Noname manuscript No.
(will be inserted by the editor)

Stochastic binary problems with simple penalties for capacity constraints violations *

B. Fortz · M. Labbé · F. Louveaux · M. Poss

Received: date / Accepted: date

Abstract This paper studies stochastic programs with first-stage binary variables and capacity constraints, using simple penalties for capacities violations. In particular, we take a closer look at the knapsack problem with weights and capacity following independent random variables and prove that the problem is weakly $\mathcal{NP}$-hard in general. We provide pseudo-polynomial algorithms for three special cases of the problem: constant weights and capacity uniformly distributed, subset sum with Gaussian weights and arbitrary random capacity, and subset sum with constant weights and arbitrary random capacity. We then turn to a branch-and-cut algorithm based on the outer approximation of the objective function. We provide computational results for the stochastic knapsack problem (i) with Gaussian weights and constant capacity and (ii) with constant weights and capacity uniformly distributed, on randomly generated instances inspired by computational results for the knapsack problem.

Keywords Stochastic programming · Knapsack problem · Pseudo-polynomial algorithm · Mixed-integer non-linear programming · Branch-and-cut algorithm.

* This research is supported by an Actions de Recherche Concertées (ARC) project of the Communautée française de Belgique. Michael Poss is a research fellow of the Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture (FRIA).

B. Fortz
Department of Computer Science, Faculté des Sciences, Université Libre de Bruxelles, Brussels, Belgium
E-mail: bfortz@ulb.ac.be

M. Labbé
Department of Computer Science, Faculté des Sciences, Université Libre de Bruxelles, Brussels, Belgium.
E-mail: mlabbe@ulb.ac.be

F. Louveaux
Louvain School of Management, FUNDP, University of Namur, Namur, Belgium.
E-mail: flouveau@fundp.ac.be

M. Poss
Department of Computer Science, Faculté des Sciences, Université Libre de Bruxelles, Brussels, Belgium.
E-mail: mposs@ulb.ac.be

1 Introduction

An important class of optimization problems with binary variables involves capacity constraints. In some of these problems, we must transport a set of goods from suppliers to customers without exceeding the capacity of transporting units. In others, we must prescribe a production plan that respects the capacity of producing units. An important subproblem that arises in both examples is how to fill the capacity of each transporting (or producing) unit in the best way, leading to the knapsack problem. Many efficient solution methods for capacitated problems benefit from fast algorithms for the knapsack problem, for instance, by generating cover cuts or by providing strong bounds within decomposition algorithms, see [3, 26], among others. In what follows, we work more specifically with uncertain parameters.

Different frameworks are able to handle capacity constraints with parameters following random variables. Probabilistic constraints are convenient to use because they do not interfere with the structure of the original model. However, they yield very difficult optimization problems, although recent works have proposed solution methods for integer programs with probabilistic constraints [4, 17]. Herein we study a different framework, where additional capacity can be added at a given unitary cost, yielding a stochastic program with linear simple-recourse. This leads to tractable reformulations, as long as we can handle the multi-variate integral involved in the recourse function. Efficient solution methods for stochastic linear programs with simple-recourse have been developed in the past, see for instance [31]. The case of integer recourse has also been studied extensively [12, 22].

In this paper, we apply simple penalty recourse to the difficult class of binary problems with capacity constraints. We look more specifically at the stochastic knapsack problem and derive two types of results. First, we prove that three special cases of the stochastic knapsack problem are weakly $\mathcal{NP}$-hard:

- Stochastic knapsack with fixed weights and uniformly distributed capacity.
- Stochastic subset sum with Gaussian weights and arbitrary random capacity.
- Stochastic subset sum with fixed weights and arbitrary random capacity.

Then, we show that the LP/NLP algorithm of Quesada and Grossman [29] enables us to solve efficiently the stochastic knapsack problem with Gaussian random variables. Our computational results show that this algorithm is able to solve in less than a minute problems involving up to a few thousands of variables, outperforming previous works on the problem [10, 20]. Finally, we compare the pseudo-polynomial algorithms and the LP/NLP algorithm on the special cases. It turns out that some of the stochastic subset sum problems can be solved extremely fast by using the specialized algorithm from Pisinger [27].

The paper is structure as follows. Section 2 introduces the general models studied herein and provides some examples, including the stochastic knapsack problem. Section 3 studies the complexity of the special cases of the stochastic knapsack problem, providing pseudo-polynomial algorithms. Section 4 presents an algorithm for the general models and provide extensive computational experiments. A comparison of the general algorithm and the pseudo-polynomial approaches is also presented for the special cases. The paper concludes with Section 5.

2 The models

2.1 Capacity constraints

In this paper, we study binary problems featuring capacity constraints, that is

$$\sum_{i=1}^{n} a_i x_i \leq a_0, \tag{1}$$

where a_i and a_0 are positive numbers. We denote such "capacitated problems" by (CAP) in what follows. When coefficients a_i and a_0 are random variables, (1) yields one equation for each scenario in a set of scenarios Ω, which is not convenient when Ω contains a large (sometimes infinite) number of scenarios or when some of the probability weights are very small. Replacing (1) by the probabilistic constraint

$$P\left(\sum_{i=1}^{n} a_i x_i \leq a_0\right) \geq 1 - \epsilon, \tag{2}$$

for $\epsilon \in (0, 1)$ is thus a natural choice in the stochastic context. Although easy to incorporate into models, probabilistic constraints yield, in general, very difficult optimization problems. Moreover, they neglect the amount by which the constraint is violated while this can be crucial in some applications. For instance, electrical lines are built to resist against higher power flow than expected for a reasonable period of time, while an extremely high power flow even for a short period of time is likely to burn some of the lines. These two problems can be handled by the introduction of a penalty y for capacity violation. Then, (1) becomes

$$\sum_{i=1}^{n} a_i(\omega) x_i \leq a_0(\omega) + y(\omega) \qquad \omega \in \Omega, \tag{3}$$

and the weighted expected violation of the capacity $K\mathbb{E}_a[y(\omega)]$ is subtracted from the objective function (for a maximization problem), where $\mathbb{E}_a[y(\omega)] = \int_A y(a) dF_a(a)$ and $F_a : \mathbb{R}^{n+1} \to \mathbb{R}_+$ is the distribution function of the random vector $a = (a_0, a_1, \ldots, a_n) : \Omega \to A \subset \mathbb{R}^{n+1}$. This simple-recourse formulation weights capacity violation through parameter K and yields tractable optimization problems, thus addressing the two drawbacks of probabilistic constraints mentioned above. In what follows, we denote by (SCAP) problems featuring penalized capacity constraints (3).

Because constraints (3) are satisfied at equality in any optimal solution with $y(\omega) > 0$, variable $y(\omega)$ can be replaced with $\max(0, \sum_{i=1}^{n} a_i(\omega) x_i - a_0(\omega))$, resulting in the non-linear objective function:

$$\sum_{i=1}^{n} c_i x_i - K\mathbb{E}_a\left[\max\left(0, \sum_{i=1}^{n} a_i(\omega) x_i - a_0(\omega)\right)\right]. \tag{4}$$

Note that models (2) and (4) are asymptotically equivalent when K grows to infinity, see for instance [11].

We review next two examples of (SCAP) that have been previously studied in the literature.

2.1.1 Traveling salesman

Given a complete directed graph with node set N, and travel costs c_{ij}, $i, j \in N$, the traveling salesman problem (TSP) looks for a cycle of minimum cost passing exactly once through each node. A generalization of the problem considers that a time limit T is given and that each edge has a travel time. The obtained tour can not exceed the time limit. The stochastic extension written below describes each travel time by a random variable a_{ij}. Whenever the total travel time of a tour exceeds T, a penalty must be paid at the cost of K per unit of delay. This results in the subsequent problem, which has been extended to a three-index model by Laporte *et al.* [21]:

$$\begin{aligned}
\min \quad & \sum_{i,j\in N} c_{ij}x_{ij} + K\mathbb{E}_a\left[\max\left(0, \sum_{i,j\in N} a_{ij}(\omega)x_{ij} - T\right)\right] \\
\text{s.t.} \quad & \sum_{j\in N} x_{ij} = 1 && i \in N \\
& \sum_{i\in N} x_{ij} = 1 && j \in N \\
& \sum_{i,j\in S} x_{ij} \leq |S| - 1 && V \neq S \subset V \\
& x_{ij} \in \{0,1\}.
\end{aligned}$$

The only difference between the problem above and the deterministic (TSP) is the presence of a non linear term in the objective.

2.1.2 Elastic generalized assignment

The generalized assignment problem looks for optimal assignment of n jobs to m machines that have a limited capacity. The problem has numerous applications, including fixed-charge plant location models and scheduling of tasks in servers, among others. Being $\mathcal{NP}$-hard, the problem has attracted a considerable amount of attention in the past, leading to efficient solution methods, see [9] and [26], among others. Brown and Graves [8] introduce penalties for unsatisfied capacity constraints, yielding the elastic generalized assignment problem. Spoerl and Wood extended the latter to stochastic settings in [30], yielding

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{j=1}^{m} K_j\mathbb{E}_a\left[\max\left(0, \sum_{i=1}^{n} a_i x_{ij} - a_{0j}\right)\right] \\
\text{(SEGAP)} \qquad & \sum_{j=1}^{m} x_{ij} = 1 && i = 1, \ldots, n \qquad (5) \\
& x_{ij} \in \{0,1\}.
\end{aligned}$$

The resulting (SEGAP) presents the structure of the assignment problem with a non-linear cost function.

2.2 The knapsack problem

Given a set of items N, the classical knapsack problem looks for a subset of N not exceeding the capacity a_0 of the knapsack and maximizing the total profit. Each item $i \in \mathsf{N}$ has a profit p_i and a weight a_i. Our two-stage stochastic version of the problem considers that weights $a_i(\omega)$ and capacity $a_0(\omega)$ are independent random variables. The subset of items is chosen first, then any real amount of additional capacity can be bought at the unitary price of K, depending on the scenario ω. The objective function maximizes the profit of the chosen subset minus the expected cost of the additional capacity:

$$\text{(SKP)} \qquad \max \quad \sum_{i\in\mathsf{N}} p_i x_i - K\mathbb{E}_a\left[\max\left(0, \sum_{i\in\mathsf{N}} a_i(\omega)x_i - a_0(\omega)\right)\right]$$
$$\text{s.t.} \quad x_i \in \{0,1\}.$$

This problem can be thought of as the following resource allocation problem [16]. A decision maker has to choose a subset of n known alternative projects to take on. For this purpose, a known quantity a_0 of relatively low-cost resource is available to be allocated. Any additional amount of resource required can be obtained at a known incremental cost of K per unit of resource. The amount a_i of resource required by each project is not known at the time the decision has to be made, but we assume that the decision maker has an estimate of the probability distribution of those a_i. Finally, each project i has an expected reward of p_i.

(SKP) has been first formulated by Cohn and Barnhart [10] who consider that a_0 is given and that a_i follow Gaussian variables. They derive basic properties and propose a simple branch-and-bound algorithm that they test on an example with 15 variables. Recently, Kosuch and Lisser [20] use a stochastic gradient method to solve (SKP) with Gaussian variables. They solve the problem up to 150 variables in at most two hours. Kleywegt *et al.* [16] work on a similar model with discrete random variables to test their sample average approximation method. Other ways of considering uncertainties in the parameters of the knapsack problem include knapsack with a probabilistic constraint [19,24], robust knapsack [18] and dynamic knapsack [15], among others.

3 Pseudo-polynomial cases

In this section we show that Problem (SKP) is in general weakly $\mathcal{NP}$-hard. We then turn to special cases of (SKP) that can be solved in pseudo-polynomial time.

Proposition 1 *Problem (SKP) is at least as hard as the classical knapsack problem.*

Proof Consider (SKP) for a unique scenario. Writing the capacity constraint explicitly, the problem reads:

$$\max_{x\in\mathsf{B}, y\geq 0} \left\{ \sum_{i\in\mathsf{N}} p_i x_i - Ky \text{ s.t. } \sum_{i\in\mathsf{N}} a_i x_i \leq a_0 + y \right\}. \tag{6}$$

Taking K large enough, y is equal to 0 in any solution so that (6) is equivalent to the classical knapsack problem. □

In the remainder of this section, we use the notations B and $\mathcal{B}$ to denote $\{0,1\}^n$ and $[0,1]^n$, respectively; the summation $\sum$ refers to the sum over N unless stated otherwise.

3.1 Fixed weights and uniformly distributed capacity

In this section, we consider that weights a_i are fixed so that (SKP) becomes:

$$\max_{x\in\mathsf{B}} \sum p_i x_i - K \int_0^{\sum a_i x_i} \Big(\sum a_i x_i - a_0\Big)\, dF(a_0). \tag{7}$$

Assuming that a_0 is uniformly distributed between positive integers $\underline{a_0}$ and $\overline{a_0}$ and that all parameters are integers we show next that the optimization problem can be solved by a pseudo-polynomial algorithm under a mild assumption on its parameters. Note that this problem can also be seen as a robust knapsack problem with linear penalty [25]. With a_0 uniformly distributed, (7) becomes:

$$\max_{x\in\mathsf{B}} \sum p_i x_i - \frac{K}{\overline{a_0}-\underline{a_0}} \int_{\min(\sum a_i x_i, \underline{a_0})}^{\min(\sum a_i x_i, \overline{a_0})} \Big(\sum a_i x_i - a_0\Big)\, da_0, \tag{8}$$

where we assume $\sum a_i > \underline{a_0}$ to avoid the trivial solution $x = (1, \ldots, 1)$.

Theorem 1 *Problem* (8) *is in general $\mathcal{NP}$-hard. However, if* $\Big(p_i + \frac{K a_i}{2(\overline{a_0}-\underline{a_0})}(2\underline{a_0} - a_i)\Big) \geq 0$ *for each $i \in \mathsf{N}$, Z can be maximized in $O(nK\sum a_i)$.*

For a large number of items with individual volumes small enough, this condition is likely to be satisfied. Applying an argument similar to Proposition 1 to a problem with $\overline{a_0} = \underline{a_0} + 1$, we obtain:

Lemma 1 *Problem* (8) *is at least as hard as its deterministic counterpart.*

The remainder of the section shows how (8) can be solved in pseudo-polynomial time whenever $\Big(p_i + \frac{K a_i}{2(\overline{a_0}-\underline{a_0})}(2\underline{a_0} - a_i)\Big) \geq 0$ for each $i \in \mathsf{N}$. Let Z be the the function to be maximized in (8). We rewrite Z as follows:

$$Z(x) = \begin{cases} Z_1(x) = & \sum p_i x_i & for\ \sum a_i x_i \leq \underline{a_0} \\ Z_2(x) = \sum p_i x_i - \frac{K}{2(\overline{a_0}-\underline{a_0})}\big(\sum a_i x_i - \underline{a_0}\big)^2 & for\ \underline{a_0} \leq \sum a_i x_i \leq \overline{a_0} \\ Z_3(x) = & K\frac{\overline{a_0}+\underline{a_0}}{2} + \sum (p_i - K a_i) x_i & for\ \overline{a_0} \leq \sum a_i x_i. \end{cases} \tag{9}$$

Lemma 2 *Let x^*, x_1^*, x_2^* and x_3^* be optimal solutions of* $\max_{\mathsf{B}} Z(x)$, $\max_{\mathsf{B}}\{Z_1(x)$ *s.t.* $\sum a_i x_i \leq \underline{a_0}\}$, $\max_{\mathsf{B}} Z_2(x)$ *and* $\max_{\mathsf{B}}\{Z_3(x)$ *s.t.* $\sum a_i x_i \geq \overline{a_0}\}$, *respectively. Then,* $Z(x^*) = \max(Z_1(x_1^*), Z_2(x_2^*), Z_3(x_3^*))$.

Proof We can relax the domain restriction $\underline{a_0} \leq \sum a_i x_i \leq \overline{a_0}$ for Z_2 because $Z_1(x) \geq Z_2(x)$ and $Z_3(x) \geq Z_2(x)$ for any $x \in \mathcal{B}$. □

The following three lemmas show that each of the problems from Lemma 2 can be solved in pseudo-polynomial time.

Lemma 3 *Maximizing $Z_1(x)$, for x binary and $\sum a_i x_i \leq \underline{a_0}$, can be done in $O(n\sum a_i)$.*

Proof This is a knapsack problem, which can be optimized in $O(n\underline{a_0})$ and thus in $O(n\sum a_i)$ because $\underline{a_0} < \sum a_i$. □

Lemma 4 *Maximizing $Z_3(x)$, for x binary and $\sum a_i x_i \geq \overline{a_0}$, can be done in $O(n\sum a_i)$.*

Proof In the following, we assume that $\sum a_i > \overline{a_0}$, otherwise $\sum a_i x_i$ is always smaller than $\overline{a_0}$ so that the problem does not have a solution. We show that the problem is a knapsack problem.

1. Define $\mathsf{M} = \{i \in \mathsf{N} \mid p_i - Ka_i < 0\}$ and $\tilde{a_0} = \sum a_i - \overline{a_0}$; let x^* be the solution to the following knapsack problem ($x_i^* = 0$ for $i \in \mathsf{N}/\mathsf{M}$)

$$\max_{\mathsf{B}} \left(\sum_{i \in \mathsf{M}} (Ka_i - p_i) x_i \quad \text{s.t} \quad \sum_{i \in \mathsf{M}} a_i x_i \leq \tilde{a_0} \right). \tag{10}$$

2. An optimal solution to $\max_{\mathsf{B}} \{Z_3(x) \text{ s.t. } \sum a_i x_i \geq \overline{a_0}\}$ is given by $x_i = 1 - x_i^*$, for each $i \in \mathsf{N}$.

Then, because $\tilde{a_0} \leq \sum a_i$, (10) can be solved in $O(n \sum a_i)$. □

Lemma 5 *If* $\left(p_i + \frac{Ka_i}{2(\overline{a_0} - \underline{a_0})}(2\underline{a_0} - a_i)\right) \geq 0$ *for each* $i \in \mathsf{N}$, *maximizing* $Z_2(x)$, *for* x *binary, can be done in* $O(nK \sum a_i)$.

Proof Expanding the square in Z_2, and using the identity $x_i^2 = x_i$ because $x \in \mathsf{B}$, we obtain:

$$-\frac{K\underline{a_0}^2}{2(\overline{a_0} - \underline{a_0})} + \max_{x \in \mathsf{B}} \left(\sum_{i \in \mathsf{N}} \left(p_i + \frac{Ka_i}{2(\overline{a_0} - \underline{a_0})}(2\underline{a_0} - a_i) \right) x_i - \frac{K}{\overline{a_0} - \underline{a_0}} \sum_{\substack{i,\, j \,\in\, \mathsf{N} \\ i \neq j}} a_i a_j x_i x_j \right). \tag{11}$$

Assuming that $\left(p_i + \frac{Ka_i}{2(\overline{a_0} - \underline{a_0})}(2\underline{a_0} - a_i)\right) \geq 0$ for each $i \in \mathsf{N}$, linear coefficients of (11) are all positive. We need them to be integer as well to apply the tools from optimization of pseudo boolean functions.

Multiplying all terms by $4K(\overline{a_0} - \underline{a_0})$, (11) becomes a particular case of half-products [2]

$$f = \sum_{i \in \mathsf{N}} c_i x_i - \sum_{\substack{i,\, j \,\in\, \mathsf{N} \\ i \neq j}} a_i b_j x_i x_j,$$

where $c \mapsto 4K(\overline{a_0} - \underline{a_0}) \left(p + \frac{Kw}{2(\overline{a_0} - \underline{a_0})}(2\underline{a_0} - a_i)\right)$ and $a = b \mapsto 2Kw$. Badics and Boros [2] provide a dynamic programming algorithm for general half-products with positive coefficients. Its running time is $O(nA)$, where $A = 2K \sum a_i$. □

Besides this dynamic programming approach, new versions of optimization softwares, including CPLEX 11, can manage maximization of integer problems with a concave and quadratic objective function. Nevertheless, we show in Section 4.2 that the LP/NLP algorithm described in the Section 4.1.1 solves (11) much faster than CPLEX 11 and the algorithm from Badics and Boros do.

3.2 Subset sum

A well known particular case of the deterministic knapsack problem is the subset sum problem that assumes that weight a_i is equal to profit p_i for each item $i \in \mathsf{N}$. Even though weakly $\mathcal{NP}$-hard to solve, adapted algorithms can have a much better behavior for this problem than for the general knapsack.

To the best of our knowledge, no stochastic version of the subset sum has been addressed in the literature so far. To define the stochastic subset sum, we replace the deterministic constraint $a = p$ by:

- $\mathbb{E}_a[a] = p$,
- $\mathrm{Var}[a] = \lambda p$ for some $\lambda \geq 0$,

where $\mathbb{E}_a[v] = (\mathbb{E}_a[v_1], \ldots, \mathbb{E}_a[v_n])$ and $\mathrm{Var}[v] = (\mathrm{Var}[v_1], \ldots, \mathrm{Var}[v_n])$ for any random vector v. The constraint $\mathbb{E}_a[a] = p$ is the direct extension of $a = p$. Then, to enforce the link between p and a we also impose $\mathrm{Var}[a] = \lambda p$. Note that the case $\lambda = 0$ results in a deterministic knapsack, where additional capacity can be purchased at the incremental cost of K per unit, see problem (6).

3.2.1 Gaussian weights

In this section, we assume that weights a_i, $i \in \mathsf{N}$, are independent Gaussian variables with parameters μ_i and $\sigma_i^2 = \lambda\mu_i$, $i \in \mathsf{N}$, for some $\lambda \geq 0$, and that capacity a_0 is a positive random variable. This is motivated by the following summation property: if $a_1, \ldots, a_n$ are independent Gaussians of mean μ_i and variance σ_i^2, and x_i are real numbers, then $Y := \sum x_i a_i \sim \mathcal{N}(\mu(x), \sigma^2(x))$, with $\mu(x) = \sum x_i\mu_i$ and $\sigma^2(x) = \sum x_i^2\sigma_i^2$. Moreover, Gaussians are often used to represent the error made on estimations of parameters for many physical and economical problems.

Theorem 2 *Consider the problem*

$$\max_{x \in \mathsf{B}} \sum \mu_i x_i - K\mathbb{E}_a\left[\max\left(0, \sum a_i(\omega)x_i - a_0(\omega)\right)\right], \tag{12}$$

where $a_i \sim \mathcal{N}(\mu_i, \lambda\mu_i)$, $0 \leq \lambda \leq 1$, μ is an integer vector and a_0 is a positive random variable. Problem (12) *is weakly $\mathcal{NP}$-hard and can be solved by a pseudo-polynomial algorithm in $O(n\sum \mu_i)$.*

The fact that (12) is at least weakly $\mathcal{NP}$-hard easily follows from an argument similar to the one used in Proposition 1, taking $\lambda = 0$. The rest of the section shows how to construct a pseudo-polynomial algorithm for (12). Let $a(x) = \sum a_i x_i \sim \mathcal{N}(\mu(x), \sigma^2(x))$, so that the usual recourse function reads

$$\mathcal{Q}(x) = -K\mathbb{E}_a[\max(0, a(x;\omega) - a_0(\omega))],$$

and consider the auxiliary function

$$\mathcal{R}(x) = -K\mathbb{E}_a[\max(0, \hat{a}(x;\omega) - a_0(\omega))],$$

where $\hat{a}(x) \sim \mathcal{N}(\mu(x), \hat{\sigma}^2(x))$, with $\mu(x) = \sum \mu_i x_i$ as before and $\hat{\sigma}^2(x) = \sum \sigma_i^2 x_i$. Note that for each $i \in \mathsf{N}$, $x_i^2 = x_i$ when $x \in \mathsf{B}$ so that $\mathcal{Q}(x) = \mathcal{R}(x)$ when $x \in \mathsf{B}$.

We define then $Z_{\mathcal{Q}}(x) = \sum \mu_i x_i + \mathcal{Q}(x)$ and $Z_{\mathcal{R}}(x) = \sum \mu_i x_i + \mathcal{R}(x)$, so that functions $Z_{\mathcal{Q}}(x)$ and $Z_{\mathcal{R}}(x)$ coincide on B, and $\max_{\mathsf{B}} Z_{\mathcal{Q}}(x) = \max_{\mathsf{B}} Z_{\mathcal{R}}(x)$. In what follows, we focus on the maximization of $Z_{\mathcal{R}}$. This is motivated by the following property:

Lemma 6 *If $\sigma_i^2 = \lambda\mu_i$ for each $1 \leq i \leq n$ and some $\lambda \geq 0$, then there exists a function $\hat{Z} : [0, \sum \mu_i] \to \mathbb{R}$ such that for all $x \in \mathcal{B}$, $Z_{\mathcal{R}}(x) = \hat{Z}(z)$ with $z = \sum \mu_i x_i$.*

Proof By definition of z, $Z_{\mathcal{R}}$ can be rewritten as $z + \mathcal{R}(x)$. Then, we see that $\mu(x) = \sum \mu_i x_i = z$ and $\hat{\sigma}(x) = \sqrt{\sum \sigma_i^2 x_i} = \sqrt{\lambda \sum \mu_i x_i} = \sqrt{\lambda z}$ proving the result. □

Note that $\mathcal{R}(x) = \mathcal{Q}(x)$ only when $x \in \mathsf{B}$; when $x \in \mathcal{B}/\mathsf{B}$ these functions may be different. In particular, neither $Z_{\mathcal{R}}$ nor $\hat{Z}$ inherit from the concavity of recourse functions [5]. Nevertheless, we can prove the result analytically for $\hat{Z}$:

Lemma 7 *If $0 < \lambda \leq 1$ and a_0 takes positive values only, the function $\hat{Z}$ is concave on its domain.*

Proof Let f and F_0 be the density function of $\mathcal{N}(0,1)$ and the distribution function of a_0, respectively. $\hat{Z}$ is defined by the following expression

$$\hat{Z}(z) = z - K \int_0^\infty \left\{ \frac{1}{\lambda z} \int_0^\infty f\left(\frac{a - z + a_0}{\lambda z}\right) da \right\} dF_0(a_0). \tag{13}$$

Following [10], among others, the inner integral can be simplified, yielding:

$$\hat{Z}(z) = z - K \int_0^\infty \left\{ \lambda z f\left(\frac{a_0 - z}{\lambda z}\right) + (z - a_0) G\left(\frac{a_0 - z}{\lambda z}\right) \right\} dF_0(a_0), \tag{14}$$

where $G = 1 - \Phi$ and Φ is the distribution function of $\mathcal{N}(0,1)$. Computing the second derivative of (14) for any $z > 0$, we obtain:

$$\hat{Z}''(z) = -K \int_0^\infty \left\{ \frac{\lambda^{-2}(z + a_0)^2 - z}{8\lambda\sqrt{\pi}\sqrt{z^5}} e^{-\frac{(z-a_0)^2}{2\lambda^2 z}} \right\} dF_0(a_0). \tag{15}$$

The integrand of (15) is non-negative when $a_0 \geq 0$ and F_0 is equal to 0 otherwise, so that $\hat{Z}''(z)$ is non-positive for all $z > 0$. □

Recalling that $\sigma^2 = \lambda\mu$, the assumption $\lambda \leq 1$ becomes $\sigma^2 \leq \mu$. We see in Section 4.1 that this assumption is required if we want $P(a_i \leq 0)$ to be negligible. Hence, in the following we always assume $\sigma^2 \leq \mu$, so that the function $\hat{Z}$ is concave.

Because $\hat{Z}$ is concave, it has at most one maximum. Suppose that we can compute the maximum z^* of $\hat{Z}$ over $\mathbb{R}^+$, which may be greater than $\sum \mu_i$. The concavity of $\hat{Z}$ implies that

$$z_1 \leq z_2 \leq z^* \Rightarrow \hat{Z}(z_1) \leq \hat{Z}(z_2) \qquad \text{and} \qquad z^* \leq z_2 \leq z_1 \Rightarrow \hat{Z}(z_1) \leq \hat{Z}(z_2),$$

for any $z_1, z_2 \in [1, \sum \mu_i]$. Recalling that $\hat{Z}(\sum \mu_i x_i) = Z_{\mathcal{R}}(x)$, we can write similar inequalities for $Z_{\mathcal{R}}$:

$$\sum \mu_i x_{1i} \leq \sum \mu_i x_{2i} \leq z^* \Rightarrow Z_{\mathcal{R}}(x_1) \leq Z_{\mathcal{R}}(x_2)$$
$$\text{and} \quad z^* \leq \sum \mu_i x_{2i} \leq \sum \mu_i x_{1i} \Rightarrow Z_{\mathcal{R}}(x_1) \leq Z_{\mathcal{R}}(x_2),$$

for any fractional vectors $x_1, x_2 \in \mathcal{B}$.

Hence, the closer $\sum \mu_i x_i$ is to z^*, the higher is $Z_{\mathcal{R}}(x)$. Then, two situations can happen. If $\sum \mu_i \leq z^*$, the closest $z = \sum \mu_i x_i$, $x \in \mathsf{B}$, to z^* is given by $x_i^* = 1$ for each $i \in \mathsf{N}$. This x^* is the solution to (12). If $\sum \mu_i > z^*$, we must look for x_1^* and x_2^* in B that minimize the distances between $\sum \mu_i x_{ji}^*$ and z^*, where $\sum \mu_i x_{1i}^* \leq z^*$

and $\sum \mu_i x^*_{2i} \geq z^*$. Namely, we need to solve two subset sum problems written below, where z^* has been replaced by $\lfloor z^* \rfloor$ and $\lceil z^* \rceil$ because of the integrality of μ and x:

$$\begin{aligned} \max \quad & \sum_{i \in \mathsf{N}} \mu_i x_i \\ \text{s.t.} \quad & \sum \mu_i x_i \leq \lfloor z^* \rfloor \\ & x_i \in \{0,1\}, \end{aligned} \tag{16}$$

and

$$\begin{aligned} \min \quad & \sum_{i \in \mathsf{N}} \mu_i x_i \\ \text{s.t.} \quad & \sum \mu_i x_i \geq \lceil z^* \rceil \\ & x_i \in \{0,1\}. \end{aligned} \tag{17}$$

Denote by x^*_1 and x^*_2 solutions to (16) and (17). A solution to (12) is given by $x^* \in \{x^*_1, x^*_2\}$ such that $Z_{\mathcal{R}}(x^*) = \max(Z_{\mathcal{R}}(x^*_1), Z_{\mathcal{R}}(x^*_2))$.

Problems (16) and (17) are weakly polynomial, because they are particular cases of the knapsack problem. Then, $\lfloor z^* \rfloor$ and $\lceil z^* \rceil$ can be computed in $O(\log_2 \sum \mu_i)$, using a dichotomic search based on the sign of $\hat{Z}'$.

3.2.2 Fixed weights

We show next a result similar to Theorem 2 when a_i, $i \in \mathsf{N}$, are fixed and a_0 is an arbitrary random variable.

Theorem 3 *Consider the problem*

$$\max_{x \in \mathsf{B}} \sum a_i x_i - K\mathbb{E}_a[\max(0, \sum a_i x_i - a_0(\omega))], \tag{18}$$

where a_0 is a random variable. Problem (18) *is weakly $\mathcal{NP}$-hard and can be solved by a pseudo-polynomial algorithm in $O(n \sum a_i)$.*

Proof The reduction from Proposition 1 holds when the variance of a_0 is zero and K is large enough. Let Z be the objective function from (18). We must prove that Z has the same properties as $Z_{\mathcal{R}}$ from previous section so that the same argument as in the proof of Theorem 2 can be applied:

- Z depends only on $z = \sum a_i x_i$: $Z(x) = \hat{Z}(z)$.
- $\hat{Z} : [0, \sum a_i] \to \mathbb{R}$ is concave.
- We can compute $\lceil z^* \rceil$ and $\lfloor z^* \rfloor$ in $O(n \sum a_i)$.

It is straightforward to see that Z depends only on $z = \sum a_i x_i$, with $\hat{Z}(z) = z - \mathbb{E}_{a_0}[\max(0, z - a_0(\omega))]$.

Then, the concavity of Z (and therefore $\hat{Z}$) follows from a general result in stochastic programming [5]. Using $\delta > 0$ small enough, we can use compute $\lceil z^* \rceil$ and $\lfloor z^* \rfloor$ by a dichotomic search based on the sign of $\hat{Z}(x) - \hat{Z}(x + \delta)$. □

4 Algorithms and computational experiments

In this section, we assume that random vector a is absolutely continuous, with density function f_a. One of the main difficulty of (SCAP) is to evaluate the concave expectation term from (4):

$$\begin{aligned}\mathcal{Q}(x) &= -K\mathbb{E}_a\left[\max\left(0, \sum_{i=1}^{n} a_i(\omega)x_i - a_0(\omega)\right)\right] \\ &= -K\int_A f_a(a)\max\left(0, \sum_{i=1}^{n} a_i x_i - a_0\right) da_0 da_1 \dots da_n. \end{aligned} \tag{19}$$

For an arbitrary continuous random vector a, the multivariate integral (19) is non-trivial and must be solved using efficient packages for numerical integration, see [28]. To avoid this computational burden, we restrict ourselves to particular cases involving Gaussian and uniform distributions in the next subsections. Note that for special distributions, such as the exponential, the recourse function has a closed form [13].

4.1 General case

4.1.1 LP/NLP algorithm

Problem (SCAP) belongs to the class of mixed-integer non-linear programs, which have witnessed a tremendous attention in recent years. Efficient algorithms and solvers are now available to handle convex MINLP, see [1,6,7]. Moreover, results from [7] suggest that LP/NLP algorithms are particularly efficient to handle problems with linear constraints only, such as (SCAP). Notice that these particular LP/NLP algorithms turn out to be branch-and-cut algorithms where cuts are linearizations of the objective function. The main steps of our LP/NLP algorithm are explained in detail in what follows.

Consider the following linearly constrained convex MINLP:

$$\text{(P)} \qquad \begin{aligned} \max \quad & h(x) \\ \text{s.t.} \quad & Ax \leq b \\ & x_i \in \{0,1\}, \end{aligned}$$

where h is assumed concave and differentiable. Problem (P) is equivalent to

$$\begin{aligned} \max \quad & \gamma \\ \text{s.t.} \quad & h(\overline{x}) + \sum_{i=1}^{n} \frac{\partial h}{\partial x_i}(\overline{x})(x_i - \overline{x}_i) \geq \gamma \qquad \overline{x} \in \mathbb{R}^n_+ \\ & Ax \leq b \\ & x_i \in \{0,1\}, \end{aligned} \tag{20}$$

which has an infinite number of constraints. The main idea of outer-approximation is that, given a sensitivity parameter $\epsilon > 0$, only a finite number of constraints (20) are

required in a solution. Given a cut pool R, we define the upper bounding problem

$$\text{(MP)} \qquad \begin{array}{lll} \max & \gamma & \\ \text{s.t.} & h(\overline{x}) + \sum_{i=1}^{n} \frac{\partial h}{\partial x_i}(\overline{x})(x_i - \overline{x}_i) \geq \gamma & \overline{x} \in \mathsf{R} \\ & Ax \leq b & \\ & x_i \in \{0,1\}. & \end{array}$$

Our algorithm `lp/nlp` solves (MP) with the branch-and-cut described in Algorithm 1. T represents the branch-and-cut tree and solving a node $o' \in \mathsf{T}$ means solving the LP relaxation of (MP) augmented with branching constraints of o'.

Algorithm 1: `lp/nlp`

```
begin                                                    /* Initialization */
    T = {o} where o has no branching constraints;
    UB = +∞;
end
while T is nonempty do
    select a node o' ∈ T;
    T ← T\{o'};                                  /* withdraw node o' from the tree */
    solve o';
    let (γ̄, x̄) be an optimal solution;
    if γ̄ < UB then
        if x̄ ∉ {0,1}^n and depth(o') ≥ 1 then
            branch, resulting in nodes o* and o**;
            T ← T ∪ {o*, o**};                       /* add children to the tree */
        else if γ̄ ≥ h(x̄) + ε then
            add (20) to R;
            T ← T ∪ {o'};                        /* put node o' back in the tree */
        if x̄ ∈ {0,1}^n and γ̄ < h(x̄) + ε then
            UB ← γ̄;                                  /* define a new upper bound */
            x* ← x̄;                                    /* save current incumbent */
return x*
```

4.1.2 Computational results

We present next results of `lp/nlp` for solving (SKP) with Gaussian weights and a fixed capacity. As previously mentioned, more general distributions could be handled as long as numerical integration packages are available. Recall that when each a_i is a Gaussian $\mathcal{N}(\mu_i, \sigma_i)$ and a_0 is a constant, (SKP) can be rewritten as

$$\max_{x \in B} \sum_{i \in N} p_i x_i - K \left\{ \sigma(x) f\left(\frac{a_0 - \mu(x)}{\sigma(x)}\right) + (\mu(x) - a_0) G\left(\frac{a_0 - \mu(x)}{\sigma(x)}\right) \right\}, \qquad (21)$$

where $\mu(x) = \sum_{i=1}^{n} \mu_i x_i$, $\sigma^2(x) = \sum_{i=1}^{n} \sigma_i^2 x_i^2$, f is the density function of $\mathcal{N}(0,1)$, $G = 1 - \Phi$ and Φ is the distribution function of $\mathcal{N}(0,1)$. Note that function G is read from a table. Algorithm 1 is implemented within CPLEX 11 [14], with $\epsilon = 0.1$. Since the model does not contain explicitly all constraints, we must deactivate the dual

presolve, setting *BooleanParam.PreInd* to false. Then, we implemented our (global) cut generation with a *LazyConstraintCallback*, preventing CPLEX from using the dynamic search. The algorithm has been coded in JAVA on a HP Compaq 6510b with a processor Intel Core 2 Duo of 2.40 GHz and 2 GB of RAM memory. We fix a time limit of 100 seconds per instance and the solution time has been set to 100 seconds for instances who could not be solved within this time limit or who exceeded the available memory.

We generated randomly different sets of instances, inspired by the instances from Martello *et al.* [23]. We consider two data ranges: $R = 1000$ and $R = 10000$. Then, parameters μ_i and p_i for each item $i \in \mathsf{N}$ are integers uniformly generated between 4 and R. Each variance σ_i is an integer uniformly generated between 1 and $\lfloor \mu/4 \rfloor$ for each item $i \in \mathsf{N}$, so that negative outcomes are negligible as explained next. We generated 100 instances for each value of parameters K, n and R, with a capacity $a_0 = (h/101) \sum \mu_i$ for instance number h; all results take the average over the groups of 100 instances.

Note that Gaussian variables can take negative values which do not make sense in real applications. Nevertheless, when the ratio σ/μ is small enough, the probability that this happens is negligible so that it does not affect sensibly the objective. For instance, the probability that $\mathcal{N}(\mu, \sigma^2)$ takes a value less than $\mu - 4\sigma$ is slightly less than 0.0001. Therefore we will assume $\mu/4\sigma \geq 1$ when generating our instances.

The following results show that the the penalty factor has little impact on the solution times, while a larger number of items makes the problem more difficult to solve. We study more specifically the time spent at the root node of Algorithm 1. The tables below report the total time in seconds (Time), the fraction of time spent at the root node (Initialization), the number of cuts generated at the root node and the number of cuts added deeper in the tree. We can see in Table 1 that the penalty factor K has little

K	Time		Initialization		InitCuts		AddCuts	
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
2	0.346	0.332	65%	65%	8.33	8.26	0.69	0.73
4	0.369	0.357	64%	64%	8.83	8.73	0.97	0.85
6	0.358	0.357	64%	65%	8.98	8.98	0.98	0.78
8	0.362	0.36	65%	66%	9.09	9.06	0.79	0.78
10	0.343	0.341	64%	63%	9.15	9.18	0.8	0.91
12	0.341	0.341	63%	62%	9.23	9.19	0.98	1.0
14	0.341	0.379	65%	65%	9.33	9.28	0.79	0.84
16	0.352	0.363	67%	66%	9.37	9.1	0.7	0.78
18	0.381	0.361	66%	64%	9.46	9.14	0.76	0.86
20	0.386	0.369	65%	66%	9.71	9.3	0.88	0.81

Table 1 Uncorrelated Instances, $n = 500$.

influence on `lp/nlp`. Therefore, we fix $K = 10$ in the other tests. Results from Table 2 show that we can easily solve problems up to 5000 variables, even though times are significantly larger than in the deterministic case. For example, uncorrelated instances with 5000 items are solved by [23] on average in 0.01 seconds, whereas we need on average 19 seconds to solve such problems. Note that an important fraction of the time is spent in the generation of the cut pool at the root node, because most instances need to explore less than 100 nodes in their branch-and-cut trees. Pursuing our comparison with the deterministic knapsack, we wondered whether strongly correlated and Avis

n	Time		Initialization		InitCuts		AddCuts	
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
200	0.072	0.073	54%	49%	9.04	8.77	1.09	1.28
400	0.209	0.193	65%	62%	9.08	8.75	0.98	0.95
600	0.381	0.353	70%	72%	8.85	9.03	0.94	0.59
800	0.616	0.579	75%	77%	8.96	8.97	0.66	0.49
1000	0.896	0.876	78%	78%	8.73	8.92	0.61	0.61
2000	3.447	3.361	83%	84%	8.81	9.2	0.5	0.42
3000	7.329	7.5	86%	87%	8.85	9.22	0.34	0.28
4000	13.302	12.682	86%	87%	8.93	8.9	0.42	0.25
5000	20.405	17.462	86%	86%	9.02	7.8	0.4	0.28

Table 2 Uncorrelated Instances, $K = 10$.

instances [23] are harder to solve than uncorrelated ones. Strongly correlated instances are characterized by the relations $p_i = \mu_i + R/10$, $i \in \mathsf{N}$, while Avis instances are defined as follows: p_i is an integer uniformly generated between 1 and 1000, $\mu_i = n(n+1)+i$, and $a_0 = n(n+1)\lfloor(n-1)/2\rfloor + n(n-1)/2$. Results from Table 3 show that strongly correlated instances are roughly of the same difficulty as the uncorrelated ones, whereas strongly correlated ones are harder in the deterministic case. Avis instances are significantly harder to solve, see Table 4. Column (Unsolved) reports the number of unsolved instance within 100 seconds. While other problems were essentially solved a the root node, solving even small Avis required to spend a large amount of time exploring branch-and-cut trees. In fact, unreported results show that thousands of nodes where required to solve Avis instances, while uncorrelated and strongly correlated instances were usually solved by exploring less than a hundred of nodes.

n	Time		Initialization		InitCuts		AddCuts	
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
200	0.122	0.149	29%	26%	8.46	8.66	2.27	2.22
400	0.353	0.374	33%	31%	8.64	8.6	1.86	1.94
600	0.927	0.718	25%	33%	8.24	8.42	1.81	1.51
800	0.842	1.064	47%	36%	8.23	8.16	1.17	1.33
1000	2.09	1.006	30%	57%	8.05	8.16	0.89	0.99
2000	3.155	2.727	76%	78%	7.91	7.85	0.59	0.4

Table 3 Strongly Correlated Instances, $K = 10$.

n	Time	Unsolved	Initialization	InitCuts	AddCuts
200	6.639	0	0.23%	11.34	1.79
400	29.379	6	0.05%	12.55	1.52

Table 4 Avis Instances, $K = 10$.

4.2 Pseudo-polynomial cases

In Section 3 we proved that special cases of (SKP) can be solved in pseudo-polynomial time. Since we provide constructive proofs, a natural question is to find out whether the pseudo-polynomial algorithms proposed in the proofs perform better than `lp/nlp`. Although we present results for (SKP) only, the algorithms described next could be used within decomposition schemes for more general problems of type (SCAP). For instance, a classical method to solve the generalized assignment problem consists in relaxing the demand constraints in a Lagrangian fashion, obtaining m independent knapsack problems [9, 26]. Similarly, one can relax constraints (5) in (SEGAP), yielding m independent problems (SKP) that can be solved through dynamic programming whenever weights and capacity are distributed according to one of the assumptions from Section 3.

In all the following results, we fix a time limit of 100 seconds per instance and the solution time has been set to 100 seconds for instances which could not be solved within this time limit or which exceeded the available memory.

4.2.1 Fixed weights and uniformly distributed capacity

Lemma 2 shows how to obtain the solution to problem (7) from the solutions to two knapsack problems and to the problem of maximizing the concave and quadratic function of binary variables (11). The computational results from Martello *et al.* [23] show that the two knapsack problems can be solved in a very short amount of time, and we provide below results of `lp/nlp` applied to (11). Notice that other methods than `lp/nlp` can be used to solve (11). As mentioned in the proof of Lemma 5, (11) can be identified to a half-product that can be solved in pseudo-polynomial time. However, the algorithm from Badics and Boros performed much worse than `lp/nlp`, because very large numbers of states needed to be enumerated. Since the algorithm could hardly solve small instances with 200 variables, we do not report these computational experiments. Commercial MIP solvers are also able to handle (11). We provide below a numerical comparison of CPLEX 11 and `lp/nlp` described in Section 4.1. The instances are gen-

K	`lp/nlp` Time		`cplex` Time		`cplex` Unsolved		Time ratios	
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
2	0.019	0.02	0.019	0.038	0	1	0.7	0.91
4	0.018	0.02	0.036	0.943	0	0	0.78	0.82
6	0.02	0.021	0.246	5.718	0	0	0.77	1.27
8	0.019	0.023	1.311	9.913	0	1	1.05	0.59
10	0.02	0.022	2.598	11.894	0	1	1.15	0.91
12	0.02	0.022	3.233	14.764	0	1	1.11	0.94
14	0.021	0.022	3.299	17.98	0	0	1.09	3.93
16	0.021	0.023	3.329	21.403	1	0	1.29	6.15
18	0.022	0.025	4.662	25.744	0	2	1.14	1.19
20	0.021	0.024	5.256	28.096	0	2	1.23	1.39

Table 5 Comparison between `lp/nlp` and `cplex` when K increases, $n = 100$.

erated as follows. The parameters a_i and p_i for each item $i \in \{1, \ldots, n\}$ are integers

uniformly generated between 1 and R. For each data range R, each value of the penalty factor K and number of items n, we generate 100 instances, with $\mathbb{E}_a(a_0) = (h/101)\sum a_i$ for instance number h. Capacity a_0 varies uniformly between 90% of $\mathbb{E}_a(a_0)$ and 110% of $\mathbb{E}_a(a_0)$. We report on Tables 5 and 6 the average solution time in seconds and the number of unsolved instances for `cplex` only because `lp/nlp` could solve all of them within the time limit. Ratios, given by ($Time$ `cplex`) / ($Time$ `lp/nlp`), are computed for each instance separately; we report the geometric average of the ratios, whereas we report the arithmetic average of solution times. We compute geometric average for ratios because of the following observation : if the ratio for one instance is 1/2 and the one for another instance is 2, their "average" should be equal to one, which is the case using the geometric average.

From Table 5, we see that the `lp/nlp` performance does not depend on the value of K, and that instances for the two range values are of the same difficulty. However `cplex` requires more time to solve instances with $R = 10^4$ than those with $R = 10^3$, which becomes even more significant when K increases. Even tough `cplex` takes on average more time than `lp/nlp`, the ratios close to one tell us that some instances are still solved faster by `cplex` than by `lp/nlp`. Table 6 studies the impact of increasing the number n

n	`lp/nlp` Time		`cplex` Time		`cplex` Unsolved		Time ratios	
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
200	0.029	0.032	0.08	0.269	0	0	1.93	1.77
400	0.052	0.057	0.484	0.239	0	0	5.33	4.14
600	0.076	0.081	1.878	0.638	1	0	9.51	7.75
800	0.097	0.109	2.083	1.447	0	0	14.83	13.31
1000	0.12	0.129	3.814	2.607	0	0	22.19	20.26
1200	0.144	0.166	4.176	4.064	2	1	28.52	28.29
1400	0.174	0.19	5.797	5.727	1	0	31.88	30.33
1600	0.203	0.222	6.982	7.984	1	0	34.8	36.09
1800	0.225	0.249	10.708	11.142	0	0	43.05	44.94
2000	0.264	0.281	12.313	14.309	2	0	48.28	50.86

Table 6 Comparison between `lp/nlp` and `cplex` when n increases, $K = 2$.

of items, hence variables in the model. It is clear from the values of the ratios that the `lp/nlp` handles better bigger instances than `cplex` does, the ratio average increases more or less linearly with the number of items. This may be explained by the following observation: `cplex` deals with $O(n^2)$ variables so that its solution time is very impacted by n. On the other hand, the number of variables in `lp/nlp` only increases linearly with n, because this algorithm deals implicitly with the non linear objective. Then, because K is small enough, solution times required by `cplex` to solve instances with $R = 10^3$ are similar to those required to solve instances with $R = 10^4$. We stopped our test to 2000 variables because `cplex` required almost all the available memory to solve these instances.

4.2.2 Subset sum with gaussian weights

Our last group of tests studies subset sum instances with Gaussian weights, which satisfy the assumptions of Theorem 2. Therefore, μ_i are (integer) uniformly distributed

between 1 and R and $\sigma_i = \sqrt{\lambda \mu_i}$, where λ must be chosen between 0 and 1. We have two algorithms at our disposal to solve these problems. As for general knapsack problem with Gaussian weights, we can use `lp/nlp` described in Section 4.1.1. Alternatively, using Theorem 2, we can solve these problems by the algorithm described in Algorithm 2. The later essentially solves two subset-sum problems using the method `decomp` from Pisinger [27], available at `www.diku.dk/hjemmesider/ansatte/pisinger/`. Algorithm 2 has been coded in C on the same computer as the one used for `lp/nlp`. Table 7 studies

Algorithm 2: `stoch-subsum`

1 compute $\lfloor z^* \rfloor$ and $\lceil z^* \rceil$ using a dichotomic search;
2 solve (16) and (17) with `decomp` from [27], yielding solutions x_1^* and x_2^*;
3 **if** $Z_{\mathcal{R}}(x_1^*) > Z_{\mathcal{R}}(x_2^*)$ **then** $x^* := x_1^*$ **else** $x^* := x_2^*$;
return x^*

the sensibility to parameter λ for `lp/nlp`, results for `stoch-subsum` are not reported because all problems are solved within 0.001 seconds. As expected, Theorem 2 enables us to solve subset sum problems orders of magnitude faster than using `lp/nlp`, which is even more striking in Table 8 below.

λ	Time		Initialization		InitCuts		AddCuts	
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
1/64	1.783	1.979	7%	5%	11.14	7.03	2.11	1.42
1/16	1.625	1.852	8%	3%	10.9	11.19	1.58	1.63
1/4	1.609	1.891	6%	3%	10.42	10.97	1.45	1.96

Table 7 Subset Sum with $n = 500$ and $K = 10$.

To respect the condition $\mu/4\sigma \geq 1$, λ must satisfy $\lambda \leq \mu/16$. This becomes $\lambda \leq 1/16$ since μ is comprised between 1 and R. Therefore, we fix $\lambda = 1/16$ in our subsequent computational results. Table 8 compares `lp/nlp` and `stoch-subsum` for different values of n. Whenever `stoch-subsum` was able to solve an instance in less than 0.001 seconds, its solution time was set to 0.001 seconds. Ratios, given by ($Time$ `lp/nlp`) / ($Time$ `stoch-subsum`), are computed for each instance separately; we report the geometric average of the ratios, whereas we report the arithmetic average of solution times. Comparing Table 8 with Tables 2 and 3, we see that stochastic subset sum problems are significantly harder to solve than uncorrelated and correlated instances. This is due to the size of the branch-and-cut trees, hundreds of nodes being explored for the subset sum instances.

Similarly to the Avis knapsack instances, we can define Avis subset sum instances as follows: $p_i = \mu_i = n(n+1) + i$, and $a_0 = n(n+1)\lfloor (n-1)/2 \rfloor + n(n-1)/2$. To obtain groups of 100 "different" instances, we shuffle the order in which the items are read. Avis subset sum are extremely hard to solve already in the deterministic case, specialized algorithms are needed to solve large instances. Table 9 compares `lp/nlp` and `stoch-subsum` on these difficult instances. In fact, `lp/nlp` can not solve problem with more than 20 variables within the time limit of 100 seconds. This is due to the very large number of nodes explored. For $n = 10$, `lp/nlp` explores around 751 nodes on average, while exactly 705430 nodes are explored for each instance with $n = 20$.

n	lp/nlp				stoch-subsum		Time ratios	
	Time		Unsolved		Time			
	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$	$R = 10^3$	$R = 10^4$
200	0.199	0.244	0	0	0.001	0.001	170	198
400	0.917	0.991	0	0	0.001	0.001	636	736
600	2.542	2.726	0	0	0.002	0.002	1331	1364
800	5.306	5.568	0	0	0.002	0.002	2190	2096
1000	10.12	9.716	0	0	0.003	0.003	3422	3085
2000	84.768	79.023	31	26	0.005	0.005	12660	11475

Table 8 Comparison of `lp/nlp` and `stoch-subsum` on subset sum instances with $K = 10$ and $\lambda = 1/16$.

Although `stoch-subsum` requires more time than for other subset sum problems, it can still handle problems containing up to 600 variables within 100 seconds of CPU time.

n	lp/nlp	stoch-subsum	Time ratios
10	0.055	0.001	51.8
20	40.124	0.001	40119
100	–	0.055	–
200	–	1.003	–
300	–	5.22	–
400	–	16.6	–
500	–	41.6	–
600	–	88.1	–

Table 9 Comparison of `lp/nlp` and `stoch-subsum` on Avis subset sum instances with $K = 10$ and $\lambda = 1/16$.

5 Conclusion

In this paper, we study two important aspects of the stochastic knapsack problem with penalty recourse. We provide complexity results for three special cases of the problem, and propose a branch-and-cut algorithm for the general problem. The algorithm is tested on a large test of instances randomly generated and inspired by the instances used in [23] for the deterministic knapsack problem. We can solve in less than a minute uncorrelated and correlated problems involving up to 5000 variables. Only a few linearizations are required and less than 100 nodes are explored in the branch-and-cut tree. Similarly to the deterministic situation, Avis instances are very hard to solve, requiring the exploration of much larger trees.

The branch-and-cut algorithm is less efficient with subset sum problems, usually exploring up to a thousand of nodes. These subset sum problems can however be solved almost as fast as the deterministic version of the problem, by using a specialized algorithm from Pisinger [27]. The pseudo-polynomial algorithm devised for the knapsack problem with constant weights and capacity uniformly distributed is also tested, but does not provide positive results.

An interesting subject for future research would be to study how to extend efficiently the methods described herein to handle more general problems. We should

evaluate numerically our branch-and-cut algorithm `lp/nlp` for more general (SCAP), and test decomposition algorithms that reduce general problems to a sequence of knapsack problems.

References

1. K. Abhishek, S. Leyffer, and J. T. Linderoth. Filmint: An outer-approximation-based solver for nonlinear mixed integer programs. *INFORMS Journal on computing*. In press.
2. T. Badics and E. Boros. Minimization of half-products. *Math. Oper. Res.*, 23(3):649–660, 1998.
3. C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2):318–326, 2000.
4. P. Beraldi and M. E. Bruni. An exact approach for solving integer problem under probabilistic constraints with random technology matrix. *Annals of Operations Research*, 177:127–137, 2010.
5. J. R. Birge and F. V. Louveaux. *Introduction to Stochastic programming (2nd edition)*. Springer Verlag, New-York, 2011.
6. P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.
7. P. Bonami, M. Kilinc, and J. Linderoth. *IMA Volumes*. University of Minnesota, 2010. Algorithms and Software for Convex Mixed Integer Nonlinear Programs.
8. G.G. Brown and G.W. Graves. Real-time dispatch of petroleum tank trucks. *Management Science*, 27:19–32, 1981.
9. D. G. Cattrysse and L. N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260 – 272, 1992.
10. A. Cohn and C. Barnhart. The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning. In *Proceedings of the Triennial Symposium on Transportation Analysis (TRISTAN III)*, 1998.
11. Y.M. Ermoliev, T.Y. Ermolieva, G.J. MacDonald, and V.I. Norkin. Stochastic optimization of insurance portfolios for managing exposure to catastrophic risks. *Annals of Operations Research*, 99:207–225, 2000.
12. W.K. Klein Haneveld, L. Stougie, and M.H. van der Vlerk. Stochastic integer programming with simple recourse. Technical Report Research Memorandum 455, University of Groningen, 1991.
13. B. Hansotia. Some special cases of stochastic programs with recourse. *Operations Research*, 25(2):361–363, 1977.
14. ILOG CPLEX Division, Gentilly, France. *ILOG. ILOG CPLEX 11.0 Reference Manual.*, 2007.
15. A. J. Kleywegt and J. D. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Oper. Res.*, 49(1):26–41, 2001.
16. A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.*, 12(2):479–502, 2002.
17. O. Klopfenstein. Solving chance-constrained combinatorial problems to optimality. *Computational Optimization and Applications*, In press.
18. O. Klopfenstein and D. Nace. A note on polyhedral aspects of a robust knapsack problem. Optimization Online, 2007.
19. O. Klopfenstein and D. Nace. A robust approach to the chance-constrained knapsack problem. *Oper. Res. Lett.*, 36(5):628–632, 2008.
20. S. Kosuch and A. Lisser. Upper bounds for the 0-1 stochastic knapsack problem and a b&b algorithm. *Ann. Oper. Res.*, 2009. Article in press.
21. G. Laporte, F. V. Louveaux, and H. Mercure. The vehicle routing problem with stochastic travel times. *Transp. Science*, 26(3):161–170, 1992.
22. F. V. Louveaux and M. H. van der Vlerk. Stochastic programming with simple integer recourse. *Mathematical Programming*, 61:301–325, 1993.
23. S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.
24. D. P. Morton and R. K. Wood. *Advances in computational and stochastic optimization, logic programming and Heuristic Search*, chapter 5, pages 149–168. Woodruff, 1998.

25. J. M. Mulvey, R. J. Vanderbei, and A. Z. Stavros. Robust optimization of large-scale systems. *Oper. Res.*, 43(2):264–281, 1995.
26. R. M. Nauss. Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS J. on Computing*, 15(3):249–266, 2003.
27. D. Pisinger. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, 114(3):528 – 541, 1999.
28. A. Prékopa. *Stochastic Programming*. Kluwer, 1995.
29. I. Quesada and I. E. Grossman. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Comput. Chem. Eng.*, 16(10/11):937–947, 1992.
30. D. Spoerl and R.K Wood. A stochastic generalized assignment problem. INFORMS Annual Meeting, Atlanta, GA, 1922 October, 2003.
31. R.J.B. Wets. Solving stochastic programss with simple recourse. *Stochastics*, 10(3):219–242, 1983.