - \tilde{s}_j^{1,t-1})$$

$$\lambda_j^{\omega,t} = \hat{\lambda}_j^{\omega} - \gamma^t(\tilde{s}_j^{\omega,t-1} - \tilde{s}_j^{\omega+1,t-1})$$

where $\hat{\alpha}_j$ and $\hat{\lambda}_j^{\omega}$ are the Lagrange multipliers from the last iteration in which the incumbent solution was improved; $\tilde{z}_j^t$ and $\tilde{s}_j^{\omega,t}$ are given by the following convex combination

$$\tilde{z}^t = \tau z^t + (1-\tau)\tilde{z}^{t-1}$$

$$\tilde{s}^{\omega,t} = \tau s^{\omega,t} + (1-\tau)\tilde{s}^{\omega,t-1}$$

with $0 < \tau \leq 1$; z^t is an optimal solution of problem P_0 at iteration t; $(s^{\omega,t}, \delta^{\omega,t})$ is an optimal solution of problem P_ω at iteration t for $\omega \in \Omega$, obtained from an optimal

solution $(\bar{s}^{\omega,t}, \bar{\delta}^{\omega,t})$ of $(\overline{\mathrm{P}}_\omega)$ by doing $(s^{\omega,t}, \delta^{\omega,t}) = (1 - \bar{s}^{\omega,t}, 1 - \bar{\delta}^{\omega,t})$; finally,

$$\gamma^t = \frac{\pi^t(v_t(\mathrm{LD}) - \mathrm{LB})}{\sum_{j\in J}(\tilde{z}_j^t - \tilde{s}_j^{1,t})^2 + \sum_{j\in J}\sum_{\omega=1}^{|\Omega|-1}(\tilde{s}_j^{\omega,t} - \tilde{s}_j^{\omega+1,t})^2}$$

with $0 < \pi^t \leq 2$.

Since the values of π^t and τ can be set to fixed values for a number of iterations and could change (usually decrease) afterwards, we follow the advice in [3] for re-computing π and τ. (In any case, it can be shown that $v_t(\mathrm{LD}) \to v(\mathrm{LD})$ if $\gamma^t \to 0$ and $\sum_{k=0}^{\infty}\pi^t \to \infty$, whenever $t \to +\infty$, see [3, 39]). Let $v_{1,j}^t = z_j^t - s_j^{1,t}$, $v_{2,j}^{\omega,t} = s_j^{\omega,t} - s_j^{\omega+1,t}$, $\tilde{v}_{1,j}^t = \tilde{z}_j^t - \tilde{s}_j^{1,t}$, $\tilde{v}_{2,j}^{\omega,t} = \tilde{s}_j^{\omega,t} - \tilde{s}_j^{\omega+1,t}$ for $j \in J$ and $\omega = 1, \ldots, |\Omega| - 1$.

- τ updating. At each iteration we compute τ_{opt} as the value that minimizes

  $$\|\tau(v_1^t, v_2^t) + (1 - \tau)(\tilde{v}_1^t, \tilde{v}_2^t)\|.$$

  If this τ_{opt} is negative, equivalently $(\tilde{v}_1^t, \tilde{v}_2^t)((v_1^t, v_2^t) - (\tilde{v}_1^t, \tilde{v}_2^t))$ is positive, we would set $\tau = \tau/10$. Otherwise, we would set $\tau = \min\{\tau, \tau_{\mathrm{opt}}\}$.

- π updating. The iterations are classified into three types: *red*, *green* and *yellow* iterations (the notation proposed in [3] is followed). *Red* iterations are those Lagrangian iterations in which the incumbent solution is not improved. If the incumbent solution is improved we compute

  $$(v_1^t, v_2^t) \cdot (\tilde{v}_1^t, \tilde{v}_2^t).$$

  If it is negative it means that a longer step in the direction (v_1^t, v_2^t) would have given a larger value for $v^t(\mathrm{LD})$, this iteration is called *yellow*. If it is non negative, it is called *green*. At each *green* iteration, we would set $\pi = \min\{2, 1.1\pi\}$. After a sequence of 50 consecutive *red* iterations we would decrease the value, i.e., $\pi = 0.66\pi$.

The initial values for τ and π in the computational experiment whose results are reported in Section 4 are 0.1 and 1.5, respectively.

3.4 Initial and final multipliers

The performance of our algorithm, like many Lagrangian relaxation algorithms, is somewhat sensitive to the choice of the initial Lagrange multipliers. We have observed that values of the Lagrange multipliers for the instances of our computational study varies depending on the value of β and the difference $\phi - \beta$. We can find suggestions in the literature for taking the first value of the Lagrange multipliers as the dual variables associated with the nonanticipativity constraints (2) and (3) in the LP relaxation of model SSPP. However, since the results in the computational section were satisfactory enough we preferred not to add a formula for initializing multipliers, which evidently would depend on our data set instances. Instead we initialize $\alpha = 0$ and $\lambda = 0$.

When we start with $\alpha = 0$, $\lambda = 0$, it results that $v(\mathrm{LD}) = v(\mathrm{P}_0)$ since $v(\mathrm{P}_1) = \ldots = v(\mathrm{P}_{|\Omega|}) = 0$, what is obviously a bad upper bound for the cases in which a non-empty set of δ-variables take value 1 at the optimal solution of SSPP. We have empirically observed that the Lagrange multipliers which give the best incumbent solution satisfy $\alpha_j > \lambda_j^1 > \ldots > \lambda_j^{|\Omega|}$ for all $j \in J$. However, initializing α and λ in such a way does not guarantee that the initial upper bound is better than the upper bound obtained with $\alpha = \lambda = 0$ except if the increase in $v(\mathrm{P}_0)$ is smaller than the summation of decrease in $v(\mathrm{P}_1) = \ldots = v(\mathrm{P}_{|\Omega|})$.

3.5 Algorithm

Taking into account all the previous results, we describe the following algorithm to solve Problem SSPP. It is a Lagrangian algorithm in which the lower bounds are computed following the advices in Subsection 2.2 and the lagrangian multipliers are updated with the Volume Algorithm methodology (Subsection 3.3).

ALGORITHM 3.1

Step 1 *Initialize* α, λ, *UB (Upper Bound),* π, τ, *and* iterations *(maximum number of iterations).*

Step 2 *Check if there is an "easy optimal solution", see Subsections 2.1 and 2.3. If there are any, then stop.*

Step 3 *Compute a lower bound (LB) with* Procedure III *presented in Subsection 2.2.*

Step 4 *Perform the Lagrangian algorithm with the properties presented in Subsection 3.3. At each iteration,* P_0 *is optimally solved calling an IP optimization engine, and* $\overline{\mathrm{P}}_1, \ldots \overline{\mathrm{P}}_{|\Omega|}$ *are optimally solved through the exact algorithm COMBO for 0-1 knapsack problems proposed in [32]. The algorithm stops when the optimum is found or when the number of iterations exceeds* iterations.

In particular, we have set $\alpha = 0$, $\lambda = 0$, UB=10^7, $\pi = 1.5$, $\tau = 0.1$ and iterations= 1000. In Step 3 we set $\text{LBitera}_{\max} = 500$. Values 0 for α and λ are justified in Section 3.4. UB is a large constant with respect to the c_j^ω we are going to use in Section 4 (see there that $c_j^\omega \in \{1, \ldots, 100\}$.) Values for π and τ are recommended in [3] for the volume algorithm. For "iterations" and "$\text{LBitera}_{\max}$" we checked several possibilities and one for which we obtained satisfactory results was iterations= 1000 and $\text{LBitera}_{\max} = 500$.

4 Computational study

We tested our algorithm over 90 instances. In all cases the Set Packing problem is the assignment problem and the likelihood of each scenario is $w^\omega = 1/|\Omega|$. Thus we solve

$$
\begin{aligned}
\max \quad & \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{i \in I} \sum_{j \in J} c_{ij}^\omega x_{ij} - \frac{\beta}{|\Omega|} \sum_{\omega \in \Omega} \delta^\omega \\
\text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1, \quad \forall j \in J \\
& \sum_{j \in J} x_{ij} \leq 1, \quad \forall i \in I \\
& \sum_{i \in I} \sum_{j \in J} c_{ij}^\omega x_{ij} \geq \phi - \phi \delta^\omega, \quad \forall \omega \in \Omega \\
& x_{ij}, \delta^\omega \in \{0, 1\} \quad \forall i \in I, j \in J, \omega \in \Omega.
\end{aligned}
$$

and we do $I' = I$ in Procedure I of Section 2.2. The costs are randomly generated, such that c_{ij}^ω are drawn from $\mathcal{U}\{1, \ldots, 100\}$.

Throughout all the testing, we used CPLEX version 9.1, default settings, and we coded in C, on a Sun Java Workstation W2100z with 2 Opteron dual processors 2.6 GHz and 4 GB of RAM.

					CPLEX 9.1					Algorithm 3.1				
β	ϕ	$\|I\|$	$\|J\|$	$\|\Omega\|$	v^*	$\approx v^*$	gap	t	$\#\delta$'s	LB	UB	itera.	t	gap
100	1000	15	15	15	858.3	–	–	2	10	858	900	720	5	4.66
200	900	15	15	15	878.9	–	–	2	3	870	906	713	5	3.97
300	800	15	15	15	925.0	–	–	0	0	925	925	0	0	0
100	1000	15	15	30	789.8	–	–	80	24	786	854	767	4	7.96
200	900	15	15	30	777.7	–	–	12774	14	768	857	910	9	10.38
300	800	15	15	30	839.9	–	–	16	2	820	861	812	8	4.76
200	2000	30	30	30	1614.4	–	–	4768	23	1599	1708	861	24	6.38
300	1700	30	30	30	–	1723.7	1.23	7571	4	1717	1732	961	25	0.86
600	1600	30	30	30	1766.0	–	–	5	0	1766	1766	0	5	0
200	2000	30	30	60	–	1501.3	4.95	24444	49	1491	1601	567	22	6.87
300	1700	30	30	60	–	1569.6	4.97	37231	23	1548	1623	966	38	4.62
600	1600	30	30	60	–	1588.6	4.68	34394	9	1564	1613	967	39	3.04
400	4000	60	60	60	–	3058.6	7.33	22711	55	3044	3182	708	127	4.33
500	3500	60	60	60	–	3202.2	3.11	24150	28	3150	3241	862	159	2.81
700	3300	60	60	60	–	3350.1	2.25	24527	7	3299	3357	811	165	1.73
400	4000	60	60	100	–	2939.6	8.67	17896	94	2937	3313	415	206	11.35
500	3500	60	60	100	–	2985.3	10.16	23746	68	2973	3316	306	210	10.32
700	3300	60	60	100	–	3071.4	7.64	34741	37	3063	3275	188	198	6.47

Table 1: Performance of the algorithm for 18 instances

Table 1 summarizes the results for 18 instances out of our testbed. The first five columns describe the parameters and the size of the model. In order to appreciate the influence of the number of scenarios in the solution, the first and the second horizontal blocks only differ in the number of scenarios, analogously the third and the fourth horizontal blocks and the fifth and sixth, respectively. Moreover, for analyzing the influence of the value $\phi - \beta$ on the difficulty of the problem we have decreased this difference for any horizontal block: $800 - 300 < 900 - 200 < 1000 - 100$; $1600 - 600 < 1700 - 300 < 2000 - 200$; $3300 - 700 < 3500 - 500 < 4000 - 400$.

The following five columns in Table 1 are obtained by asking CPLEX to solve SSPP directly: v^* is $v(\text{SSPP})$; $\approx v^*$ and *gap* are the best solution that CPLEX obtains before giving the message "out of memory" and the smallest gap associated with this best solution (if "–" appears in those columns it is because CPLEX finds the optimal value and thus, the gap is zero); t is the time in seconds that CPLEX needs either for giving v^* or $\approx v^*$. Finally, $\#\delta$'s is the number of δ variables that take value one at the optimal solution (this number is obviously always smaller than $|\Omega|$). Moreover, we have checked that all our data instances are not easy instances in the sense of Section 2.1, i.e. $\phi < \min_{\omega\in\Omega} v(\text{SPP}_\omega)$.

The rest of the columns in the table measure the performance of our Lagrangian

β	ϕ	$\|I\|$	$\|J\|$	$\|\Omega\|$	$\uparrow LB$	$\tau/10$	τ_{opt}	$1.1 * \pi$	$0.66 * \pi$	itera.	t	gap
100	1000	15	15	15	1	615.2	7.8	17	19	828	8	5.23
200	900	15	15	15	0.2	607.8	6.6	12.4	19	804.2	8	4.18
300	800	15	15	15	0	514	5.6	9.4	15.2	609.6	6.4	1.05
100	1000	15	15	30	0.4	627.8	10.4	12.8	19	794.6	8.2	7.48
200	900	15	15	30	0.2	441.6	15	12.4	19	883.4	9.4	9.44
300	800	15	15	30	0.2	693.6	8.4	10.8	19	832.6	8.2	3.77
200	2000	30	30	30	2	869.8	4.8	15.2	19	925.8	25.2	6.67
300	1700	30	30	30	0	889.2	3.4	9.2	19	850.2	23.2	1.00
600	1600	30	30	30	0.4	221.8	1	3.6	7.6	383.8	10.4	0.09
200	2000	30	30	60	1.2	842	5.2	15	19	837.2	33.4	7.57
300	1700	30	30	60	0.2	817.4	6.8	11.8	19	883.8	35.6	4.92
600	1600	30	30	60	0	898.8	7.2	10	19	751.4	30.2	4.25
400	4000	60	60	60	1	831.6	2.2	9.2	19	699.6	121.8	4.62
500	3500	60	60	60	1.6	847.8	0.4	11.4	19	842.2	146.8	2.36
700	3300	60	60	60	0.8	673.4	6.2	7.6	19	749	127.4	0.99
400	4000	60	60	100	0	0	18	12.6	9	196.6	125.2	11.23
500	3500	60	60	100	0.4	0.2	18.6	10.4	13.4	119	178.2	10.33
700	3300	60	60	100	0.6	4.2	18.4	15.8	15	130.8	197.8	6.82

Table 2: Performance of the algorithm for 90 instances (each line is the average of five data instances)

algorithm: LB and UB are the final lower and upper bounds (values in these columns are rounded off); *itera* is the number of iterations that our algorithm needs for obtaining UB (it is not the total number of iterations because we have set this parameter to 1000, in fact, all the instances with positive gap have stopped at iteration number 1000); t is the total time of our algorithm in seconds; and gap is $((UB - LB)/UB) * 100$.

Table 1 illustrates that the smaller is the difference $\phi - \beta$, the smaller is usually the gap of our Lagrangian algorithm. We see on the table data instances with $\#\delta$'s$\simeq 0.2*|\Omega|$, with $\#\delta$'s$\simeq 0.5 * |\Omega|$ and with $\#\delta$'s$\simeq 0.8 * |\Omega|$. In general, the instances with $\#\delta$'s $\simeq 0.5 * |\Omega|$ are more difficult for CPLEX while for our Lagrangian algorithm the larger gaps are associated with cases $\#\delta$'s $\simeq 0.8 * |\Omega|$. On average, the reduction in time (seconds) is large and it is more evident for large problems. In cases where CPLEX cannot find the optimum, our algorithm frequently obtains smaller gaps in rather less time. Comparing the first and the second horizontal blocks, we observe that $\#\delta$'s increases when the number of scenarios also increases, analogously comparing the third and the fourth and the fifth and sixth, respectively; notice that $\#\delta$'s comes from 10 to 24, from 3 to 14, ..., from 7 to 37. Finally, in case *itera* was larger than 800 we have checked if the resulting gap was smaller with 1500 iterations and in all cases it did not improve.

Table 1 evidences that it is quite difficult to solve SSPP by just calling CPLEX for the cases that we have experimented with. Since, in order to fill in this table for the whole testbed of 90 instances, excessive space will be required, we have summarised in Table 2 our computational experiment, which concentrates on the performance of our Lagrangian algorithm for the testbed. Each line in the table gives the mean for five instances with the same characteristics and different random seeds. $\uparrow LB$ is the number of times that the lower bound is updated with the Lagrangian algorithm. The small values in this column indicate the efficiency of the Procedure III**:** see Subsection 2.2. $\tau/10$ is the number of times we do this computation. According to [3], $\tau/10$ occurs when the algorithm is near the optimum, so its is foreseeable that, for instance, if $\tau/10$ is 800 the value of τ has even decreased during the first 200 iterations (remember that the number of iterations is fixed to 1000). τ_{opt} is the number of times it occurs: in case that $\tau/10$ is not done, a condition should be satisfied for doing τ_{opt}. Next, $1.1 * \pi$ and $0.66 * \pi$ also count the number of times it occurs. Finally, the last vertical block of the table summarizes the number of iterations needed for finding the best upper bound, the total time required for the 1000 iterations and the final gap, $((UB - LB)/UB) * 100$. The information in this last vertical block strengthens the information shown in Table 1. The times are reasonable in all the 90 instances and gaps are small in the majority of cases.

5 Concluding Remarks

In this paper we have studied a scheme for solving large-scale stochastic set packing problems SSPP via one-stage scenario analysis, Lagrangian Decomposition and the Volume Algorithm for multipliers updating. The function is a composite mean–risk function. The problem is decomposed in a deterministic SPP to be iteratively solved by a general 0-1 optimization engine and a scenario based 0-1 knapsack problem. The upper bounding of the optimal solution can be performed in parallel given the independent character of the SPP solving and the knapsack problems to solve. From the computational experience that we have carried out, we can draw the following conclusions: (1) The mean–risk SSPP instances are very difficult to solve by general 0-1 optimiza-

tion packages; (2) The knapsack algorithm from [32] works very well; (3) SPP with totally unimodular matrices (assignment problem, flow problems, ...) are easy for the optimization engine CPLEX; (4) Our decomposition approach, jointly with the Volume Algorithm from [3], gives a computing time that is reasonable; (5) Its gap is small in the majority of the cases we have tested and, in particular, the computing time is much smaller (several orders of magnitude) than the time required by plain use of CPLEX, mainly, for the cases where it does not obtain the optimal solution with a comparable gap; and (6) The smaller the difference $\phi - \beta$, the smaller usually is the gap that we obtain.

Acknowledgement

The authors would give thanks for the fruitful discussions that have held with Francisco Barahona on the performance and description of the Volume Algorithm.

This research was partially supported by the grants OPTIMOS2 MTM2009-14039-C06-04 and MTM2007-67433-C02-02 from Ministerio de Ciencia y Tecnología, RDEF funds, 08716/PI/08 from Fundación Séneca, RM URJC-CM-2008-CET-3703 and RIEGOS CM from Comunidad de Madrid, P06-FQM-01364 and P06-FQM-01366 from Junta de Andalucía and PLANIN MTM2009-14087-C04-01 from Ministerio de Ciencia e Innovación, Spain.

References

[1] A. Alonso-Ayuso, L.F. Escudero, M.T. Ortuño. BFC, a Branch-and-Fix Coordination algorithmic framework for solving some types of stochastic pure and mixed 0-1 programs. *European Journal of Operational Research*, 151:503-519, 2003.

[2] R. Andrade, A. Lisser, N. Maculan, G. Plateau. Enhancing a Branch-and-Bound algorithm for two-stage stochastic integer network design-based models. *Management Science*, 52:1450-1455, 2006.

[3] F. Barahona, R. Anbil. The volume algorithm: Producing primal solutions with a subgradient methhod. *Mathematical Programming*, Ser. A, 87:385-399, 2000.

[4] J.F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 4:238-252, 1962.

[5] P. Beraldi, A. Ruszczynski. Beam search heuristic to solve stochastic integer problems under probabilistic constraints. *European Journal of Operational Research*, 167:35-47, 2005.

[6] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[7] J.R. Birge, F.V. Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.

[8] L. Cánovas, M. Landete, A. Marín. On the facets of the simple plant location polytope. *Discrete Applied Mathematics* 124 27-53, 2002.

[9] L. Cánovas, M. Landete, A. Marín. Facet obtaining procedures for set packing problems. *SIAM J. Discrete Math. 16(1),* 127-155, 2003.

[10] L. Cánovas, M. Landete, A. Marín. New formulations for the uncapacitated multiple allocation hub location problem. *EJOR* 172 274-292, 2006.

[11] C.C. Carøe, J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83:451-464, 1998.

[12] C.C. Carøe, R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37-45, 1999.

[13] S. De Vries and R.V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing* 15(3), 284-309 (2003).

[14] B. C. Dean. Approximation algorithms for stochastic scheduling problems. *Phd thesis, Massachusetts Institute of Technology*. MASS USA 2005.

[15] B. C. Dean. M.X. Goemans, J. Vondrak. Approximating the stochastic knapsack problem: the benefit of adaptivity. *Proceedings of the 45th annual IEEE Symposium on Foundations of Computer Science*. Rome, Italy 208-217, 2004.

[16] B. C. Dean, M.X. Goemans, J. Vondrak. Adaptivity and approximation for stochastic packing problems. *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms*. Vancouver, Canada 395-484, 2005.

[17] L.F. Escudero On a mixture of the fix-and-relax coordination and lagrangean substitution schemes for multistage stochastic mixed integer programming. *TOP*, 17:5-29, 2009.

[18] L.F. Escudero, A. Garín, M. Merino, G. Pérez. A two-stage stochastic integer programming approach as a mixture of Branch-and-Fix Coordination and Benders Decomposition schemes. *Annals of Operations Research*, 152:395-420, 2007.

[19] L.F. Escudero, A. Garín, M. Merino, G. Pérez. A multistage Stochastic Integer Programming model and algorithm for incorporating logical constraints in assets and liabilities management under uncertainty. *Computational Management Science*, 6:307-327, 2009.

[20] L.F. Escudero, A. Garín, M. Merino, G. Pérez. A general algorithm for solving two-stage stochastic mixed 0-1 first stage problems. *Computers and Operations Research*, 36:2590-2600, 2009.

[21] L.F. Escudero, A. Garín, M. Merino, G. Pérez. BFC-MSMIP: an exact Branch-and-Fix Coordination approach for solving multistage stochastic mixed 0-1 problems. *TOP* 17:96-122, 2009.

[22] M. Goemans, J. Vondrak. Stochastic covering and adiptivity, in: J.R. Correa, A. Hevia, M. Kiwi, editors. *LATIN 2006, LNCS 3887*, 532-543, 2006, Springer-Verlag.

[23] N. Groewe-Kuska, K. Kiwiel, M.P. Nowak, W. Römisch, I. Wegner. Power management in a hydro-thermal system under uncertainty by Lagrangian relaxation, In C. Greengard, A. Ruszczynski, editors. *Decision making under uncertainty: Energy and Power*, 39-70, 2001.

[24] M. Guignard, Lagrangean relaxation. *TOP*, 11:151-228, 2003.

[25] M. Held, R.M. Karp, The travelling salesman problem and the minimum spanning trees. *Operations Research*, 18:1138-1162, 1970.

[26] R. Hemmecke, R. Schultz. Decomposition methods for two-stage stochastic integer programs. In M. Grötschel, S.O. Krumke and J. Rambau, editors. *Online Optimization of Large Scale Systems*. Springer, 601-622, 2001.

[27] N. Jimenez Redondo, A.J. Conejo. Short-term hydro-thermal coordination by Lagrangian relaxation: solution of the dual problem. *IEEE Transactions on Power Systems*, 14:89-95, 1999.

[28] W.K. Klein Haneveld, M.H. van der Vlerk. Optimizing electricity distribution using integer recourse models. In S. Uryasev and P.M. Pardalos, editors. *Stochastic Optimization: Algorithms and Applications*. Kluwer Academic Publishers, 137-154, 2001.

[29] G. Laporte, F.V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133-142, 1993.

[30] G. Laporte, F.V. Louveaux. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50:415-423, 2002.

[31] G. Lulli, S. Sen. A heuristic procedure for stochastic integer progrmas with complete recourse. *European Journal of Operational Research*, 171:879-890, 2006.

[32] S. Martello, D. Pisinger, P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45:414-424, 1999.

[33] K. Munagala, P. Shi. The stochastic machine replenishment problem. *Proceedings of the 13th International Conference on Integer Programming and Combinatorial Optimization*. Bertinoro, Italy, 2008.

[34] G.L. Nemhauser, L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, 1998.

[35] L. Ntaimo, S. Sen. The million variable 'march' for stochastic combinatorial optimization. *Journal of Global Optimization* 32:385-400, 2005.

[36] L. Ntaimo, S. Sen. A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications* 40:299-319, 2008.

[37] W. Ogryczak, A. Ruszczynski. From stochastic dominance to mean–risk models: semi-deviations as risk measures. *European Journal of Operational Research*, 116:33-50, 1999.

[38] M.W. Padberg, On the facial structure of set packing polyhedra. *Mathematical Programming,* vol. 5, pp. 199–215,1973.

[39] B.T. Polyak. *Introduction to Optimization Software*. 1987.

[40] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37: 130-143, 1988.

[41] R.T. Rockafellar, S. Uryasev. Optimization of Conditional Value-at-Risk. *Journal of Risk*, 2:21-41, 2000.

[42] R.T. Rockafellar, R.J-B Wets. Scenario and policy aggregation in optimisation under uncertainty. *Mathematics of Operations Research*, 16:119-147, 1991.

[43] W. Römisch, R. Schultz. Multi-stage stochastic integer programs: An introduction. In M. Grötschel, S.O. Krumke, J. Rambau, editors. *Online Optimization of Large Scale Systems*. Springer, 581-600, 2001.

[44] T. Santoso, S. Ahmed, M. Goetschalckx, A. Shapiro. A stochastic programming approach for supply network design under uncertainty. *European Journal of Operational Research*, 167:96-115, 2005.

[45] R. Schultz. Stochastic programming with integer variables. *Mathematical Programming*, Ser. B 97:285-309, 2003.

[46] R. Schultz, M. Nowak, M. Westphalen A stochastic integer Programming model for incorporating day-ahead trading of electricity into hydro-thermal unit commitment. *Optimization and Engineering*, 6:163-176, 2005.

[47] R. Schultz, S. Tiedemann. Risk aversion via excess probabilities in stochastic programs with mixed-integer recourse. *SIAM Journal on Optimization*, 14:115-138, 2004.

[48] R. Schultz, S. Tiedemann. Conditional Value-at-Risk in stochastic programs with mixed integer recourse. *Mathematical Programming*, Series B 105:365-386, 2006.

[49] S. Sen, H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, Series A 106:203-223, 2006.

[50] S. Sen, J.L. Higle. The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, Series A 104:1-20, 2005.

[51] H.D. Sherali, J.C. Smith. Two-stage hierarchical multiple risk problems: models and algorithms. *Mathematical Programming*, Series A 120:403-427, 2009.

[52] H.D. Sherali, X. Zhu. On solving discrete two stage stochastic programs having mixed-integer first and second stage variables. *Mathematical Programming*, Series A 108:597-611, 2006.

[53] L. V. Snyder, M. S. Daskin. Reliability Models for faclity location: The expected failure cost case. *Transportation Science* 39(3) 440-416, 2005.

[54] S. Takriti, J.R. Birge. Lagrangian solution techniques and bounds for loosely coupled mixed-integer stochastic programs. *Operations Research*, 48:91-98, 2000.

[55] Y. Yuan, S. Sen. Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed integer programs. *INFORMS Journal of Computing*, 21:480-487, 2006.