Recoverable Robust Knapsacks: the Discrete Scenario Case

Christina Büsing* Arie M. C. A. Koster† Manuel Kutschka†

August 2010

Abstract

Admission control problems have been studied extensively in the past. In a typical setting, resources like bandwidth have to be distributed to the different customers according to their demands maximizing the profit of the company. Yet, in real-world applications those demands are deviating and in order to satisfy their service requirements often a robust approach is chosen wasting benefits for the company. Our model overcomes this problem by allowing a limited recovery of a previously fixed assignment as soon as the data are known by violating at most k service promises and serving up to ℓ new customers. Applying this approaches to the call admission problem on a single link of a telecommunication network leads to a recoverable robust version of the knapsack problem.

In this paper, we show that for a fixed number of discrete scenarios this recoverable robust knapsack problem is weakly **NP**-complete and any such instance can be solved in pseudo-polynomial time by a dynamic program. If the number of discrete scenarios is part of the input, the problem is strongly **NP**-complete and in special cases not approximable in polynomial time, unless $\mathbf{P} = \mathbf{NP}$.

Next to its complexity status we were interested in obtaining strong polyhedral descriptions for this problem. We thus generalized the well-known concept of covers to gain valid inequalities for the recoverable robust knapsack polytope. Besides the canonical extension of covers we introduce a second kind of extension exploiting the scenario-based problem structure and producing stronger valid inequalities. Furthermore, we present two extensive computational studies to (i) investigate the influence of parameters k and ℓ to the objective and (ii) evaluate the effectiveness of our new class of valid inequalities.

keywords: admission control, recoverable robustness, knapsack, extended cover inequalities

1 Introduction

Motivation We consider a telecommunication network in which an operator has to decide which demand is granted admission at a time. In many cases those requests made by the customers specifying the amount of traffic, the source and destination are submitted before their actual realization. The task of the operator is to give service promises according to these data maximizing the total profit for the company. Yet, in many applications the requests change at the time of realization. In case of slight deviations the customer still expects the promised quality of service.

A straight-forward approach dealing with demand uncertainties is by estimating the unknown demand with its deviations, and basing the resource distribution on the worst case occurence, a robust approach. In most cases this leads to over-conservative decisions where the largest part of the available bandwidth capacity is unused. This is clearly neither cost-efficient nor resource exploiting. There are at least two means of increasing the economical benefits of the company: first by relaxing the deviation assumption and second by satisfying all but k service promises. The

*Technische Universität Berlin, Institut für Mathematik, Straße des 17. Juni 136, D-10623 Berlin, Germany, cbuesing@math.tu-berlin.de

†RWTH Aachen University, Lehrstuhl II für Mathematik, Wüllnerstr. 5b, D-52062 Aachen, Germany, {koster,kutschka}@math2.rwth-aachen.de

first idea is reflected in the consideration of different scenario sets. The second idea is captured in the two stage concept of recoverable robustness. Here over-conservatism in robust optimization is avoided by allowing a limited recovery after the full data is revealed. This concept has been introduced by Liebchen et al. in [18] and applied to railway optimization problems as delay-management, platforming and shunting.

We adopted these approaches for the call admission problem on a single link of a telecommunication network. This problem can be seen as a classical knapsack problem, where the demands are represented by items and the amount of traffic is equivalent to the weights; the profit of an item is divided into two parts: the first stage profit captures the basic fee for a customer to use the network and the scenario profit the profit obtained by routing the demand. Finally, the bandwidth of the link is modeled by the knapsack capacity. As an adaption of the recoverable robust approach a first stage solution is in our case a subset of items such that the sum of weights does not exceed the link capacity. In the second stage, when the scenario is revealed and the scenario profit and weights are known, k service promises, i.e., k items, may be removed from the first stage solution. This new subset must satisfy the link capacity constraint of this scenario. The objective is to find a first stage solution with maximal total profit. The total profit of a given subset is the sum of the first stage profit representing the basic fee of the customers to use the network and the minimal scenario profit representing the actual routing rates.

Another motivation for the investigation of this recoverable robust knapsack problem can be found in wireless (cellular) networks (e.g., WLAN, 3G). Each antenna has a limited bandwidth capacity to be partitioned among the users in its cell (the area covered by the antenna). Users, however, do not generate a constant traffic rate. Depending on their needs (e.g., telephony, data traffic, web browsing), the requested data rate fluctuates (e.g., 64 Kbit/s, 384 Kbit/s, 2 Mbit/s). In the network capacity planning phase usually averages are considered. During operation, individual users with their actual data rates are admitted to the antenna. In the context of recoverable robust knapsack, the planning problem can be enhanced by considering the capacity planning as first stage knapsack, whereas snapshots of the operation can be taken as scenarios in the second stage. Compared to the planning phase, up to k users can be refused a connection, whereas up to ℓ new users can be admitted. Since this admission problem has to be considered by each antenna, the recoverable robust knapsack problem is a subproblem of such a recoverable robust wireless network planning problem.

Related Results The knapsack problem is one of the basic problems in combinatorial optimization. An instance of the knapsack problem (KP) consists of an item set $N = \{1, \dots, n\}$ with profits p_j and weights w_j for all items $j \in N$, and a capacity c. The values p_j, w_j, and c are taken as integers. The objective is to select a subset X of these items such that the total profit of X is maximized and the total weight does not exceed c. The knapsack problem provides a relaxation of several combinatorial optimization problem (e.g., generalized assignment, capacitated facility location, capacitated vehicle routing, airline scheduling). Despite its simple structure the problem is known to be weakly **NP**-hard [16] but solvable via dynamic programming in pseudo-polynomial time [7, 10]. Different branch-and-cut algorithms are used to solve this problem in practice. A detailed introduction to the knapsack problem and its variations can be found in Martello and Toth [19] and Kellerer, Pfeschy and Pisinger [17].

In Yu [23] a robust version of the knapsack problem is defined by introducing uncertainty in the profit values: Let $\mathcal{S}$ be a set of scenarios, each scenario S determining a profit function $p^S : N \to \mathbb{N}$. A robust knapsack problem is to find a set of items maximizing the minimum profit over all scenarios such that the total weight does not exceed the capacity. By a reduction from the set covering problem, Yu showed that for discrete scenarios sets with an unbounded number of scenarios the decision version of the problem is strongly **NP**-complete. As mentioned by Aissi et al. [3] this proof includes the result that the optimization version is not f-approximable for any function $f : N \to (1, \infty)$. For a discrete scenario set with a bounded number of scenarios the problem is weakly **NP**-complete, solvable in pseudo-polynomial time [14, 23] and there exist

an FPTAS [3]. Iida [12] provided a computational study on the robust knapsack problem with discrete scenarios comparing several methods to derive upper and lower bounds for the profit.

Model and Notation We introduce a recoverable robust version of the knapsack problem, in which the weights as well as the profits are subject to uncertainties. Those uncertainties are given as before via a set of scenarios. In this recoverable robust counterpart a first stage solution is a subset of items such that its first stage weight does not exceed the first stage capacity. In the second stage, i.e., when the scenario is revealed and the profits and weights are known, k items may be removed from the first stage solution and ℓ items may be added. This new subset must satisfy the scenario capacity restriction c^S according to the weights of scenario S. The objective is to find a first stage solution with maximum total profit. The total profit is the sum of the first stage profit and the minimum scenario profit.

Definition 1.1 ((k,ℓ)-Recoverable Robust Knapsack Problem ((k,ℓ)-rrKP))**.** Let $N = \{1,\dots,n\}$ be a set of items, $c^0 \in \mathbb{N}$ the first stage capacity, p_j^0 the first stage profit and w_j^0 the first stage weight of each item $j \in N$. Each scenario S of a given set of scenarios $\mathcal{S}$ defines a profit function $p^S : N \to \mathbb{N}$, a weight function $w^S : N \to \mathbb{N}$ and a capacity $c^S \in \mathbb{N}$. Furthermore, the parameter $k \in \mathbb{N}$ limits the number of deletable items from a first stage solution and the parameter $\ell \in \mathbb{N}$ the number of new items contained in a recovered solution. For a given subset $X \subseteq N$ the *recovery set* $\mathcal{X}_X^{(k,\ell)}$ includes all subsets of N which contain at most ℓ additional elements and fail to contain at most k elements of X, i.e.,

$$\mathcal{X}_X^{(k,\ell)} = \{X' \subseteq N \mid |X \backslash X'| \leq k \text{ and } |X' \backslash X| \leq \ell\}.$$

With $\mathcal{X}^S$ for $S \in \mathcal{S}$ we denote all subsets of $X' \subseteq N$ that satisfy the *scenario weight constrain* $\sum_{i \in X'} w_i^S \leq c^S$. A *feasible first stage solution* of the (k,ℓ)-recoverable robust knapsack problem is a subset $X \subseteq N$ which satisfies the *first stage weight constraint* $\sum_{i \in X} w_i^0 \leq c^0$ and $\mathcal{X}_X^{(k,\ell)} \cap \mathcal{X}^S \neq \emptyset$ for all $S \in \mathcal{S}$. An *optimal* solution $X^* \subseteq N$ of the (k,ℓ)-recoverable robust knapsack problem is a feasible solution with maximum total profit. The *total profit* $p(X)$ of a feasible solution X is defined as

$$p(X) = \sum_{i \in X} p_i^0 + \min_{S \in \mathcal{S}} \max_{X^S \in \mathcal{X}_X^{(k,\ell)} \cap \mathcal{X}^S} \sum_{i \in X^S} p_i^S.$$

In a discrete scenario set $\mathcal{S}_D$ every scenario is explicitly given with its weight function $w^S : N \to \mathbb{N}$, its profit function $p^S : N \to \mathbb{N}$ and its capacity c^S. The following integer program models the (k,ℓ)-recoverable robust knapsack problem

(ILP1)
$$\max \sum_{i \in N} p_i^0 x_i + \omega \tag{1}$$
$$\sum_{i \in N} w_i^0 x_i \leq c^0 \tag{2}$$
$$\sum_{i \in N} w_i^S x_i^S \leq c^S \quad \forall S \in \mathcal{S}_D \tag{3}$$
$$x_i^S - x_i \leq y_i^S \quad \forall S \in \mathcal{S}_D,\ i \in N \tag{4}$$
$$x_i - x_i^S \leq z_i^S \quad \forall S \in \mathcal{S}_D,\ i \in N \tag{5}$$
$$\sum_{i \in N} y_i^S \leq \ell \quad \forall S \in \mathcal{S}_D \tag{6}$$
$$\sum_{i \in N} z_i^S \leq k \quad \forall S \in \mathcal{S}_D \tag{7}$$
$$\omega - \sum_{i \in N} p_i^S x_i^S \leq 0 \quad \forall S \in \mathcal{S}_D \tag{8}$$
$$\omega \in \mathbb{Q}_+,\ x_i, x_i^S, y_i^S, z_i^S \in \{0,1\} \quad \forall S \in \mathcal{S}_D, i \in N$$

The variable x represents the first stage solution, x^S the solution taken in scenario $S \in \mathcal{S}_D$, y_i^S determines if item i is added in S and z_i^S if item i is removed from x in S. Inequalities (4)-(7) guarantee that x^S is a feasible recovery for the first stage solution x and (2) and (3) that the weight constraints are obeyed. The last inequality (8) in combination with the objective function models the total profit

$$\max \sum_{i \in N} p_i^0 x_i + \min_{S \in \mathcal{S}_D} \max \sum_{i \in N} p_i^S x_i.$$

The minimum scenario profit, i.e., $\min_{S \in \mathcal{S}_D} \max \sum_{i \in N} p_i^S x_i$, is captured in the variable ω. The size of the program depends on the number of scenarios.

Results Our research focuses on adapting classical results of the (robust) knapsack problem for this recoverable robust version of the knapsack problem with discrete scenarios. We start with a complexity study. For a bounded number of discrete scenarios, the recoverable robust knapsack problem is weakly **NP**-complete. Any such instance can be solved in pseudo-polynomial time by a dynamic program. If the number of discrete scenarios is not bounded, the problem is strongly **NP**-complete and in special cases not approximable in polynomial time, unless $\mathbf{P} = \mathbf{NP}$ (Section 2). Next to its complexity we were interested in obtaining strong polyhedral descriptions for this problem. We adapted the well known (minimal/extended) cover inequalities for the knapsack problem to gain valid inequalities for the recoverable robust knapsack polytope (Section 3). Any 0-1-point satisfying all those inequalities is a feasible solution to the rrKP instance. Finally, we present computational studies to investigate the impact of these new inequalities for solving rrKP instances as well as the the gain of recovery (Section 4).

2 Complexity of the (k, ℓ)-rrKP

We start with an analysis of the complexity status of the (k, ℓ)-rrKP for discrete scenarios and thus consider its decision version:

Given: A set $N = \{1, \dots, n\}$, a first stage capacity $c^0 \in \mathbb{N}$, a first stage profit function $p^0 : N \to \mathbb{N}$ and a first stage weight function $w^0 : N \to \mathbb{N}$, a set of discrete scenarios $\mathcal{S}_D$ each scenario defining a profit function $p^S : N \to \mathbb{N}$ and a weight function $w^S : N \to \mathbb{N}$, and some parameters $k, \ell, K \in \mathbb{N}$.

Find: A feasible first stage solution $X \subseteq N$ with a total profit $p(X) \geq K$.

If such an instance I is a yes-instance, there exits a set $X \subseteq N$ and sets $X^S \subseteq N$ for all $S \in \mathcal{S}_D$, such that X is a feasible first stage solution, $X^S \in \mathcal{X}_X^{(k,\ell)} \cap \mathcal{X}^S$ and $p(X) = \sum_{i \in X} p_i^0 + \min_{S \in \mathcal{S}} \sum_{j \in X^S} p_j^S \geq K$. Since the feasibility of X and $X^S \in \mathcal{X}_X^{(k,\ell)} \cap \mathcal{X}^S$ can be tested in polynomial time, the decision version of the (k, ℓ)-rrKP problem with discrete scenarios is in **NP**. Yet, there will be no polynomial certificate to test for a given subset $X \subseteq N$ without any further information whether X is a feasible solution with a total profit $p(X) \geq K$, unless $\mathbf{P} = \mathbf{NP}$. More precisely, it is weakly **NP**-hard to compute the total profit of a given set of items even for one scenario.

Lemma 2.1. *The decision if the total profit of a feasible first stage solution X is greater or equal to a constant K for just one scenario is weakly* **NP***-hard, even if $k = 0$ or $\ell = 0$.*

Proof. We start with the general case of $k \neq 0$ and $\ell \neq 0$ and reduce from the knapsack problem. Let I be a KP instance with an item set $N = \{1, \dots, n\}$, a weight function $w : N \to \mathbb{N}$, a profit function $p : N \to \mathbb{N}$ and a capacity value c. We construct a (k, ℓ)-recoverable robust knapsack instance I' with one scenario S in the following way: the set of items remains the same,

$w^0 = p^0 = c^0 = 0$, $k = n$, $\ell = n$, $w^S = w$, $p^S = p$ and $c^S = c$. Any subset $X \subseteq \{1, \dots, n\}$ is a feasible solution of the instance I'. The total profit of any set X is greater or equal than K if and only if there exists a feasible solution to I with a profit greater or equal than K. Replacing $k = n$ by $k = 0$ in I', the total profit of the feasible first stage solution $X = \emptyset$ in I' indicates whether I is a yes-instance. For $\ell = 0$, the total profit of N is greater or equal than K if and only if I is a yes-instance. □

If k and ℓ are constant, the recovery set $\mathcal{X}_X^{(k,\ell)}$ contains a constant number of solutions for any $X \subseteq N$. Hence, the total profit can be computed in polynomial time by enumeration.

Since the Knapsack problem is a special case of the (k, ℓ)-rrKP, the (k, ℓ)-rrKP remains at least weakly **NP**-complete for one scenario. We will later show by introducing a pseudo-polynomial algorithm, that the problem is indeed weakly **NP**-complete if the number of scenarios is bounded. We now prove that even in the case $p^0 = 0$ the (k, ℓ)-rrKP is strongly **NP**-complete if the number of scenarios in $\mathcal{S}_D$ is part of the input.

Theorem 2.2. *The (k, ℓ)-rrKP is strongly* **NP***-complete for an unbounded sets of discrete scenarios even if $p^0 = 0$.*

Proof. We reduce from 3SAT. Let I be an instance of 3SAT with n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$. Each clause is formed by three literals. The corresponding instance I' of the (k, ℓ)-rrKP problem contains a set of items $N = N_1 \cup N_2 \cup N_3$ with $N_1 = \{1, \dots, 2n\}$, $N_2 = \{2n + 1, \dots, 2n + k\}$ and $N_3 = \{2n + k + 1, \dots, 2n + k + \ell\}$. The items in N_1 represent the true or false assignment of the variables x. The last $k + \ell$ items are auxiliary items to fix the recovery action for any reasonable first stage solution. The parameters of the first stage are set to

$$c^0 = n + k,\ w^0(i) = \begin{cases} 1 & i \in N_1 \cup N_2 \\ c^0 + 1 & \text{otherwise} \end{cases},\ p^0(i) = 0\ \forall i \in N,\ \text{and } K = 1 + \ell.$$

Hence, no feasible first stage solution contains the last ℓ items. But the profit functions of each scenario will be constructed in a way such that all those items have to be added to the recovery solution for each scenario to obtain a total profit of at least K. It remains to define those scenarios $\mathcal{S}$. This set consists of three different types: The first type forces any first stage solution with a total profit greater than K to incorporate the items in N_2. Furthermore, those items need to be removed in any other recovery solution and thus no item of N_1 can be removed. The second type guarantees that for $j = \{1, \dots, n\}$ either $2j - 1$ or $2j$ is part of the first stage solution. The choice models the assignment of the variables x_i to true or false. The last type assures that all clauses are satisfied.

First type of scenarios: For the items $j \in N_2$ we add a scenario S_j^1 to I' with $c^{S_j^1} = 0$,

$$p^{S_j^1}(i) = \begin{cases} 1 & i = j \text{ or } i \in N_3 \\ 0 & \text{otherwise} \end{cases} \text{ and } w^{S_j^1}(i) = 0$$

for $i \in N$. Hence, a first stage solution must contain those k items, to get a value greater than K. Second type of scenarios: For each variable x_j, $j \in \{1, \dots, n\}$, we add S_j^2 to I' defining $c^{S_j^2} = 0$,

$$p^{S_j^2}(i) = \begin{cases} 1 & i \in \{2j - 1, 2j\}, i \in N_3 \\ 0 & \text{otherwise} \end{cases},\ w^{S_j^2}(i) = \begin{cases} 1 & i \in N_2 \\ 0 & \text{otherwise} \end{cases}$$

for $i = N$. Third type of scenarios: For each clause $j = 1, \dots, m$ we construct one scenario S_j^3 which puts a profit of 1 to all items representing a verification of the clause, i.e., $p^{S_j^3}(i) = 1$ for an even $i \in N_1$ if and only if $\overline{x}_i \in C_j$ and $p^{S_j^3}(i) = 1$ for an odd $i \in N_1$ if and only if $x_i \in C_j$. Furthermore, all items $i \in N_3$ are assigned a profit of 1. The weight of all items $i \in N_1 \cup N_3$ in all

those scenarios and the scenario capacity are set to 0. The weights of items $i \in N_2$ are set to 1. The size of the constructed instance is polynomial in the size of I. A true assignment of the 3SAT instance I exists if and only if there exists a solution of I' with a total profit greater than K. □

Note that the optimal value of the rrKP instance in the reduction determines a lower bound on the approximation ratio of $\frac{\ell+1}{\ell}$ for $\ell \in \mathbb{N}$. This implies that any $(k,0)$-rrKP is inapproximable, unless $\mathbf{P} = \mathbf{NP}$. Finally, we consider the special case, in which all scenario profits are set to 0.

Theorem 2.3. *The (k,ℓ)-rrKP is strongly* **NP***-complete for unbounded sets of discrete scenarios even when $p^S = 0$ for all $S \in \mathcal{S}_D$.*

Proof. The reduction is a modification of the proof of Theorem 2.2. In that case the first stage profit is set to 1 for all items $i \in N_1$, to $2n$ for $i \in N_2$, to 0 otherwise and $K = n + 2nk$. Hence, any first stage solution X with $p(X) \geq K$ needs to contain all items of N_2. The first stage weight equals 1 for all items and the first stage capacity $n + k$. A scenario S_j^1 representing a clause C_j sets $w^{S_j^1}(i) = 1$ if and only if the corresponding variable falsifies the clause for $i \in N_1$, $w^{S_j^1}(i) = 3$ for $i \in N_2$ and 0 otherwise. The capacity for each clause equals 2. Furthermore, for each variable x_j, $j = 1, \ldots, n$, we add a scenario S_j^2 with $w^{S_j^2}(2j-1) = 1$, $w^{S_j^2}(2j) = 1$, $w^{S_j^2}(i) = 2$ for $i \in N_2$, $w^{S_j^2}(i) = 0$ otherwise and $c^{S_j^2} = 1$. Those scenarios guarantee that either the item $2j-1$ or $2j$ is in a feasible first stage solution. □

This reduction does not imply a lower bound on the approximation factors.

Bounded number of scenarios We will now demonstrate, that the (k,ℓ)-recoverable robust knapsack problem can be solved in pseudo-polynomial time for a bounded number of scenarios. The presented algorithm is based on dynamic programming and extends the idea of Yu [23] for the robust case. The concept of the algorithm is to compute recursively the value of the optimization problem when the optimal selection is made among the first t items under the first stage capacity d^0, the scenario dependent recovery parameters (j^S, i^S), the scenario capacities d^S and a guaranteed extra profit in each scenario S of α^S induced by the items $\{t+1, \ldots, n\}$. This problem is modeled by the integer program

$$
\begin{aligned}
g(t, d^0, v^{S_1}, \ldots, v^{S_r}) = \max & \sum_{i=1}^{t} p_i^0 x_i + \omega \\
& \sum_{i=1}^{t} w_i^0 x_i \leq d^0 \\
& \sum_{i=1}^{t} w_i^S x_i^S \leq d^S \quad \forall S \in \mathcal{S}_D \\
& x_i^S - x_i \leq y_i^S \quad \forall i \in N_t,\ \forall S \in \mathcal{S}_D \\
& x_i - x_i^S \leq z_i^S \quad \forall i \in N_t,\ \forall S \in \mathcal{S}_D \\
& \sum_{i=1}^{t} y_i^S \leq i^S \quad \forall S \in \mathcal{S}_D \\
& \sum_{i=1}^{t} z_i^S \leq j^S \quad \forall S \in \mathcal{S}_D \\
\omega - \sum_{i=1}^{t} p_i^S x_i^S + \alpha^S \leq 0 & \quad \forall S \in \mathcal{S}_D \qquad (9) \\
x_i, x_i^S, y_i^S, z_i^S \in \{0,1\} & \quad \forall i \in N_t,\ S \in \mathcal{S}_D
\end{aligned}
$$

to the given parameters t, d^0, v^S with $v^S = (d^S, \alpha^S, j^S, i^S)$ for all $S \in \mathcal{S}_D$ and $N_t := \{1, \dots, t\}$. As in the ILP1-formulation the variable x represents the first stage solution, x^S the solution taken in scenario $S \in \mathcal{S}_D$, y_i^S indicates if item i is added in S and z_i^S if item i is removed from x in S. The inequality (9) differs from the inequality (8) in the ILP1-formulation by adding the parameter α^S to the maximum profit achieved by any recovered solution in scenario S. This parameter α^S models a guaranteed profit in scenario S obtained by items $t+1, \dots, n$ under consideration of the remaining capacity $c^S - d^S$ and recovery means $(k - j^S, \ell - i^S)$.

In order to simplify the initialization, we add an item $i = 0$ with $p_0^0 = 0$, $p_0^S = 0$ for all scenarios $S \in \mathcal{S}_D$, $w_0^0 = c^0 + 1$ and $w_0^S = c^S + 1$ for all $S \in \mathcal{S}$. Hence, any feasible solution sets $x_0 = 0$ and $g(0, d^0, v^{S_1}, \dots, v^{S_r}) = 0$ if $d^0 \geq 0$, $d^S \geq 0$, $j^S \geq 0$ and $i^S \geq 0$ for all $S \in \mathcal{S}_D$. We denote this set of all feasible parameter sets with $\mathcal{Z}$, i.e.,

$$\mathcal{Z} = \{0, \dots, n\} \times \{0, \dots, c^0\} \times \prod_{i=1}^{r} \big\{\{0, \dots, c^S\}, \{0, \dots, P_{\max}\}, \{0, \dots, k\}, \{0, \dots, \ell\}\big\}$$

with $P_{\max} = \max_{S \in \mathcal{S}_D} \sum_{i=1}^{n} p_i^S$. For $\gamma = (t, d^0, d^{S_1}, \alpha^{S_1}, j^{S_1}, i^{S_1}, \dots, i^{S_r})$ we denote γ_1 with $t(\gamma)$, γ_2 with $d^0(\gamma)$ and so on. If there is a parameter set $\gamma \notin \mathcal{Z}$ with $\gamma(t) = 0$, there exists no feasible solution for $g(\gamma)$ and thus, we set $g(\gamma) = -\infty$.

In the next step, we define a recursion formula computing the value of $g(y)$ out of the values of $g(y')$ with $t(y') = t(y) - 1$. Loosely speaking, we decide if item $t(y)$ is part of the first stage solution and in which scenarios the recovery action is taking place. More formally, for a given parameter set γ we define $g^+(\gamma)$ as the maximum value of $g(\gamma)$ if the item $t(\gamma)$ is added to the first stage solution. As we will show $g^+(\gamma)$ can recursively be computed by

$$g^+(t, d^0, v^{S_1}, \dots, v^{S_r}) = \max_{\beta \in \{-,\sim\}^r} g(t-1, d^0 - w_t^0, v_{\beta_1}^{S_1}, \dots, v_{\beta_r}^{S_r}) + p_t^0$$

with $v_-^S = (d^S, \alpha^S, j^S - 1, i^S)$ and $v_\sim^S = (d^S - w_t^S, \alpha^S + p_t^S, j^S, i^S)$. Here, the vector v_-^S represents the decision to delete this item in the recovered solution of scenario S and $v_\sim^S$ the decision to keep the item. In the same way we define $g^-(\gamma)$ as the maximum value of $g(\gamma)$ if the item $t(\gamma)$ is not added to the first stage solution. This can also be recursively be computed by

$$g^-(t, d^0, v^{S_1}, \dots, v^{S_r}) = \max_{\beta \in \{+,0\}^r} g(t-1, d^0, v_{\beta_1}^{S_1}, \dots, v_{\beta_r}^{S_r})$$

with $v_+^S = (d^S - w_t^S, \alpha^S + p_t^S, j^S, i^S - 1)$ and $v_0^S = v^S$. The vector v_+^S represents the case to add this item in the recovered solution of scenario S and v_0^S to refrain from doing so. The combination of those two values determines $g(\gamma)$ as

$$g(\gamma) = \max\{g^+(\gamma), g^-(\gamma)\}.$$

The value of an optimal solution is given by $g(n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$ with $\overline{v}^{S_i} = (c^{S_i}, 0, k, \ell)$. To obtain an optimal solution $x^*, x^{S_1}, \dots, x^{S_r}$ to $g(n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$ we consider a feasible sequence γ^t, $t = 0, \dots, n$, in $\mathcal{Z}$ with $\gamma^n = (n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$. A sequence γ^t, $t = 0, \dots, \ell$, is called *feasible*, if $\gamma^t \in \mathcal{Z}$ for all $t = 0, \dots, \ell$, $t(\gamma^i) = i$, and $\gamma^{t-1} \in \mathcal{Z}(\gamma^t)$ with

$$g(\gamma^t) = \begin{cases} g(\gamma^{t-1}) & \text{if } \gamma^{t-1} \in \mathcal{Z}^-(\gamma^t) \\ g(\gamma^{t-1}) + p_i^0 & \text{if } \gamma^{t-1} \in \mathcal{Z}^+(\gamma^t) \end{cases}$$

for $t = 1, \dots, \ell$. The set $\mathcal{Z}(\gamma)$ contains all possible predecessors of a given parameter γ. An element γ' is called a *predecessor* of γ, if $t(\gamma') = t(\gamma) - 1$, $d^0(\gamma') = d^0(v) - w_{t(\gamma)}^0$ or $d^0(\gamma') = d^0(\gamma)$ and

$$v^S(\gamma') \in \begin{cases} \{v_+^S(\gamma), v_0^S(\gamma)\} & \text{if } d^0(\gamma') = d^0(\gamma) \\ \{v_-^S(\gamma), v_\sim^S(\gamma)\} & \text{if } d^0(\gamma') = d^0(\gamma) - w_{t(\gamma)}^0. \end{cases}$$

It is element of $\mathcal{Z}^+(y)$, if $d^0(\gamma') = d^0(\gamma) - w^0_{i(\gamma)}$ and otherwise it is an element of $\mathcal{Z}^-(\gamma)$. According to such a feasible sequence γ^t, $t = 0, \ldots, n$, with $\gamma^n = (n, c^0, \overline{v}^{S_1}, \ldots, \overline{v}^{S_r})$, also called a *solution sequence*, a solution $x^*, x^{S_1}, \ldots, x^{S_r}$ is defined by $x_0 = 0$, $x^S_0 = 0$ for all $S \in \mathcal{S}_D$,

$$x^*_t = \begin{cases} 0 & \text{if } \gamma^{t-1} \in \mathcal{Z}^-(\gamma^t) \\ 1 & \text{otherwise} \end{cases} \text{ and } x^S_t = \begin{cases} 0 & \text{if } v^S(\gamma^{t-1}) = \{v^S_0(\gamma^t), v^S_\sim(\gamma^t)\} \\ 1 & \text{otherwise} \end{cases}$$

for all $S \in \mathcal{S}_D$ and $t \geq 1$. Based on the initial condition and the recursion a dynamic program can easily be achieved with a run-time of $\mathcal{O}(n \cdot c_{\max}(c_{\max} \cdot P_{\max} \cdot k \cdot \ell)^{|\mathcal{S}_D|} \cdot 2^{|\mathcal{S}_D|})$ with $c_{\max} = \max\{c^0, \max_{S \in \mathcal{S}_D} c^S\}$ and $P_{\max} = \max\{\sum_{i=1}^n p^0_i, \max_{S \in \mathcal{S}_D} \sum_{i=1}^n p^S_i\}$.

Theorem 2.4. *Let $x^*, x^{S_1}, \ldots, x^{S_r}$ be some 0-1 points computed according to the recursive formula 2 and a solution sequence γ^t, $t = 0, \ldots, n$. Then this is an optimal solution to the given (k, ℓ)-rrKP instance with $p(x^*) = g(\gamma^n)$.*

In the following subsection we will focus on the polyhedral structure of an (k, ℓ)-rrKP instance.

3 Extended Cover-Inequalities

As pointed out by Crowder et al. [9] one motivation for studying the polytope of the knapsack problem is that valid inequalities can be used as cutting planes for general 0-1 linear integer programs (ILP). The idea is to consider each individual constraint of a 0-1 ILP as a 0-1 knapsack constraint. Several classes of valid inequalities are known, such as the lifted cover inequalities (LCIs) of Balas [4] and Wolsey [22], the weight inequalities (WIs) of Weismantel [21], or the $(1, k)$-configurations introduced by Padberg [20]. For the class of (k, ℓ)-rrKP problems we will focus on extended cover inequalities which have been shown to be quite efficient in solving knapsack instances by Kaparis and Letchford [15].

The knapsack polytope is the convex hull of the set of solution vectors for a knapsack problem, i.e.,

$$\mathcal{K} := \operatorname{conv}\left\{ x \in \{0,1\}^n \mid \sum_{i \in N} w_i x_i \leq c \right\}.$$

An important class of valid inequalities for the knapsack problem, are the so-called minimal cover inequalities and their extensions. A subset $C \subseteq N$ is called a *cover* for $\mathcal{K}$ if $\sum_{j \in C} w_j > c$. A cover is *minimal* if $\sum_{j \in C \setminus \{i\}} w_j \leq c$ for every $i \in C$. For a (minimal) cover C the *(minimal) cover inequality*

$$\sum_{j \in C} x_j \leq |C| - 1$$

provides a valid inequality for $\mathcal{K}$. The minimal cover inequalities are facet-defining for $C = N$, but in general they are not. Yet, they can be lifted to define a facet of the knapsack polytope. One way to lift cover inequalities is based on an *extension* $E(C)$ of a cover C

$$E(C) := C \cup \{j \in N \setminus C \mid w_j \geq w_i, i \in C\}.$$

The resulting *extended cover inequality* for $\mathcal{K}$ reads

$$\sum_{j \in E(C)} x_j \leq |C| - 1.$$

In general, it is not reasonable to generate all extended cover inequalities, but solve the LP-relaxation and find for a non-integer point a violated inequality. This problem is called the *separation problem*. Thus, in the case of (extended) cover inequalities we are interested in finding

for a given fractional solution such an inequality that is violated or state that no such inequality exists. More formally, let $x^* \in [0,1]^n$ be the fractional solution with $\sum_{i \in N} w_i x_i \leq c$. The separation problem is to find a cover $C \subseteq N$, such that

$$\sum_{j \in C} x_j^* \geq |C|.$$

This task can be translated into:

$$\begin{aligned} & \sum_{j \in C} x_j^* \leq |C| - 1 && \forall\, C \text{ cover} \\ \Leftrightarrow \quad & \sum_{j \in C} x_j^* - |C| \leq -1 && \forall\, C \text{ cover} \\ \Leftrightarrow \quad & \max_{C \text{ cover}} |C| - \sum_{j \in C} x_j^* \leq -1 \end{aligned}$$

Introducing 0-1 variables y_i, taking the value 1 if and only if item j is part of the cover, the last problem is equivalent to solving the knapsack problem:

$$\begin{aligned} z^* = \max & \sum_{j \in N} (x_j^* - 1) y_j \\ & \sum_{j \in N} w_j y_j \geq c + 1 \\ & y_j \in \{0,1\}. \end{aligned} \tag{10}$$

There is a violated cover inequality to the point x^* if and only if $z^* > -1$, as Crowder et. al. [9] noted in 1983. Yet, solving the problem is weakly **NP**-hard [11].

In the following part we show how the classical concept of cover inequalities can be generalized for the (k,ℓ)-rrKP polytope in a quite natural fashion.

rrKP extended cover inequalities Before we start with the investigation of robust covers, we introduce the following notation: for a function $f : N \to \mathbb{N}$, a set $X \subseteq N$ and an integer r we define $f(C) = \sum_{i \in C} f(i)$, $f(\max, X, r) = \max_{\substack{K \subseteq X \\ |K| \leq r}} f(K)$ and $f(\min, X, r) = \min_{\substack{K \subseteq X \\ |K| \geq r}} f(K)$.

For a given (k,ℓ)-rrKP instance the (k,ℓ)-recoverable robust knapsack polytope $\mathcal{K}_D(k)$ of a discrete scenario set $\mathcal{S}_D$ is the convex hull over all valid first stage solutions, i.e.,

$$\mathcal{K}_D(k) := \operatorname{conv}\left\{ x \in \{0,1\}^n \mid \sum_{i \in N} w_i^0 x_i \leq c^0 \text{ and } \min_{\substack{T \subseteq N \\ |T| \leq k}} \sum_{i \in N \setminus T} w_i^S x_i \leq c^S \right\}.$$

Note that ℓ does not play a role in the feasibility of a first stage solution. The polytope $\mathcal{K}_D(k)$ has full dimension if and only if $w_i^0 \leq c^0$ for all $i \in N$ and $(1-k) w_i^S \leq c^S$ for all $S \in \mathcal{S}$.

Following the concept of covers we call a set $C \subseteq N$ an *rrKP cover* if one of the next condition holds

1. $w^0(C) \geq c^0 + 1$
2. $\exists S \in \mathcal{S}_D$ with
$$w^S(C) - w^S(\max, C, k) \geq c^S + 1.$$

An rrKP cover C is *minimal* if

1. $w^0(C) - w^0(\min, C, 1) \leq c^0$
2. $w^S(C) - w^S(\max, C, k) - w^S(\min, C, 1) \leq c^S \quad \forall S \in \mathcal{S}_D$.

and an rrKP cover defines the following *cover inequality*

$$\sum_{i \in C} x_i \leq |C| - 1.$$

Theorem 3.1. *Let x be a 0-1 point. Then $x \in \mathcal{K}_D(k)$ if and only if x satisfies all minimal cover inequalities.*

Proof. ($\Rightarrow$): Let $x \in \mathcal{K}_D(k)$ and let us assume there exists a minimal rrKP cover C such that

$$\sum_{i \in C} x_i = |C|.$$

Since C is a cover, there exits either a scenario $S \in \mathcal{S}_D$ such that

$$w^S(C) - w^S(\max, C, k) \geq c^S + 1$$

or

$$w^0(C) \geq c^0 + 1.$$

This is a contradiction to the feasibility of x.

($\Leftarrow$): Let $T_x := \{i \in N \mid x_i = 1\}$ be the support of x. Let us further assume x is infeasible, i.e., either

$$w^0(T_x) \geq c^0 + 1$$

or there exists a scenario $S \in \mathcal{S}_D$ with

$$w^S(T_x) - w^S(\max, T_x, k) \geq c^S + 1.$$

Hence, T_x is an rrKP cover by definition. In a last step we modify T_x to become a minimal rrKP cover. For this reason we repeatedly remove an item $i \in T_x$ with $i = \arg\min_{i \in T_x} w_i^0$ if $w^0(T_x) \geq c^0 + 1$ or $i = \arg\min_{i \in T_x} w_i^S$ if $w^S(T_x) - w^S(\max, T_x, k) \geq c^S + 1$, until the remaining set T_x' is a minimal rrKP cover. Thus, x violates the (minimum) cover inequality defined by T_x'. □

As in the case of the deterministic knapsack polytope, a cover inequality is facet defining, if a minimal rrKP cover C contains all items of N: Let us consider the sets $N_i := N \backslash \{i\}$ for $i = 1, \ldots, n$. All those sets are feasible solutions to the knapsack instance, since $C = N$ is a minimal rrKP cover. Thus, N_i are n affine independent solutions and C is therefore facet defining.

The concept of covers can be extended to strengthen the cover inequalities in a similar fashion as for the knapsack polytope. A canonical way to define an extension is by adding all items whose weight is greater or equal than the highest not recovered item in an rrKP cover C. More formally, let C be an rrKP cover and S a scenario such that the scenario weight inequality is violated by C. A *canonical extension* $\overline{E}^S(C)$ is given by

$$\overline{E}^S(C) = \{i \in N \mid w_i^S \geq w^S(\max, C, k+1) - w^S(\max, C, k)\} \cup C. \tag{11}$$

Yet, it even suffice for an item to be added to C if its weight exceeds the residual capacity according to the weight of the first $|C| - k - 1$ items with lowest weight and the weight of the item with the second highest not recovered item in C.

Definition 3.2. Let C be an rrKP cover and S a scenario with $w^S(C) - w^S(\max, C, k) \geq c^S + 1$. An *extension* $E^S(C)$ of C according to S is defined by

$$\begin{aligned} E^S(C) =& C \cup \{i \in N \mid w_i^S \geq c^S - w^S(C) + w^S(\max, C, k+1) + 1, \\ & w_i^S \geq w^S(\max, C, k+2) - w^S(\max, C, k+1)\} \end{aligned}$$

and determines the *rrKP extended cover inequality*

$$\sum_{i \in E^S(C)} x_i \leq |C| - 1.$$

For the deterministic knapsack problem the set of inequalities obtained by extending minimal covers is independent of the extension method: Let C be a minimal cover and $j_{\max} = \arg\max_{j \in C} w(C)$ and $i_{\min} = \arg\min_{i \in E(C) \setminus \overline{E}(C)} w(i)$. Hence, $C' = C \cup \{i_{\min}\} \setminus \{j_{\max}\}$ is a minimal cover with $|C'| = |C|$ and $\overline{E}(C') = \overline{E}(C)$. Yet, for the recoverable robust knapsack problem this is not the case: In the instance described in Table 2 with 6 items $\{1, \ldots, 6\}$ and two scenarios S_a and S_b and $k = 1$, the set $\overline{C} = \{1, 2, 4, 5\}$ is a minimal cover. If we extend $\overline{C}$ by the second method according to scenario S_a, $E^{S_a}(\overline{C}) = \{1, 2, 3, 4, 5, 6\}$ with $w^{S_a}(3) < w^{S_a}(4)$. But the set $C' = \{1, 2, 3, 5\}$ is not a minimal cover, since $w^{S_b}(C') - w^{S_b}(\max, C', 1) - w^{S_b}(\min, C', 1) > 3$. Also no other minimal cover induces a canonical extended cover inequality as strong as the one from $E^{S_a}(\overline{C})$.

$\mathcal{S} \backslash N$	1	2	3	4	5	6	c^S
S_a	2	2	3	4	8	9	6
S_b	10	1	5	2	1	0	3

Table 2: The table shows the knapsack capacity and the weights for each item according to the different scenarios.

In the following lemma we prove that the rrKP extended cover inequalities are feasible.

Lemma 3.3. *Let C be an rrKP cover, $E^S(C)$ an extension according to scenario S. Then*

$$\sum_{i \in E^S(C)} x_i \leq |C| - 1 \tag{12}$$

is a valid inequality for $\mathcal{K}_D(k)$.

Proof. Let us assume there exists an integer point $x \in \mathcal{K}_D(k)$, a minimal cover C and a scenario S such that

$$\sum_{i \in E^S(C)} x_i = |C|.$$

Let $T_x = \{i \in N \mid x_i = 1\}$ be the support of x and i_a an item in $T_x \cap E^S(C) \backslash C$ with minimum weight according to S. We furthermore order the items in C increasingly according to the weights of S, i.e., $w^S(i_1) \leq w^S(i_2) \leq \ldots \leq w^S(i_{|C|})$ and define $X = \{i_1, \ldots, i_{|C|-k-1}\}$. Since $w^S(i_a) \geq c^S - w^S(C) + w^S(\max, C, k+1) + 1$, $w^S(i_a) \geq w^S(\max, C, k+2) - w^S(\max, C, k+1)$ and $w^S(X \cup \{i_{|C|-k}\}) \geq c^S + 1$,

$$\begin{aligned} w^S(T_x) - w^S(\max, T_x, k) &= \min_{\substack{K \subseteq T_x \\ |K| \leq k}} w^S(T_x \backslash K) \\ &\geq w^S(X) + \min\{w^S(i_{|C|-k}), w^S(i_a)\} \\ &\geq c^S + 1. \end{aligned}$$

This is a contradiction to the feasibility of x. □

Separation of the rrKP extended cover inequalities for $\mathcal{S}_D$ The separation problem is to find an inequality, that is violated by a given non-integer point, or prove that non exists. In our case, we are interested in finding for a given fractional solution $x^* \in [0,1]^n$ an rrKP extended cover inequality. Since the definition of a cover is based on the consideration of a single scenario or the first stage weight set, we can separately consider each scenario and hence are interested in finding an rrKP extended cover inequality according to one scenario $S \in \mathcal{S}_D$ or the first stage weight constraint. For simplicity we drop the S on the variables, the weights and the knapsack capacity and set $k = 0$ when considering the first stage weight constraint.

Our first approach uses integer programming and is based on the following observation: In order to find a violated extended cover inequality, it suffices to determine the items which are part of the cover and not recovered (i.e., not deleted to satisfy the capacity constraint). We call those items the *heart of the cover*. In a second step all other items which exceed the weight of any item in the heard of the cover, can be added to form possibly an extended rrKP cover. This is the case, if more than k items are added to the heart. Although those extra items are fixed as soon as the heart of a cover is known, we introduce an integer program determine both sets for two reasons: first the number of additional items is crucial for the detection of a rrKP cover and secondly even for deterministic knapsack instances a fractional point x^* may not violate a cover inequality but an extended cover inequality. To this end we introduce two different binary variables y_i and z_i for each item $i \in N$. The variable y_i determines whether item i is part of the heart of the cover and z_i whether item i is added in the second step. In order to define a cover the inequality $\sum_{i \in N} w_i y_i \geq c+1$ has to be obeyed by y. An item is added as extension of the heart ($z_i = 1$), if its weight exceeds the weight of every item in the heart. Note, that if an item with a sufficient large weight is added to the recovery or the extension, all items with larger weights may also be added to the extension as they are exchangeable with the first one. In order to efficiently implement this condition, we group the items according to their weights: Let $0 \leq w_{i_1} < w_{i_2} < \ldots w_{i_\theta} \leq c$ be an ordering of all different item weights occurring in the scenario and $T := \{1, \ldots, \theta\}$. We define $N(t) := \{j \in N : w_j = w_{i_t}\}$ for all $t \in T$. Then $z_i \leq z_j$ is valid for all $i \in N(t)$, $j \in N(t+1)$, $t = 1, \ldots, \theta - 1$. Hence, we obtain the following ILP

$$\max \sum_{j \in N} (x_j^* - 1) y_j + \sum_{j \in N} x_j^* z_j - k \tag{13}$$

$$\sum_{j \in N} w_j y_j \geq c + 1 \tag{14}$$

$$y_j + z_j \leq 1 \quad \forall j \in N \tag{15}$$

$$z_i \leq z_j \quad \forall i \in N(t), j \in N(t+1), t \in T \tag{16}$$

$$y_j, z_j \in \{0,1\} \quad \forall j \in N. \tag{17}$$

An optimal solution defines a violated rrKP extended cover inequality, if the value of the objective function is great than -1. Note that in this case more than k items are added to the heart of the cover.

Based on this integer formulation and dynamic programming we will finally introduce a pseudo-polynomial algorithm to solve the separation problem for the extended cover inequalities. We define $U = \cup_{i=1}^n \{w_i\}$ and $D = \sum_{i=1}^n w_i$. For all $t = 1, \ldots, n$, $d = 0, \ldots, D$ and $\omega \in U$ we solve the problem to find a part of the heart of a cover and added items within the set $\{1, \ldots, t\}$ such that the violation of x^* is maximizes, the weight of the heart equals d, their maximum weight is below ω and the weight of all items added in the extension are greater or equal than ω. More formally

we consider the following function

$$
\begin{aligned}
f_\omega(t,d) = \max \sum_{i=1}^{t} (x_i^* - 1) y_i + \sum_{i=1}^{t} x_i^* z_i & \\
\sum_{i=1}^{t} w_i y_i &= d \\
y_i + z_i &\leq 1 && \forall i = 1, \ldots, t \\
z_i &= 0 && \forall i : w_i < \omega \\
y_i &= 0 && \forall i : w_i > \omega \\
y_i, z_i &\in \{0,1\} && \forall i = 1, \ldots, t.
\end{aligned}
$$

The optimal solution to the separation problem is given by

$$\max_{\substack{\omega \in U \\ d \geq c+1}} f_\omega(n,d)$$

and the function $f_\omega(1,d)$ can easily be solved via three case distinctions:

Case 1: if $w_1 = d$ and $\omega > d$, $f_\omega(1,d) = (x_1^* - 1)$

Case 2: if $d = 0$ and $w_1 \geq \omega$, $f_\omega(1,d) = x_1^*$

Case 3: otherwise, $f_\omega(1,d) = -\infty$.

For all other combinations of $t \geq 2$, $d = 0, \ldots, D$ and $\omega \in U$, the general recursive formula referring to the three cases above holds

$$
\begin{aligned}
f_\omega(t,d) = \max\{ & f_\omega(t-1,d), \\
& f_\omega(t-1,d-w_t) + (x_t^* - 1) && \text{if } w_t \leq \omega, \\
& f_\omega(t-1,d) + x_t^* && \text{if } w_t \geq \omega\}.
\end{aligned}
$$

In other words we decide, whether t is not in the heart and not added to the extension, is part of the heart of the cover or is added to the extension taking into account the weight bound imposed by ω. Obviously we can construct via an optimal solution of $f_\omega(t-1,d)$, $f_\omega(t-1,d-w_t)$, and $f_\omega(t-1,d)$ three feasible solutions for $f_\omega(t,d)$. Note that the recursion formula is valid, since otherwise we obtain a contradiction to the optimality of $f_\omega(1,d')$. The run-time of this approach is in $\mathcal{O}(D \cdot n^2)$, since $|U| \leq n$.

4 Computations

In this section we present computational results on the recoverable robust knapsack problem. First, we investigate the gain of recovery, i. e., the increase in the objective value of an optimal solution obtained by allowing recovery. Second, we study computationally the first closure of the relaxed recoverable robust knapsack problem with respect to the class of rrKP extended cover inequalities and different parameter settings.

Problem instances The considered rrKP instances are slight modifications of multi-dimensional knapsack instances taken from the ORLIB [6, 8, 5] created by Chu and Beasley. The original problem files are named mknapcb1.txt to mknapcb9.txt and contain 30 test instances each. For our studies we use only a third of them, namely those instances whose (knapsack capacity) tightness ratio is 0.50, which still provides us with a set of 90 instances. The number of items is 100, 250, or 500. The number of knapsacks is 5, 10, or 30. For each instance, the first knapsack is treated as first stage, and each remaining knapsack as individual discrete scenario. For each item, the objective function coefficient of the corresponding multi-dimensional knapsack instance is scaled by 0.7 and used as first stage profit. Each scenario profits is also determined by scaling the objective function coefficient but the scaling factor is uniformly random generated in the range from 0.2 to 0.4. Thus, the scenario profits range from 29% to 57% of the corresponding first stage profits. In summary, we collected a meaningful test set consisting of well-known heterogenous instances which is representative to recoverable robust knapsack problems which generalize (robust) multi-dimensional knapsacks problems.

Setting We implemented the ILP1 formulation of the recoverable robust knapsack problem in C++ using Scip 1.2.0 [1, 2] as branch-and-cut framework and IBM ILOG Cplex 12.1 [13] as underlying LP solver. Using the callback functionality of Scip, we added a separator for rrKP extended cover inequalities which separates violated cuts exactly by solving the ILP formulation (13)-(17). The separator is called at the root node only. In our first study concerning the gain of recovery the separator is turned off.

The computations were carried out on a Linux machine with 2.93 GHz Intel Xeon W3540 CPU and 12 GB RAM. A time limit of 1 hour was set for solving each problem instance. All other solver settings were left at their defaults.

4.1 Gain of recovery

In our first computational study we investigate the gain of recovery. Therefore we limit the allowed recovery by the parameters k and ℓ where k (ℓ) determines how many items chosen (not chosen) in the first stage may be removed (added) in the scenario. The values of these parameters are given as fraction of the number of items, e. g., $k = 0.25$ means that 25% of the first stage selected items may be removed in each scenario.

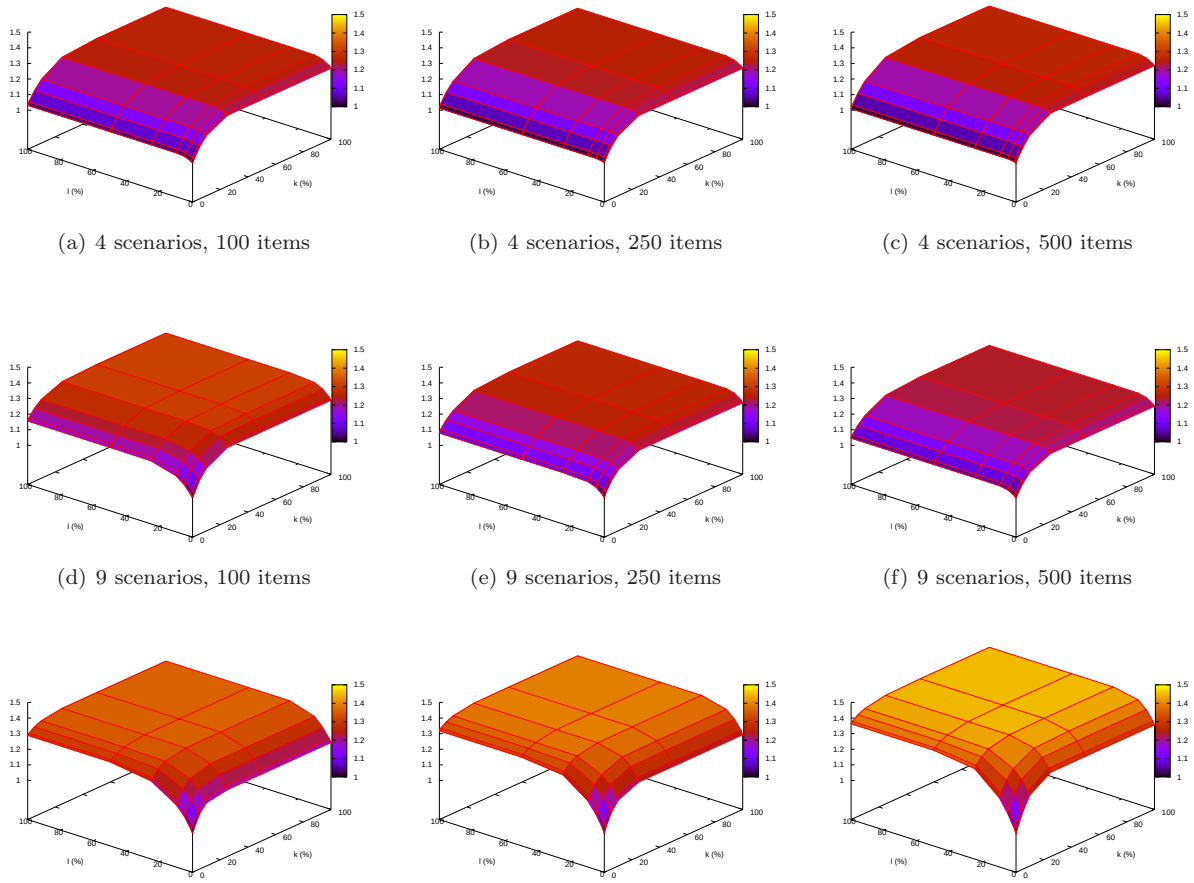
The setting $k = \ell = 0$ is the standard robust setting where no recovery is allowed and all scenarios must be fulfilled simultaneously which is equivalent to the classical multi-dimensional knapsack problem. For given values of k and ℓ we compare the objective value of the optimal solution (resp. best known solution within the time limit as the achieved optimality gaps are very small) with the $k = \ell = 0$ setting to evaluate the gain of recovery.

Table 3 (on page 15) states the computational results of our study for selected values of k and ℓ. A graphical representation is given in Figure 1. Only average values are shown for groups of 10 instances with the same number of items and scenarios. All values are normalized to the corresponding $k = \ell = 0$ setting.

Fixing the number of items we observe a rise in the gain of recovery when the number of scenarios increases (e. g., compare Figures 1(a), 1(d), and 1(g)). This can be explained as the non-recovery solution of these instances is more conservative due to the higher number of scenarios. Therefore recovery allows a larger gain. Considering 4 (9, 29) scenarios an additional gain of 26% (31%, 45%) compared to the $k = \ell = 0$ setting can be achieved. Unfortunately, fixing the number of scenarios and varying the number of items does not give us a clear correlation. For the 100 (250, 500) item instances we can achieve an additional gain up to 36% (39%, 45%). Next, we observe that increasing the parameter k, i. e., allowing more items to be removed in each scenario, clearly leads to an increase in the gain of recovery. This is plausible as large items violating a scenario

	#scenarios	4			9			29		
	#items	100	250	500	100	250	500	100	250	500
k	ℓ									
0.00	0.00	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	0.01	1.0093	1.0076	1.0059	1.0185	1.0195	1.0133	1.0233	1.0295	1.0266
	0.05	1.0268	1.0135	1.0104	1.0699	1.0542	1.0370	1.0933	1.1057	1.1112
	0.10	1.0319	1.0137	1.0104	1.1099	1.0726	1.0463	1.1504	1.1752	1.1929
	0.25	1.0320	1.0137	1.0104	1.1545	1.0798	1.0474	1.2508	1.2899	1.3295
	0.50	1.0320	1.0137	1.0104	1.1563	1.0797	1.0474	1.2888	1.3183	1.3635
	1.00	1.0320	1.0137	1.0104	1.1568	1.0799	1.0470	1.2901	1.3187	1.3637
0.01	0.00	1.0210	1.0248	1.0206	1.0244	1.0286	1.0212	1.0267	1.0323	1.0285
	0.01	1.0297	1.0314	1.0260	1.0421	1.0455	1.0329	1.0496	1.0593	1.0539
	0.05	1.0454	1.0363	1.0298	1.0907	1.0759	1.0540	1.1134	1.1332	1.1360
	0.10	1.0495	1.0365	1.0298	1.1287	1.0922	1.0625	1.1718	1.1998	1.2157
	0.25	1.0496	1.0365	1.0298	1.1700	1.0983	1.0632	1.2670	1.3068	1.3446
	0.50	1.0496	1.0365	1.0298	1.1723	1.0982	1.0631	1.2969	1.3299	1.3743
	1.00	1.0496	1.0365	1.0299	1.1720	1.0980	1.0632	1.2983	1.3304	1.3745
0.05	0.00	1.0840	1.0856	1.0826	1.0962	1.0905	1.0773	1.1054	1.1135	1.1224
	0.01	1.0906	1.0908	1.0867	1.1103	1.1037	1.0868	1.1230	1.1368	1.1456
	0.05	1.1034	1.0943	1.0888	1.1509	1.1273	1.1034	1.1805	1.1993	1.2177
	0.10	1.1056	1.0943	1.0888	1.1840	1.1390	1.1082	1.2317	1.2554	1.2829
	0.25	1.1056	1.0943	1.0888	1.2163	1.1421	1.1083	1.3073	1.3447	1.3848
	0.50	1.1056	1.0943	1.0888	1.2165	1.1421	1.1083	1.3216	1.3541	1.4002
	1.00	1.1056	1.0943	1.0888	1.2168	1.1422	1.1083	1.3231	1.3545	1.4004
0.10	0.00	1.1359	1.1367	1.1404	1.1533	1.1417	1.1267	1.1557	1.1725	1.2064
	0.01	1.1421	1.1415	1.1440	1.1663	1.1531	1.1347	1.1724	1.1945	1.2264
	0.05	1.1550	1.1450	1.1458	1.2024	1.1726	1.1481	1.2279	1.2503	1.2858
	0.10	1.1570	1.1450	1.1458	1.2303	1.1805	1.1503	1.2749	1.2998	1.3383
	0.25	1.1570	1.1450	1.1457	1.2542	1.1823	1.1502	1.3361	1.3702	1.4153
	0.50	1.1570	1.1450	1.1457	1.2544	1.1821	1.1501	1.3419	1.3725	1.4201
	1.00	1.1570	1.1450	1.1458	1.2542	1.1821	1.1502	1.3420	1.3731	1.4203
0.25	0.00	1.2047	1.2049	1.2191	1.2246	1.2074	1.1863	1.1783	1.2288	1.2909
	0.01	1.2146	1.2150	1.2271	1.2387	1.2214	1.1976	1.1983	1.2519	1.3112
	0.05	1.2327	1.2284	1.2375	1.2747	1.2424	1.2160	1.2566	1.3090	1.3664
	0.10	1.2384	1.2299	1.2379	1.2963	1.2494	1.2193	1.3111	1.3543	1.4100
	0.25	1.2386	1.2299	1.2378	1.3067	1.2501	1.2194	1.3610	1.3940	1.4450
	0.50	1.2386	1.2299	1.2379	1.3067	1.2502	1.2194	1.3611	1.3940	1.4450
	1.00	1.2386	1.2299	1.2379	1.3068	1.2502	1.2194	1.3613	1.3941	1.4451
0.50	0.00	1.2056	1.2054	1.2217	1.2250	1.2088	1.1869	1.1787	1.2287	1.2917
	0.01	1.2170	1.2168	1.2317	1.2401	1.2232	1.1992	1.1986	1.2522	1.3127
	0.05	1.2427	1.2393	1.2539	1.2788	1.2521	1.2258	1.2568	1.3099	1.3718
	0.10	1.2555	1.2497	1.2641	1.3043	1.2651	1.2370	1.3110	1.3573	1.4187
	0.25	1.2575	1.2508	1.2646	1.3164	1.2666	1.2375	1.3617	1.3958	1.4501
	0.50	1.2575	1.2508	1.2647	1.3164	1.2667	1.2375	1.3618	1.3958	1.4501
	1.00	1.2575	1.2508	1.2647	1.3164	1.2664	1.2374	1.3618	1.3959	1.4501
1.00	0.00	1.2056	1.2054	1.2218	1.2254	1.2087	1.1871	1.1793	1.2293	1.2918
	0.01	1.2170	1.2169	1.2317	1.2399	1.2235	1.1995	1.1994	1.2529	1.3131
	0.05	1.2427	1.2393	1.2540	1.2787	1.2525	1.2259	1.2582	1.3103	1.3720
	0.10	1.2555	1.2497	1.2642	1.3042	1.2652	1.2370	1.3115	1.3573	1.4188
	0.25	1.2575	1.2508	1.2648	1.3164	1.2668	1.2376	1.3617	1.3959	1.4501
	0.50	1.2575	1.2508	1.2648	1.3163	1.2667	1.2375	1.3617	1.3959	1.4501
	1.00	1.2575	1.2508	1.2648	1.3164	1.2668	1.2375	1.3619	1.3959	1.4502

Table 3: Gain of recovery for selected values of k and ℓ. Averages are shown for #scenarios=4, 9, 29 and #items=100, 250, 500. All values are normalized to $k = \ell = 0$.



(a) 4 scenarios, 100 items (b) 4 scenarios, 250 items (c) 4 scenarios, 500 items

(d) 9 scenarios, 100 items (e) 9 scenarios, 250 items (f) 9 scenarios, 500 items

(g) 29 scenarios, 100 items (h) 29 scenarios, 250 items (i) 29 scenarios, 500 items

Figure 1: Gain of recovery for selected values of k and ℓ. Averages are shown for #scenarios=4, 9, 29 and #items=100, 250, 500. All values are normalized to $k = \ell = 0$.

#scenarios	4			9			29			
#items	100	250	500	100	250	500	100	250	500	geom.
k										mean
0.00	5,27	3,02	1,53	8,54	1,98	0,63	0,76	3,69	1,61	2,19
0.01	8,60	2,00	0,62	9,09	3,46	2,15	1,11	3,28	5,37	2,89
0.05	7,47	1,25	0,07	5,22	1,87	0,08	1,57	6,42	7,77	1,49
0.10	6,99	1,52	1,20	6,09	1,64	3,24	1,62	3,65	5,89	2,89
0.25	5,60	5,21	2,42	3,71	3,11	4,95	3,04	4,36	6,03	4,09
0.50	7,87	5,61	8,20	7,78	10,16	9,29	13,28	16,04	23,90	10,34
1.00	7,74	4,91	7,25	7,40	10,00	9,05	12,70	14,60	21,56	9,67
geom. mean	6,98	2,88	1,40	6,57	3,52	2,03	2,72	6,03	7,42	

Table 4: Gap closed (%) for selected values of k. Averages are shown for #scenarios=4, 9, 29 and #items=100, 250, 500 and setting (iii) (exact separation of rrKP extended cover inequalities)

knapsack constraint may more likely be removed. Finally, we observe an increase in ℓ seems to have not a reasonable impact on the gain of recovery.

In summary, allowing recovery yields a gain of up to 45% (e. g. for $k = \ell = 50\%$). Even a more restricted setting as $k = \ell = 10\%$ gives still a gain of recovery in the range from 15% to 33%.

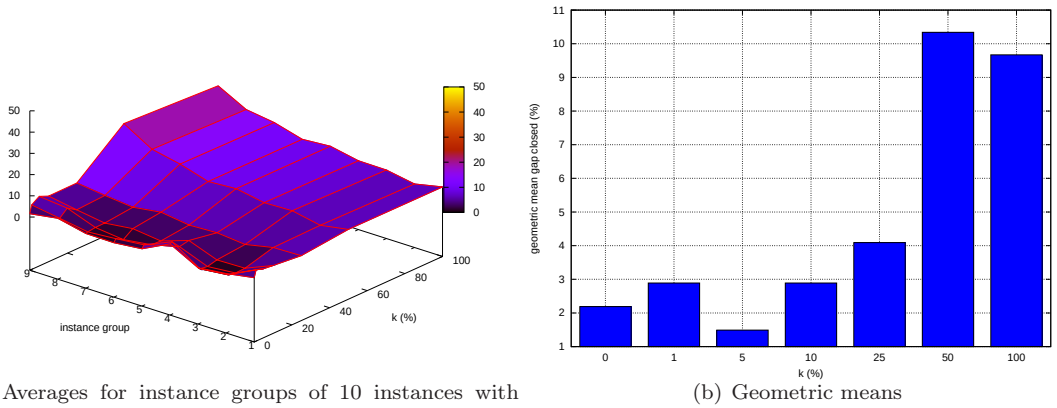
4.2 First rrKP extended cover closure

In the following we present the results of our second study where we investigate the effectiveness of rrKP extended cover inequalities using the canonical extension (11). Therefore we implemented the ILP formulation (13)-(17) of the corresponding separation problem to separate violated rrKP extended cover inequalities exactly (but still with an overall 1 hour time limit per instance). Whenever our separator is called the first stage knapsack is checked. If no violated extended cover is found, all scenarios are tested beginning with the last scenario which provided a violated cut. As soon as a violated rrKP extended cover inequality has been determined, it is added to the LP and the separation round is aborted. Hence, we separate at most one cut per call in this study. Only the root node of the recoverable robust knapsack problem is solved in this study.

We consider two different settings. First we solve the LP-relaxation without any (external or internal) separators to determine the integrality gap of the instance. Second, we solve the LP-relaxation with exact separation of violated rrKP extended cover inequalities and evaluate integrality gap to determine the percental gap closed we achieved.

In Table 4 (on page 17) the achieved gap closed is shown for selected values of k (violated rrKP extended cover inequalities do not depend on ℓ). As before results stated are averages for groups of 10 instances with the same number of items and scenarios. In addition, the geometric means are given for fixing either the value of k or the number of items and scenarios. Figure 2(a) shows the average gap closed(%) for each instance group and selected values of k. The instance groups are numbered 1 to 9 corresponding to the columns in In Table 4, e. g., instances in group 1 have 4 scenarios and 100 items, instances in group 9 have 29 scenarios and 500 items. In Figure 2(b) the geometric means of the group averages are shown depending on k.

We observe that the geometric mean of the integrality gap closed lies in the range from 1.49% ($k = 5\%$) to 10.34% ($k = 50\%$). Considering all instances with 4 (9, 29) scenarios it ranges from 0.07% (0.08%, 0.76%) to 8.60% (10.16%, 23.90%). This suggests that an increase in the number of scenarios may result in a larger gap closed. Unfortunately, there is no clear dependency. For example, the average gap closed for instances with 100 items and $k = 10\%$ are 6.99%, 6.09%, and 1.62% for 4, 9, and 29 scenarios. But considering the instances with 250 items they are 1.52%, 1.64%, and 3.65%. Other examples exist which are neither monotonic increasing nor decreasing.



(a) Averages for instance groups of 10 instances with same #scenarios and #items

(b) Geometric means

Figure 2: Integrality Gap closed (%) by separating violated rrKP extended cover inequalities

Considering all instances with 100 (250, 500) items the integrality gap closed ranges from 0.76% (1.25%, 0.07%) to 13.28% (16.04%, 23.90%). Again, there is no clear trend when fixing k or the number of scenarios.

In summary, this study shows that by adding all violated rrKP extended cover inequalities the integrality gap is always lowered. The best achievement has been a gap closed by 23.90%. In more than 50% of all considered settings (#scenarios, #items, k) the integrality gap was closed by more than 5%. In less than 27% it is closed by less that 2%. In addition, the overall computational time spend for one run of this study was less than half an hour. Hence, the separation of violated rrKP extended cover cuts has a high potential to tighten the linear relaxation of the recoverable robust knapsack problem and to speed-up the solving process significantly.

5 Conclusion

In this paper, we considered the recoverable robust knapsack problem with discrete scenarios. We showed that for a fixed number of discrete scenarios this recoverable robust knapsack problem is weakly **NP**-complete and any such instance can be solved in pseudo-polynomial time by a dynamic program. If the number of discrete scenarios is part of the input, the problem is strongly **NP**-complete and in special cases not approximable in polynomial time, unless $\mathbf{P} = \mathbf{NP}$.

Further, we investigated the related recoverable robust knapsack polyhedron and introduced the class of rrKP cover inequalities which generalize their well-known counter-part for standard knapsack problems. Next we extended these covers to the class of rrKP extended cover inequalities. In addition to a canonical extension we present a more complex extension exploiting the scenario-based structure of the problem yielding stronger valid inequalities. As the robust knapsack problem is a special case of the recoverable robust knapsack problem this second extension produces strong valid inequalities for the robust knapsack polytope as well.

Finally, we presented two extensive computational studies. Our test set consists of 90 representative problem instances taken from the well-known ORLIB. We compare a wide range of different parameter settings generating several thousand individual test runs. First, we studied the gain of recovery, i. e. the increase in the objective function caused by allowing recovery. There a gain of up to 45% could be achieved. Second, we investigated the first closure with respect to the class of rrKP extended cover inequalities. We observed that the integrality gap could be closed by more than 5% in most of times. The best average gap closed we achieved was 23.90%.

In summary, we successfully transfered several classical results for the (robust) knapsack problem to its recoverable robust version.

In future work, the polyhedral structure of the recoverable robust knapsack polytope should be further investigated to identify new classes of valid or facet-defining inequalities. In addition, further computational studies should be carried out evaluating the overall impact of those classes of inequalities on the solving process.

References

[1] T. Achterberg. SCIP - a framework to integrate Constraint and Mixed Integer Programming. Technical Report 04-19, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2004.

[2] T. Achterberg, T. Berthold, T. Koch, A. Martin, and K. Wolter. SCIP (Solving Constraint Integer Programs), 2009. `http://scip.zib.de/`.

[3] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versons of some combinatorial optimization problems: a survey. 2007. `http://hal.archives-ouvertes.fr/docs/00/15/86/52/PDF/AN7LAMSADE_1-32.pdf`.

[4] E. Balas. Facets of the knapsack polytope. *Math. Program.*, 8:146–164, 1975.

[5] J.E. Beasley. OR-Library. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[6] J.E. Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990.

[7] R. Bellman. Notes on the theory of dynamic programming iv - maximization over discrete sets. *Naval Research Logistics Quarterly*, 3:67–70, 1956.

[8] P.C. Chu and J.E. Beasley. A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, 4:63–86, 1998.

[9] H. Crowder, E. Johnson, and M. Padberg. Large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[10] G.B. Dantzig. Discrete variable extremum problems. *Operations Research*, 1957:266–277, 5.

[11] C.E. Ferreira, A. Martin, and R. Weismantel. Solving multiple knapsack problems by cutting planes. *SIAM J. Opt.*, 6:858–877, 1996.

[12] H. Iida. A note on the max-min 0-1 knapsack problem. *Journal of Combinatorial Optimization*, 3:89–94, 1999.

[13] ILOG. CPLEX version 12.1, 2009. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[14] R. Kalai and D. Vanderpooten. Lexicographic α-robust knapsack problems: complexity results. *IEEE International Conference on Services Systems and Service Managment*, pages 1103 – 1107, 2006.

[15] K. Kaparis and A. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathe. Program.*, 124(1-2):69–91, 2010.

[16] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[17] H. Kellerer, U. Pfeschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[18] Ch. Liebchen, M. E. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 1–27. Springer, 2009.

[19] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.

[20] M. Padberg. (1,k)-configurations and facets for packing problems. *Mathe. Program.*, 18:94–99, 1980.

[21] R. Weismantel. On the 0-1 knapsack polytope. *Math. Program.*, 77:49–68, 1997.

[22] L. Wolsey. Faces for a linear inequality in 0-1 variables. *Math. Program.*, 8:165–178, 1975.

[23] G. Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 44:407–415, 1996.