KNAPSACK PROBLEM WITH PROBABILITY CONSTRAINTS

GAIVORONSKI A A / LISSER A / LOPEZ R

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

07/2008

Rapport de Recherche N° 1498

Knapsack problem with probability constraints

Alexei A. Gaivoronski[*], Abdel Lisser[†] and Rafael Lopez[†]

July 11, 2008

Abstract

This paper is dedicated to a study of different extensions of the classical knapsack problem to the case when different elements of the problem formulation are subject to a degree of uncertainty described by random variables. This brings the knapsack problem into the realm of stochastic programming. Two different model formulations are proposed, based on the introduction of probability constraints. The first one is a static quadratic knapsack with a probability constraint on the capacity of the knapsack. The second one is a two-stage quadratic knapsack model, with recourse, where we introduce a probability constraint on the capacity of the knapsack in the second stage. As far as we know, this is the first time such a constraint has been used in a two-stage model. The solution techniques are based on the semidefinite relaxations. This allows for solving large instances, for which exact methods cannot be used. Numerical experiments on a set of randomly generated instances are discussed below.

Introduction

The knapsack problem (KP) is a well-known and well-studied problem in combinatorial optimization. Knapsack problems are often used to model industrial situations or financial decisions. They may also appear as sub-problems of larger or more complex problems. The most famous form of KP is the single constraint binary version: we are given N items, with a profit p_i for the item i, and a weight w_i for the item i, with $i = 1, ..., N$, and a knapsack capacity c. The problem is to select a subset of items so that the weight of the subset does not exceed c, and returns a maximal total profit. In this form, the problem is known to be NP-Hard [GJ79], and has been intensively studied in the past decades, and we now know several exact and approximate algorithms for this problem. In particular, it admits a FPTAS [IK75]. For the quadratic knapsack problem, a survey done by David Pisinger [Pis07] gives detailed information on the problem and a number of results on the performance of various relaxations and algorithms used to solve or approximate the problem.

In the case of modeling financial decisions, transportation, or production plans, however, this formulation shows its limitations, since it does not take into account uncertainty on the problem parameters, such as the prices p_i or the weights w_j. Similarly, such decisions are not static, and this model cannot take into account new information available on the prices or the weights. There have been several studies done on the stochastic knapsack problem in the past years: work has been done to find heuristics [CB98], approximation algorithms [KP98], [DGV04], [SS06].

[*]Department of Industrial Economy and Technology Management, Norwegian University of Science and Technology, Trondheim, Norway.

[†]Laboratoire de Recherche en Informatique, Université de Paris Sud, 91405, Orsay Cedex, France.

Stochastic knapsack problems can reach a number of binary variables and constraints of such magnitude that commercial packages cannot find a solution in a reasonable time or memory space, requiring the use of linear relaxations to find an upper bound on the problem. While linear relaxations were successful for many combinatorial optimization problems, it turns out that knapsack problems, especially their quadratic formulation, can not be approximated tightly by linearization based methods. Stronger relaxation methods, namely semidefinite relaxations (called SDP thereafter), have turned out to be particularly interesting for such combinatorial optimization problems [GW95], [HR98]. See also [Pis07] for a survey on the quadratic knapsack problem.

More precisely, semidefinite programming is a recent development of convex optimization, which deals with optimization problems over symmetric positive semidefinite matrices with linear cost function and linear constraints. Groetschel and al. showed that semidefinite optimization problems can be solved in polynomial time [GLS88].

In this paper we present two variants of stochastic optimization problems: the first one is a static quadratic knapsack problem with probability constraint. The probability constraint is used to model the risk we are willing to take when making our decision. We only know some information about the weights of the items, but we have to take a decision with this limited knowledge at the risk of breaking the capacity constraint. The second model is the stochastic quadratic knapsack problem with recourse. In this model, we have a two-stage formulation which models a situation where we make a decision with limited information, but where a second decision (the recourse) can be made afterwards, after receiving information about the weight, or prices. This allows to modify initial decisions. In this model, since at the second step, some information is still unknown, we face a risk of breaking the capacity constraint when making the decision.

We can view the static stochastic knapsack as a resource allocation problem. A planner has to choose among various products, various foods for humanitarian relief for instance, and has to choose among them a subset to fill crates for maximal utility, with the constraint that the crates can only carry a finite weight of food. However, he does not have all the information to select the best products since the food has not yet been collected. With this limited information, a decision can still be made where we accept a certain low risk, or probability, of selecting more products than can fit in the crate. The case with recourse would then be making a decision with part of the supplies known, and making changes in the food that will be packed once more information about the rest of the food has been collected. However, there is a cost to move the food in or out of the crate at the last moment. Again, since some of the information is not known, we also face a small risk of planning to try to fit too many products into the crate.

This paper is organized as follows: first, we present the static knapsack problem with probability constraint wherein we give a presentation of the problem. Then we reformulate this problem under the form of an SDP problem and detail two different relaxations. We then present the stochastic knapsack problem with recourse. Finally, numerical experiments are given in a third section.

1 Static quadratic knapsack problem with probability constraint

In this section, we present the static knapsack problem with a probability constraint. We formulate the uncertainty over the weights w_i with random variables, and we consider a probability $(1-\alpha)$ of satisfying the constraints, where α is the risk measurement. We then present the SDP relaxation, and the two relaxations we used to solve our problem.

1.1 Problem formulation

The objective is to maximize the value of the items contained in the knapsack. Objects pairs are assigned a value c_{ij}, reflecting the synergy obtained by taking two complementary items. For example, a hiker may carry canned food, but it has little use if he does not carry a tool e.g. can opener. Objects may have their own value c_{ii}, for example a bottle of water in itself is valuable to a hiker, without the need for extra flavoring or tools. To show that an item is taken, we use a binary variable x_i set to 1 if item i is taken, 0 otherwise. The knapsack, however, does not have an unlimited capacity, and we use the constraint that the sum of the weights w_i for all items taken is lower than the capacity of the bag. In our case, there is uncertainty on the weight of the items. Instead of requiring the weight of the items to be lower than the capacity, we want the probability of the validity of the constraint to be greater than a certain percentage $(1 - \alpha)$.

The quadratic knapsack problem with probability constraints can then be formulated as follows:

$$\max_{x} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_i x_j \tag{1}$$

$$\mathbb{P} \left\{ \sum_{i=1}^{N} w_i x_i \leq d \right\} \geq 1 - \alpha \tag{2}$$

where $x_i, i = 1 : n$ are binary variables and $w_i, i = 1 : n$ are random variables with joint probability distribution H. Let us consider the case where H is concentrated in the finite number of points $w^k = \left(w_1^k, ..., w_n^k\right), k = 1 : K$ with probabilities p_k such that

$$\sum_{k=1}^{K} p_k = 1, \quad p_k \geq 0$$

Then the problem (1)-(2) can be reformulated as follows:

$$\max_{x,\Lambda} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_i x_j \tag{3}$$

$$\sum_{i=1}^{N} w_i^k x_i \leq d, \; k \in \Lambda \tag{4}$$

$$\sum_{k \in \Lambda} p_k \geq 1 - \alpha \tag{5}$$

Constraint (5) means that we have to choose a subset Λ of scenarios such that the sum of the probabilities of this subset is greater than $(1 - \alpha)$. For this subset, the capacity constraints will be active and valid, whereas for the scenarios not in this subset, the capacity constraint is not activated.

This problem can be reformulated as binary optimization problem by introducing the auxiliary binary variable y_k for each observation $k = 1 : K$ as follows:

$$y_k = \begin{cases} 0 & \text{if } \; k \in \Lambda \\ 1 & \text{otherwise} \end{cases}$$

This yields the following problem:

$$\max_{x,y} \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_i x_j \tag{6}$$

$$\sum_{i=1}^{N} w_i^k x_i \leq d + My_k, \;\; k = 1, \dots, K \tag{7}$$

$$\sum_{k=1}^{K} p_k y_k \leq \alpha \tag{8}$$

where M is an arbitrary number such that

$$M \geq \max_k \sum_{i=1}^{N} w_i^k - d$$

This problem is a quadratic optimization problem with binary variables. The quadratic knapsack problem is NP-hard, and so is its stochastic formulation. In this case, we seek upper bounds using strong relaxations, namely SDP relaxations.

In the following we detail SDP relaxations we use for our experiments: first we present the notations we use as well as the LP relaxation. Then, we sketch a first SDP relaxation. We also present a second, tighter SDP relaxation we used.

1.2 Linear relaxation

Continuous relaxation of (6)-(8) can be derived using standard linear relaxation introduced by Fortet in [For59]:

$$\max_{x,y} \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}X_{ij}$$

$$\sum_{i=1}^{N} w_i^k x_i \leq d + My_k, \;\; k = 1, \dots, K$$

$$\sum_{k=1}^{K} p_k y_k \leq \alpha$$

$$M \geq \max_k \sum_{i=1}^{N} w_i^k - d$$

$$\begin{array}{lr}
X_{ij} \leq x_i, & i < j = 1, \dots, N \\
X_{ij} \leq x_j, & i < j = 1, \dots, N \\
X_{ij} \geq x_i + x_j - 1, & i < j = 1, \dots, N \\
0 \leq X_{ij} \leq 1, & i, j = 1, \dots, N \\
0 \leq x_i \leq 1, & i = 1, \dots, N
\end{array}$$

where $X_{ij} = x_i x_j$. During the numerical experiments, we omitted constraints $X_{ij} \geq x_i + x_j - 1$, since all of our c_{ij} are strictly positive and we have a maximization problem, these constraints would necessarily be verified.

This linear relaxation will be used in our numerical experiments as a basis for comparison when an exact solution cannot be found. While it is a fast way to have an upper bound, it also is a generally very weak bound. In order to obtain tighter bounds, we use an alternative relaxation based on semidefinite programming.

1.3 SDP relaxations

Our quadratic knapsack problem with a probability constraint uses an important number of binary variables which makes solving large problems using exact methods impractical. Two issues can come up; in particular, the CPU time necessary to solve large instances may be too long. For example, one needs a decision made quickly, while the exact result could take days or longer (months, or even years) to be computed. The second issue is that methods like branch and bound rely on search trees and search nodes that need to be stored in memory. In certain cases, data may reach the limit the system can handle, and therefore cannot be computed. When we cannot compute exact solutions, we calculate an upper bound of the optimum using relaxations. Semidefinite programming is interesting here because it requires memory and time polynomial with the input [GLS88]. This makes it a suitable tool, which is reinforced by the fact that it gives good bounds for quadratic binary problems [HR98], and for the quadratic knapsack problem [Pis07].

For the sake of simplicity for SDP relaxations, problem (6)-(8) can be rewritten as:

$$\max_{z} z^t \hat{C} z \tag{9}$$

$$\sum_{i=1}^{m} g_i^k z_i \leq d,\ k = 1, \dots, K \tag{10}$$

$$\sum_{k=1}^{m} q_k z_k \leq \alpha \tag{11}$$

where

- $m = N + K$,
- $\hat{C}$ is a $m \times m$ matrix, with each column r_i defined by $r_i^t = (c_{i1}, \dots, c_{in}, 0, \dots, 0)$,
- z is a $m-$vector defined by $z^t = (x_1, \dots, x_N, y_1, \dots, y_K)$,
- g_k is a $m-$vector defined by $g_k^t = (w_1^k, \dots, w_N^k, 0, \dots, M, \dots, 0)$. Constant M is located at index $N + k$,
- q is a $m-$vector defined by $q^t = (0, \dots, 0, p_1, \dots, p_K)$.

Using the notation above, we can easily build a SDP relaxation of (6)-(8): let X be the matrix $\begin{bmatrix} zz^T & z \\ z^T & 1 \end{bmatrix}$. We will use a modified matrix C defined as $C = \begin{bmatrix} \hat{C} & 0 \\ 0 & 0 \end{bmatrix}$. Let W_k and P be the appropriate matrices constructed from g_k and q. The SDP relaxation can then be written as:

$$(SDP1)\left\{\begin{array}{ll} Min & Trace(C\bullet X)\\ s.c & \\ & Trace(W_k\bullet X)\leq d \quad k=1,\ldots,K\\ & Trace(P\bullet X)\leq \alpha\\ & diag(X)=z\\ & X\succeq 0 \end{array}\right. \tag{12}$$

However, this relaxation is known to be weak Rendl and Helmberg show in [HRW00] that it is the case for the quadratic knapsack problem, and Rendl and Sotirov found the same results in [RS03] for the quadratic assignment problem. A tightening of the constraints is required in order to achieve better results. A tighter relaxation is required to avoid having binary variables being too far from either 0 or 1, which causes this formulation to give poor bounds.

In order to tighten the relaxation presented above, we add valid inequalities to the problem that limit the space of solutions. One method to generate valid inequalities is to multiply a constraint by the binary variables it contains. The resulting inequalities can be added to the problem and will help tighten the relaxation. Sherali and Adams proposed in [SA90]and [SA94] such a relaxation scheme, giving better results. In fact, it is possible to build a hierarchy of relaxations (see [SA90], [SA94] and [Lau03] for more details) leading to the integer polytope. However, building each relaxation when the number of constraints and variables increases becomes expensive, so we chose to restrict ourselves to the first relaxation in the hierarchy, RLT-1. In our case, we can multiply constraints (10) by x_i and $(1-x_i)$ for all $i=1,\ldots,N$ respectively and constraints (11) by y_j and $(1-y_j)$ for all $j=1,\ldots,K$. These constraints will be referred to as *Sherali-Adams constraints* hereafter.We also added the quadric polytope constraints $X_{ij}\leq z_i$ and $X_{ij}\leq z_j$ (see [Pad89] and [HPRW95] for more details) to further strengthen the relaxation.

We obtain the following SDP relaxation:

$$(SDP2)\left\{\begin{array}{lll} Min & Trace(C\bullet X) & \\ s.c & & \\ & Trace(\check{W}_{ki}\bullet X)\leq 0 & k=1,\ldots,K, i=1,\ldots,N\\ & Trace(\breve{W}_{ki}\bullet X)\leq d & k=1,\ldots,K, i=1,\ldots,N\\ & Trace(\tilde{W}_{kj}\bullet X)\leq 0 & k=1,\ldots,K, j=1,\ldots,K\\ & Trace(\tilde{\tilde{W}}_{kj}\bullet X)\leq d & k=1,\ldots,K, j=1,\ldots,K\\ & Trace(\check{Q}_j\bullet X)\leq 0 & j=1,\ldots,K\\ & Trace(\breve{Q}_j\bullet X)\leq \alpha & j=1,\ldots,K\\ & Trace(\tilde{Q}_i\bullet X)\leq 0 & i=1,\ldots,N\\ & Trace(\tilde{\tilde{Q}}_i\bullet X)\leq \alpha & i=1,\ldots,N\\ & Trace(T_{ij}\bullet X)\leq 0 & (i,j)=[1,N]^2,(i,j)=[N+1,N+K]^2\\ & Trace(\hat{T}_{ij}\bullet X)\leq 0 & (i,j)=[1,N]^2,(i,j)=[N+1,N+K]^2\\ & diag(X)=z & \\ & X\succeq 0 & \end{array}\right. \tag{13}$$

Where $\check{W}_{ki}$, $\breve{W}_{ki}$, $\tilde{W}_{kj}$, $\tilde{\tilde{W}}_{kj}$, $\check{Q}_j$, $\breve{Q}_j$, $\tilde{Q}_i$ and $\tilde{\tilde{Q}}_i$ are the matrices constructed from g_k and q modeling the sherali-Adams constraints mentioned above. T_{ij} and $\hat{T}_{ij}$ are the matrices modeling the quadric polytope constraints mentioned above.

This relaxation gives tighter results but comes at the cost of an increased number of constraints. In fact, for each capacity constraint in the original problem we have $2N$ times as many constraints.

Similarly, instead of having one probability constraint, we have K times as many. Since we also use quadric polytope constraints, we add a polynomial number (in N and K) of constraints to the problem.

We now present the second KP extension: the stochastic quadratic knapsack problem with recourse. In the static knapsack problem with a probability constraint, the decision is made once at the beginning and cannot be corrected after observing the realization of the random variables. In the model with recourse, we have a two-stage process, where one decision is made after obtaining (partial) information about the realization of the random variables, and a second decision after to correct it if necessary.

2 Stochastic quadratic knapsack problem with recourse

In this section we extend models developed in the previous section to the case when the decision problem extends to two time periods (two stages). The initial decision is made during the first period before knowing the realizations of the random variables. Then these realizations are (partially) revealed and the second stage decision is made which corrects the first stage decision, taking into account this information. This modeling scheme is known as *stochastic program with recourse*. Moreover, we introduce probability constraints in the second stage. To the best of our knowledge, this formulation has not been studied in the literature. We start by formulating the general quadratic stochastic program with recourse with probability constraint in the second stage. A generic quadratic stochastic problem with recourse can be modeled as follows:

$$\max_x x^T Cx + \mathbb{E}_\omega Q(x, \omega) \tag{14}$$

$$Rx \leq s \tag{15}$$

where (15) models generic linear constraints, with $R \in \mathbb{R}^{m,n}$ and $s \in \mathbb{R}^m$. The second stage value is given by the solution of the problem:

$$Q(x, \omega) = \max_u u^T D(\omega) u \tag{16}$$

$$\mathbb{P}\left\{W(\omega, \psi)u + T(\omega, \psi)x \leq h(\omega, \psi) | \omega\right\} \geq 1 - \alpha \tag{17}$$

In this model, the uncertainty is described by the two probability vectors ω and ψ with a given joint probability distribution. When the initial decision is made, neither ω nor ψ are known. After this decision, part of the information is revealed. This corresponds to the realization of random vector ω. The second stage decision can then be taken, with knowledge of ω and of the first stage decision x, but without the knowledge of the realization of random vector ψ. This corresponds to the probability constraint (17), which gives the conditional probability of being satisfied given the value of vector ω. Unlike in the classical formulation, we have a random vector ψ present in the second stage decision.

2.1 First stage decision

We now need to adapt this generic model to our knapsack problem. Like in the static case, we assume we have n items, and each item is characterized by its value c_{ii}, and weight w_i, $i = 1 : n$. Each item pair is characterized in the same manner by its value c_{ij}, see section 1.1 for more details about item pairs values. The objective is to maximize the value of the items contained in the knapsack, with the constraint that it has a limited capacity d. The selection of an item during the first stage is defined by a binary decision variable x_i which takes value 1 if the item i is included in the selection and 0 otherwise. The formulation of this first stage decision of the problem is the following:

$$\max_x \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_ix_j + \mathbb{E}_\omega Q(x,\omega) \tag{18}$$

$$\sum_{i=1}^{N} w_ix_i \leq d \tag{19}$$

The constraint (19) describes the knapsack capacity constraint. Equation (18) consists of two parts: the value of the knapsack during the first period, and the expected value of the same knapsack during the second period. This expected value depends on the items selected in the first period (the vector x), and the realization of the random vector ω.

2.2 Second stage decision

After the first stage decision is made, the values of items may change, as well as their weight. During the second period, the item i has the value $b_{ii}(\omega)$ and the weight $v_i(\omega,\psi)$. Each item pair (i,j) has the value $b_{ij}(\omega)$. Similarly to the first period, there's a constraint on the capacity of the knapsack, which is subject to change too. We note the new capacity $h(\omega,\psi)$. Similarly to the static case, we want the probability for this constraint to be valid to be greater than $(1-\alpha)$. The realization of vector ω is known before making the second stage decision, whereas only the distribution of ψ is known, conditioned on ω.

The second stage decision allows to change the initial decision in order to correct mistakes which appear after extra information is known. There are two possibilities: first, an item which was selected during the first period can be removed. In this case, we describe this decision with a binary variable u_i^- set to 1 to indicate that item i was removed from the knapsack, 0 otherwise. Likewise, an item that was previously rejected can be selected. In this case, we use a binary variable u_i, set to 1 if we select the item i in the second period, 0 otherwise. Note that if item i was selected during the first period, then if it is not removed during the second, it is considered selected again (i.e. $u_i = 1$) in the second period. When an item i is removed, it causes a cost, which not only includes deducting its value, but may also contain additional penalties, such as time or manipulation costs necessary to reorganize the knapsack. This allows us to formulate the second stage decision as follows:

$$Q(x,\omega) = \max_{u,u^-} \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij}(\omega)u_iu_j - \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij}^-(\omega)u_i^-u_j^- \tag{20}$$

$$u_i \geq x_i - u_i^-,\ i = 1:n \tag{21}$$

$$u_i^- \leq x_i,\ i = 1:n \tag{22}$$

$$\mathbb{P}\left\{\sum_{i=1}^{N} v_i(\omega,\psi)u_i \leq h(\omega,\psi)|\omega\right\} \geq 1-\alpha \tag{23}$$

Equation (20) is our objective: we want to maximize the value of the knapsack to which we must deduce the cost of removing items previously selected. The two constaints (21) and (22) link the first stage and second stage decisions: the first one means that if an item i was selected during the first period and not deselected, then it is necessarily considered selected during the second period. Conversely, the constraint (22) means that only an item that was selected during the first phase can be deselected. Constraint (23) represents the probability $(1-\alpha)$ of the capacity constraint to be valid.

2.3 Additional features

Our formulation (18)-(23) is general enough to allow the modeling of additional characteristics which do not appear explicitly in the problem description. The three main features are:

1. *Different composition of the allowable set of items during the first and the second stage.* Since the weights of the items and the capacity of the knapsack may change (based on ω) between the two periods, it allows us to model cases where some items may not be allowed during either period. For example, to represent the case where an item i is not allowed during the second period, we can set $v_i(\omega,\psi) > h(\omega,\psi)$. Using our introductory example of a decision maker selecting food for humanitarian relief, it would be the case when the manufacturer of the food sends a recall for his product which was found unfit for consumption. Similarly, an item could be absent during the first period and only available during the second period, in which case it would either have $c_i = 0$ or $w_i > d$.

2. *Dependence of the item values on random variables.* For example, the first stage values may depend on ω, $c_{ij} = c_{ij}(\omega)$ and the second stage values and removal costs may depend also on ψ, $b_{ij}(\omega) = b_{ij}(\omega,\psi)$, $b^-_{ij}(\omega) = b^-_{ij}(\omega,\psi)$. In this case, we consider the average of the objective function, and Equation (18) is reformulated into:

$$\max_x \mathbb{E}_\omega \left(\sum_{i=1}^N \sum_{j=1}^N c_{ij}(\omega) x_i x_j + Q(x,\omega) \right) \tag{24}$$

   and (20) is reformulated into

$$Q(x,\omega) = \max_{u,u^-} \mathbb{E}_\psi \left(\sum_{i=1}^N \sum_{j=1}^N b_{ij}(\omega,\psi) u_i u_j - \sum_{i=1}^N \sum_{j=1}^N b^-_{ij}(\omega,\psi) u^-_i u^-_j \;|\omega \right) \tag{25}$$

   Denoting now

$$\bar{c}_{ij} = \mathbb{E}_\omega c_{ij}(\omega),\ \bar{b}_{ij}(\omega) = \mathbb{E}_\psi \left(b_{ij}(\omega,\psi)\ |\omega\right),\ \bar{b}^-_{ij}(\omega) = \mathbb{E}_\psi \left(b^-_{ij}(\omega,\psi)\ |\omega\right)$$

   and by replacing c_{ij} with $\bar{c}_{ij}$ in (18) and $b_{ij}(\omega), b^-_{ij}(\omega)$ by $\bar{b}_{ij}(\omega), \bar{b}^-_{ij}(\omega)$ in (20) we recover the original formulation (18)-(23).

3. *Probabilistic knapsack constraint on the first stage.* This case is similar to the static knapsack formulation. If the first stage weights depend on the random vector ω ($w_i = w_i(\omega)$), then constraint (19) is rewritten as follows:

$$\mathbb{P}\left\{ \sum_{i=1}^N w_i(\omega) x_i \le d \right\} \ge 1-\alpha \tag{26}$$

As we showed, formulation (18)-(23) of the stochastic knapsack problem with recourse is quite general and covers many specific cases. We will now work with this problem and reformulate it in order to be able to use resolution techniques like those used in section 1. The first step of the reformulation is to rewrite the problem into deterministic equivalents. After this, we will be able to use semidefinite relaxations to the deterministic equivalents.

2.4 Deterministic rewriting of the problem

In order to rewrite the stochastic quadratic knapsack with recourse into a deterministic form, we need to consider the case when the joint distribution of the random vectors ω and ψ is concentrated in the finite number of points. We assume that the random vector ω is concentrated in the finite number of points $\omega_k, k = 1 : K$ with probabilities p_k. We will refer to these points as *scenarios.* In this case the problem (18)-(23) can be rewritten as follows:

$$\max_x \sum_{i=1}^N \sum_{j=1}^N c_{ij} x_i x_j + \sum_{k=1}^K p_k Q(x,k) \tag{27}$$

$$\sum_{i=1}^N w_i x_i \leq d \tag{28}$$

with

$$Q(x,k) = \max_{u,u^-} \sum_{i=1}^N \sum_{j=1}^N b_{ijk} u_i u_j - \sum_{i=1}^N \sum_{j=1}^N b^-_{ijk} u^-_i u^-_j \tag{29}$$

$$u_i \geq x_i - u^-_i, \ i = 1 : n \tag{30}$$

$$u^-_i \leq x_i, \ i = 1 : n \tag{31}$$

$$\mathbb{P}\left\{ \sum_{i=1}^N v_{ik}(\psi) u_i \leq h_k(\psi) \mid \omega = \omega_k \right\} \geq 1 - \alpha \tag{32}$$

where

$$Q(x, \omega_k) = Q(x,k), \ b_{ij}(\omega_k) = b_{ijk}, \ b^-_{ij}(\omega_k) = b^-_{ijk}, \ v_i(\omega_k, \psi) = v_{ik}(\psi), \ h(\omega_k, \psi) = h_k(\psi).$$

Substituting (29) into (27) and collecting constraints for each scenario we obtain an equivalent problem to (27)-(32):

$$\max_{x, u_{ik}, u^-_{ik}} \sum_{i=1}^N \sum_{j=1}^N c_{ij} x_i x_j + \sum_{k=1}^K p_k \left(\sum_{i=1}^N \sum_{j=1}^N b_{ijk} u_{ik} u_{jk} - \sum_{i=1}^N \sum_{j=1}^N b^-_{ijk} u^-_{ik} u^-_{jk} \right) \tag{33}$$

$$\sum_{i=1}^N w_i x_i \leq d \tag{34}$$

$$u_{ik} \geq x_i - u^-_{ik}, \ i = 1 : n, \ k = 1 : K \tag{35}$$

$$u^-_{ik} \leq x_i, \ i = 1 : n, \ k = 1 : K \tag{36}$$

$$\mathbb{P}\left\{\sum_{i=1}^{N} v_{ik}(\psi)u_{ik} \le h_k(\psi) \mid \omega = \omega_k\right\} \ge 1-\alpha,\ k = 1:K \tag{37}$$

Probability constraints (37) are reformulated as probability constraint (2) was for the static stochastic knapsack problem. Suppose that the random vector ψ is concentrated in the finite number of points $\psi_{kr}, k = 1:k, r = 1:R$ with probabilities p_{kr}.

$$\sum_{r=1}^{R} p_{kr} = 1, \quad p_{kr} \ge 0,\ k = 1:K$$

Then the problem (33)-(37) is equivalent to:

$$\max_{x,u_{ik},u_{ik}^{-},\Lambda_k} \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_ix_j + \sum_{k=1}^{K} p_k \left(\sum_{i=1}^{N}\sum_{j=1}^{N} b_{ijk}u_{ik}u_{jk} - \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ijk}^{-}u_{ik}^{-}u_{jk}^{-}\right)$$

$$\sum_{i=1}^{N} w_ix_i \le d$$

$$u_{ik} \ge x_i - u_i^{-},\ i = 1:n,\ k = 1:K$$

$$u_{ik}^{-} \le x_i,\ i = 1:n,\ k = 1:K$$

$$\sum_{i=1}^{N} v_{ikr}u_{ik} \le h_{kr},\ r \in \Lambda_k,\ k = 1:K$$

$$\sum_{r\in\Lambda_k} p_{kr} \ge 1-\alpha,\ k = 1:K$$

where $v_{ikr} = v_{ik}(\psi_{kr})$, $h_{kr} = h_k(\psi_{kr})$ and Λ_k is some subset of $\{1, ..., R\}$. Finally, this can be reformulated as binary optimization problem by introducing auxiliary binary variable y_{kr} for each observation $r = 1:R$ and for each scenario $k = 1:K$ as follows:

$$y_{kr} = \begin{cases} 0 & \text{if} \quad r \in \Lambda_k \\ 1 & \text{otherwise} \end{cases}$$

This yields the following deterministic equivalent problem:

$$\max_{x,u,u^{-},y} \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_ix_j + \sum_{k=1}^{K} p_k \left(\sum_{i=1}^{N}\sum_{j=1}^{N} b_{ijk}u_{ik}u_{jk} - \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ijk}^{-}u_{ik}^{-}u_{jk}^{-}\right) \tag{38}$$

$$\sum_{i=1}^{N} w_ix_i \le d \tag{39}$$

$$u_{ik} \ge x_i - u_i^{-},\ i = 1:n,\ k = 1:K \tag{40}$$

$$u_{ik}^{-} \le x_i,\ i = 1:n,\ k = 1:K \tag{41}$$

$$\sum_{i=1}^{N} v_{ikr} u_{ik} \leq h_{kr} + M y_{kr},\ r = 1 : R,\ k = 1 : K \tag{42}$$

$$\sum_{r=1}^{R} p_{kr} y_{kr} \leq \alpha,\ k = 1 : K \tag{43}$$

where M_k is an arbitrary number such that

$$M_k \geq \max_r \sum_{i=1}^{N} v_{ikr} - h_{kr}$$

This problem is also a quadratic binary problem, and cannot be solved exactly for large instances due to time and memory issues. Therefore, we are interested in good upper bounds we can find, and again, we can use the SDP relaxations for this. Similarly to the static quadratic knapsack problem, semidefinite relaxations SDP1 and SDP2 can be applied to problem (38)-(43), as well as the LP relaxation we use as a basis for comparison.

3 Numerical Results

We performed numerical experiments using randomly generated data for the two formulations. We used various sizes of instances, in terms of number of items N, number of scenarios K, and, in the case of the knapsack with recourse, number of observations R. The experiments were carried out on Intel Xeon 5130 quad-core computers. The data sets (prices and scenarios for the weights, and, in the case with recourse, the prices and weights for the various observations in each scenario) were randomly generated using Matlab. Problems were solved using Cplex for exact (MIP) results and for the linear relaxation, and with CSDP for the two semidefinite relaxations.

3.1 Results for the static quadratic knapsack problem with probability constraint

Below, we present the results of the experiments we performed for the static quadratic knapsack problem. We used a number of items N ranging from $N = 20$ to $N = 60$, and a number of scenarios K ranging from $K = 10$ to $K = 50$. LP refers to the linear programming relaxation of Problem (9)-(11), SDP1 is the simple SDP relaxation (12), and SDP2 is the relaxation (13).

Table 1 gives the sizes of the instances tested. The first column gives the number of items N. The number of scenarios K is given in the second column. For the LP relaxation, we give the number of constraints and the number of variables. For the SDP1 and SDP2 relaxations, we give the number of constraints and the size of the matrix X. For example, for $N = 20$ and $K = 10$, the size of the matrix X is 31×31 for the SDP1 and SDP2 relaxations. The LP relaxation has $(K+1+N(N-1))$ constraints and $(K + N\frac{(N+1)}{2})$ variables. The SDP1 relaxation has $(2K + N + 2)$ constraints and the matrix X has a size $S = (N + K + 1)$. The SDP2 relaxation has $(5K + 2 + K^2 + 4KN + N^2)$ constraints and the matrix X has a size $S = (N + K + 1)$. In the LP relaxations, the number of variables and constraints increases polynomially with N. SDP1 relaxation is a direct rewriting of the original problem and therefore has a similar size. On the other hand, the SDP2 relaxation is penalized by the number of quadric polytope constraints and the Sherali-Adams constraints: the total number is polynomial with N and K. We see on Table 1 that SDP1 relaxation starts with more constraints than the LP relaxation, due to the diagonal constraint $diag(X) = z$, but does not grow as quickly since SDP1 does not use the linearization inequalities that the LP relaxation uses. On the other hand, SDP2 does use such inequalities combined to the Sherali-Adams constraints, which explains the number of constraints when the size of the problem increases.

		LP		SDP1		SDP2	
N	K	#constr	#var	#constr	#var	#constr	#var
20	10	391	220	42	[31]	1352	[31]
	20	401	230	62	[41]	2502	[41]
	50	431	260	122	[71]	7152	[71]
40	10	1571	830	62	[51]	3352	[51]
	20	1581	840	82	[61]	5302	[61]
	50	1611	870	142	[91]	12352	[91]
50	10	2461	1285	72	[61]	4652	[61]
	20	2471	1295	92	[71]	7002	[71]
	50	2501	1325	152	[101]	15252	[101]
60	10	3551	1840	82	[71]	6152	[71]
	20	3561	1850	102	[81]	8902	[81]
	50	3591	1880	162	[111]	18352	[111]

Table 1: LP and SDP test sizes

Table 2 presents the results for each instance. We first give the number of items N and the number of scenarios K. The third column (MIP) gives the value of the optimum for small and medium size instances. For each relaxation, we then give the upper bound found by the relaxation, and, whenever available, the gap between the optimum and this bound. When the optimum is not known, we give the improvement rate of the SDP2 relaxation compared to the LP relaxation. We can see on Table 2 that LP relaxation is generally poor and serves as reference when an exact solution could not be found. We observe that the quality of the relaxations improves when N grows, and worsens when K grows. This is very similar to the results on the non stochastic knapsack problem shown by D'Atri in [DR80] concerning the reduction in gap when the number of items grows with randomly generated data.

The SDP1 relaxation performs poorly – slightly worse than the LP relaxation. We only see a significant improvement in terms of gap reduction after adding the quadric polytope and Sherali-Adams constraints to obtain relaxation SDP2. On larger data sets ($N \geq 40, K \geq 20$), we reached the limit for Cplex to solve programs, and only relaxations could be solved, and we see that the SDP2 still brought a consequent improvement over the LP bound.

3.2 Results for the stochastic quadratic knapsack problem with recourse

We also performed tests for the model with recourse. In this model, the LP relaxation has $(1+KR+K+N^2-N+2KN^2)$ constraints and $(KR+KN+N(\frac{N+1}{2})+KN^2)$ variables. The SDP1 relaxation has $(3+4KN+2KR+3K+N)$ constraints with the diagonal block structure $S_1 = (N+1)$, $S_2 = (KR+1)$, and $S_{i+2} = (N+1), i = 1...2K$, where $S_1 ... S_{2*K+2}$ is the size of each submatrix on the diagonal of X. Finally, the SDP2 relaxation has $(3+2KN+3KR+3K+2N+N^2+K^2R^2+2KN^2)$

N	K	MIP	LP	GAP	SDP1	GAP	SDP2	GAP
20	10	4686	6915.11	47.57	7098	51.47	5602.75	19.56
	20	4384	6805.29	55.23	6895.47	57.29	5470.58	24.79
	50	3069	6470.66	110.84	6508.42	112.07	4381.96	42.78
40	10	18921	26703.23	41.13	27086.65	43.16	20901.39	10.47
	20	†	28318.38	-	28572.1	-	22101.84	22‡
	50	†	26297.62	-	26409.51	-	18889	28‡
50	10	34163	44497.27	30.25	45265.38	32.5	37615.69	10.11
	20	†	45412.71	-	45722.95	-	36768.51	19‡
	50	†	44557.85	-	44804.64	-	34189.01	23‡
60	10	†	69644.82	-	70369.88	-	59742.14	14‡
	20	†	65009.36	-	65395.68	-	51467.86	21‡
	50	†	62888.8	-	63142.06	-	46822.91	26‡

†: No solution given by CPLEX ‡: SDP2 improvement rate of LP bounds

Table 2: LP and SDP results

constraints, and the structure (and size) of X is the same as in the SDP1 relaxation. LP relaxation has a number of constraints and variables polynomial in N, while both SDP1 relaxation and the original problem sizes are comparable, i.e. linear with N. SDP2 relaxation is heavily penalized by the quadric polytope constraints and has a number of constraints polynomial with N, K and R. this is illustrated in table 3. In the model with no recourse, with $N = 40$ and $K = 50$ this leads to an SDP2 relaxation with 12,352 constraints, whereas a problem with recourse with $N = 10$, $K = 20$ and $R = 5$ has an SDP2 relaxation with 14,883 constraints. The largest set tested has an SDP2 relaxation with 27,603 constraints, which requires more memory than CSDP can allocate on a 32bit platform.

			LP		SDP1		SDP2	
N	K	R	#constr	#var	#constr	#var	#constr	#var
5	5	5	301	190	173	[[6] [26] 10*[6]]	1053	[[6] [26] 10*[6]]
	10	5	581	365	338	[[6] [51] 20*[6]]	3318	[[6] [51] 20*[6]]
	20	5	1141	715	668	[[6] [101] 40*[6]]	11598	[[6] [101] 40*[6]]
10	5	5	1121	630	278	[[11] [26] 10*[11]]	1938	[[11] [26] 10*[11]]
	10	5	2151	1205	543	[[11] [51] 20*[11]]	5003	[[11] [51] 20*[11]]
	20	5	4211	2355	1073	[[11] [101] 40*[11]]	14883	[[11] [101] 40*[11]]
20	5	5	4411	2335	488	[[21] [26] 10*[21]]	5358	[[21] [26] 10*[21]]
	10	5	8441	4460	953	[[21] [51] 20*[21]]	11523	[[21] [51] 20*[21]]
	20	5	16501	8710	1883	[[21] [101] 40*[21]]	27603	[[21] [101] 40*[21]]

Table 3: LP and SDP test sizes

Table 3 is similar to Table 1. The first three columns are the number of items N, the number of scenarios K, and the number of observations R, respectively. For each relaxation, we give the number of constraints and number of variables for the LP relaxation or the structure of the matrix

X for SDP1 and SDP2. For example, [[5] [26] 10*[5]] means the matrix X has a first block diagonal matrix of size 5×5, then a block diagonal matrix of size 26×26, and 10 consecutive block diagonal matrices of size 5×5 representing different variables. Table 4 presents the results in the following form: the first three columns are the same as Table 3, then, when its calculation is possible, we give the optimal value of the problem in the column MIP. The column block after gives the results for the LP relaxation: first the bound found, and when the optimum is known, the gap between the bound and the optimum. The two other column blocks give the bound found by each SDP relaxation, and the gap to the optimum whenever it can be calculated. For the SDP2 relaxation, when the optimum is not known, we give the improvement rate between the SDP2 bound and the LP bound.

Compared to the model with no recourse, the LP and SDP1 relaxations perform better, with a gap lower than 71.14%, and, as can be seen on table 4, there is still a gain from going to the SDP2 relaxation. Like previously, increasing N seems to positively affect the relaxations at large, but seems to also lower the gain between LP and SDP2. In this model, we see again that SDP1 and SDP2 allow us to solve larger instances than Cplex can. When sizes are so large that CSDP cannot be used, it is then necessary to use different resolution methods for SDP, such as the spectral bundle method of Helmberg and Rendl [HR00].

N	K	R	MIP	LP	GAP	SDP1	GAP	SDP2	GAP
5	5	5	907.50	1198.4	32.06	1217.25	34.13	1181.66	30.21
	10	5	988.89	1293.98	30.85	1320.39	33.52	1284.37	29.88
	20	5	†	1126.49	-	1143.64	-	1111.49	1.33‡
10	5	5	4002.04	4718.47	17.9	4796.03	19.84	4695.87	17.34
	10	5	†	4681.27	-	4752.48	-	4620.39	1.3‡
	20	5	†	4598.08	-	4668.99	-	4584.04	0.31‡
20	5	5	†	18801.41	-	19077.7	-	18791.08	0.05‡
	10	5	†	18501.64	-	18754.41	-	18341.11	0.87‡
	20	5	†	18252.31	-	18459.01	-	⋆	⋆

†: No solution given by CPLEX ‡: SDP2 improvement rate of LP bounds ⋆: No solution given by CSDP

Table 4: LP and SDP results

Conclusion

In this paper, we presented two problems and their formulation: static quadratic knapsack problem with a probability constraint and stochastic quadratic knapsack problem with recourse. For both problems, we applied semidefinite relaxations in order to obtain an upper bound of the optimum. We performed numerical experiments in which we solved large instances of the static quadratic knapsack problem with probability constraint and large instances of the stochastic quadratic knapsack problem with recourse. We showed that the tighter SDP relaxation always improves the quality of the bound compared to the simple SDP relaxation or the LP relaxation. In particular in the static case, the SDP2 bound is 14% to 26% better than the LP one. We also showed that SDP methods can be used to find a bound on instances which cannot be solved using CPLEX due to the memory and required time.

The stochastic quadratic knapsack problem with recourse shows the limits of interior point methods in solving such large scale problems. One way to overcome this limitation is to work with a different solving method, such as the spectral bundle method developed by Helmberg and Rendl [HR00]. This should allow the exploration of both larger problems, and the use of additional constraints to further tighten the relaxations.

References

[CB98] Amy Mainville Cohn and Cynthia Barnhart. The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning. In *Proceedings from TRISTAN III*, San Juan, Puerto Rico, 1998.

[DGV04] Brian C. Dean, Michel X. Goemans, and Jan Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 208–217, Washington, DC, USA, 2004. IEEE Computer Society.

[DR80] G. D'Atri and A. Di Rende. probabilistic analysis of knapsack-type problems. *Methods of Operations Research*, 40:279–282, 1980.

[For59] R. Fortet. L'algèbre de Boole et ses applications en recherche opérationelle. 1959.

[GJ79] M. R. Garey and D. S. Johnson. *Computer and Intractability.* W. H. Freeman and Company, New York, 1979.

[GLS88] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer, 1988.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[HPRW95] C. Helmberg, S. Poljak, F. Rendl, and H. Wolkowicz. Combining semidefinite and polyhedral relaxations for integer programs. In *IPCO*, pages 124–134, 1995.

[HR98] C. Helmberg and F. Rendl. Solving quadratic (0,1)- problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82:291–315, 1998.

[HR00] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[HRW00] C. Helmberg, F. Rendl, and R. Weismantel. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization*, 4(2):197–215, 2000.

[IK75] Oscar H. Ibarra and Chul E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, 1975.

[KP98] Anton J. Kleywegt and Jason D. Papastavrou. The dynamic and stochastic knapsack problem. *Oper. Res.*, 46(1):17–35, 1998.

[Lau03] Monique Laurent. A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0-1 programming. *Math. Oper. Res.*, 28(3):470–496, 2003.

[Pad89] M. Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Math. Program.*, 45(1):139–172, 1989.

[Pis07] David Pisinger. The quadratic knapsack problem-a survey. *Discrete Appl. Math.*, 155(5):623–648, 2007.

[RS03] F. Rendl and R. Sotirov. Bounds for the quadratic assignment problem using the bundle method. Technical report, University of Klagenfurt, Universitaetsstrasse 65-67, Austria, 2003.

[SA90] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3(3):411–430, 1990.

[SA94] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 52(1):83–106, 1994.

[SS06] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006.