Büther, Marcel; Briskorn, Dirk

Reducing the 0-1 knapsack problem with a single continuous variable to the standard 0-1 knapsack problem

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 629

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 629

Reducing the 0-1 Knapsack Problem with a Single Continuous Variable to the Standard 0-1 Knapsack Problem

Marcel Büther, Dirk Briskorn

September 2007

Marcel Büther, Dirk Briskorn
Christian-Albrechts-Universität zu Kiel,
Institut für Betriebswirtschaftslehre,
Olshausenstr. 40, 24098 Kiel, Germany,
http://www.bwl.uni-kiel.de/bwlinstitute/Prod
buether@bwl.uni-kiel.de, briskorn@bwl.uni-kiel.de

Abstract

The 0-1 knapsack problem with a single continuous variable (KPC) is a natural extension of the binary knapsack problem (KP), where the capacity is not any longer fixed but can be extended which is expressed by a continuous variable. This variable might be unbounded or restricted by a lower or upper bound, respectively. This paper concerns techniques in order to reduce several variants of KPC to KP which enables us to employ approaches for KP. We propose both, an equivalent reformulation and a heuristic one bringing along less computational effort. We show that the heuristic reformulation can be customized in order to provide solutions having an objective value arbitrarily close to the one of the original problem.

Keywords: 0-1 knapsack problem with a single continuous variable, binary knapsack problem, mixed integer programming, reformulation, lower bound, binary representation

1 Introduction

The 0-1 knapsack problem with a single continuous variable, KPC for short, is a natural extension of the binary knapsack problem (KP), a well–known combinatorial optimization problem with applications in production, logistics, and distribution planning. The KP is to choose items in order to maximize profit without exceeding a given capacity. While the capacity can not be influenced according to the KP the KPC considers the opportunity to extend or reduce, respectively, the available capacity. Extending capacity reduces the profit while reducing capacity increases the profit. The KP is well–known to be NP-hard in the weak sense, meaning that it can be solved in pseudo-polynomial time, see Garey and Johnson [1] and Pisinger [9]. NP-hardness of KPC follows straightforwardly.

Of course, there is a huge amount of real life applications for the KP, e.g. the optimization of inventory policies, see Gorman and Ahire [2]. But the main importance is the use as a "building block", occurring as a subproblem of a more complex problem. The same yields for the KPC as well. In addition, a direct application for the KPC is the determination of the optimal choice of investment projects with a given budget, where the budget can be widened on credit.

The KP has been studied by numerous researchers during the last decades, see Kellerer et al. [3] for example. Martello et al. [5] develop an approach called `combo`, based on a combination of dynamic programming with tight bounds. Although published in 1999 it still seems to be the state-of-the-art, see Martello et al. [6] and Pisinger [9]. Contrarily, very few papers concerning the KPC can be found. One proposal has been presented recently in Nauss [8], which presents a branch&bound approach adopted from the KP. Another work is stated in Marchand and Wolsey [4] along with Wolsey [10] based on branch&cut: The linear programming relaxation of KPC is strengthened by adding additional constraints. The resulting upper bound is then used in a branch&bound procedure. However, a major drawback is the exponential number of cuts.

The purpose of our work is to reduce the KPC to the KP which enables us to solve it employing the algorithm `combo`. The exposition of our work is as follows: In section 2 we present the mixed integer formulation and develop several properties of solutions. The approaches for the one-sided and two-sided limitation of the capacity are given in section 3 and section 4, respectively. In section 5 we provide some computational results and section 6 finally concludes the paper.

2 Model Formulation and First Insights

Given a set $N = \{1, \ldots, n\}$ of items with profit p_j and weight w_j for each item $j \in N$, the objective is to maximize the profit sum of the chosen items regarding the capacity b, $b > 0$, of the knapsack. To this end, binary decision variable x_j, $j \in N$, is used.

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

We reasonably restrict the parameters: $p_j > 0$ and $w_j > 0$, $j \in N$. Moreover, in order to avoid trivial cases we assume $\sum_{j \in N} w_j > b$. According to the KPC the original capacity b, can be adjusted. Continuous variable s represents the amount of capacity which is added ($s > 0$) or removed ($s < 0$), respectively. The value per unit of flexible capacity is c, $c > 0$. Now, the problem can be stated mathematically as follows:

$$\max \sum_{j \in N} p_j x_j - c \cdot s \tag{1}$$

$$\text{s.t.} \sum_{j \in N} w_j x_j \leq b + s \tag{2}$$

$$x_j \in \{0, 1\} \qquad \forall\, j \in N \tag{3}$$

$$l \leq s \leq u \tag{4}$$

The objective function (1) is twofold. The first part represents the sum of chosen items' profits. The second part considers cost and sales when additional capacity is bought and superfluous capacity is sold, respectively. Constraint (2) ensures that the (modified) capacity limit of the knapsack is not exceeded. For short, a solution of the KPC can be expressed by the tuple (x, s), where x is the binary vector containing the values of x_j, $j \in N$, and s is the value of the corresponding capacity modification.

Before presenting reductions of KPC to KP for most cases, we introduce some definitions and line out basic properties of solutions to the KPC, first.

We define the following sets of items:

$$\begin{aligned} N^+ &:= \{j \in N \mid p_j > c \cdot w_j\} \\ N^= &:= \{j \in N \mid p_j = c \cdot w_j\} \\ N^- &:= \{j \in N \mid p_j < c \cdot w_j\} \end{aligned}$$

Obviously, N^+, $N^=$, and N^- form a partition of N. Accordingly, we define $W^+ := \sum_{j \in N^+} w_j$, $W^= := \sum_{j \in N^=} w_j$, $W^- := \sum_{j \in N^-} w_j$, and $W := W^+ + W^- + W^= = \sum_{j \in N} w_j$.

Based on the sets defined above we present several properties of solutions to the KPC in the following.

Lemma 1. *Given an optimal solution (x, s) with $\sum_{j \in N} w_j x_j \geq b + l$ constraint (2) is fulfilled with equality.*

Proof. Suppose $b + l \leq \sum_{j \in N} w_j x_j < b + s$. Then, (x, s') with $s' = \sum_{j \in N} w_j x_j - b < s$ is feasible and has objective value $\sum_{j \in N} p_j x_j - c \cdot s' > \sum_{j \in N} p_j x_j - c \cdot s$. Therefore, (x, s) can not be optimal. □

Thus, if the condition for lemma 1 is fulfilled, we can replace s by $\sum_{j\in N} w_j x_j - b$ in (1) and (4) and drop (2). After regrouping the objective and splitting (4) into two constraints, the problem can be stated as follows:

$$c \cdot b + \max \sum_{j\in N} (p_j - c \cdot w_j) x_j \tag{5}$$

$$\text{s.t.} \sum_{j\in N} w_j x_j \geq b + l \tag{6}$$

$$\sum_{j\in N} w_j x_j \leq b + u \tag{7}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in N \tag{8}$$

This problem, the KPC_{eq} for short, will be used as a building block in the following sections.

Lemma 2. *Given a solution (x, s) with $\sum_{j\in N} w_j x_j \geq b + l$ any solution (x', s') with $\sum_{j\in N} w_j x'_j \geq b + l$ where $x_j = x'_j$, $j \in N^- \cup N^+$, has the same objective value.*

Proof. Since x and x' contain identical items out of $N^- \cup N^+$ the following holds using lemma 1 and the corresponding KPC_{eq}:

$$\begin{aligned} & c \cdot b + \sum_{j\in N} (p_j - c \cdot w_j) x_j - \left(c \cdot b + \sum_{j\in N} (p_j - c \cdot w_j) x'_j \right) \\ = & \sum_{j\in N^=} (p_j - c \cdot w_j) x_j - \sum_{j\in N^=} (p_j - c \cdot w_j) x'_j \\ = & \; 0 - 0 = 0 \end{aligned}$$

□

Lemma 3. *If the optimal solution (x, s) contains at least one item out of N^- then $s - l < \min_{j\in N^-_*} w_j$ holds where $N^-_* = \{j \in N^- \mid x_j = 1\}$.*

Proof. Suppose there is an item with $w_j < s - l$, $j \in N^-_*$. Then, removing j from (x, s) yields a solution with $s > l$. Thus the condition for lemma 1 is still satisfied and the resulting objective value surpasses the one of (x, s) by $c \cdot w_j - p_j > 0$. □

Before dealing with the one-sided and the two-sided limitation of the capacity adjustment in the two following sections, we first highlight the special case, where s is not limited by any effective bound, e.g. $l \leq -b$ and $u \geq W - b$. Obviously, for each solution constraint (2) is tight and the KPC_{eq} can be applied. As both bounds are not restrictive, we can drop (6) and (7). Now, one optimal solution is obtained in $O(n)$ by setting $x_j = 1$ for all items $j \in N^+$ and $x_j = 0$ for all items $j \notin N^+$. Adding an arbitrary subset of items out of $N^=$ generates an other optimal solution, see lemma 2. An item of N^- will never be part of an optimal solution, since its objective coefficient is always smaller than zero. Hence, the number of optimal solutions equals $2^{|N^=|}$.

3 Capacity Adjustment with a Single Bound

3.1 Capacity Adjustment with an Upper Bound

For this case the upper bound is restrictive while the lower bound is not: $l \leq -b$ and $u < W - b$. We apply KPC_{eq} according to lemma 1 and drop (6). Thus, the KPC_{eq} is a standard KP:

$$c \cdot b + \max \sum_{j \in N} (p_j - cw_j)\, x_j \tag{9}$$

$$\text{s.t.} \sum_{j \in N} w_j x_j \;\leq\; b + u \tag{10}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in N \tag{11}$$

Obviously, regarding (9), only items of N^+ and $N^=$ will be in an optimal solution.

Lemma 4. *Each optimal solution has $x_j = 0$ for each $j \in N^-$ and there is at least one optimal solution where $x_j = 0$ for each $j \in N^=$.*

Proof. First, suppose an optimal solution (x, s) where $x_j = 1$ for any $j \in N^-$. Then, solution (x', s') with $x'_j = 0$, $x'_{j'} = x_{j'}$ for each $j' \in N$, $j' \neq j$, and $s' = \sum_{j' \in N} w_j x'_j - b$ has a larger objective value than (x, s) has.

Second, suppose an optimal solution (x, s) where $x_j = 1$ for any $j \in N^=$. Then, solution (x', s') with $x'_j = 0$, $x'_{j'} = x_{j'}$ for each $j' \in N$, $j' \neq j$, and $s' = \sum_{j' \in N} w_j x'_j - b$ must have the same objective value as (x, s) has, according to lemma 2. □

Note that according to lemma 4 we can fix $x_j = 0$ for each $j \in N^= \cup N^-$ before solving (9) to (11).

3.2 Capacity Adjustment with a Lower Bound

Here, s is only limited by a lower bound: $l > -b$ and $u \geq W - b$ and $s \geq l$ replaces (4). In the following, we first show how to reduce the number of variables. Afterwards, the optimal solution might be found. Otherwise, we reduce the KPC for this case to KP.

Lemma 5. *Each optimal solution has $x_j = 1$ for each $j \in N^+$ and there is at least one optimal solution where $x_j = 1$ for each $j \in N^=$.*

Proof. First, suppose an optimal solution (x, s) where $x_j = 0$ for any $j \in N^+$. Then, solution (x', s') with $x'_j = 1$, $x'_{j'} = x_{j'}$ for each $j' \in N$, $j' \neq j$, and $s' = \max\{l, \sum_{j' \in N} w_j x'_j - b\}$ must have a larger objective value than solution (x, s). Therefore, (x, s) can not be optimal.

Second, suppose an optimal solution (x, s) where $x_j = 0$ for any $j \in N^=$. Then, solution (x', s') with $x'_j = 1$, $x'_{j'} = x_{j'}$ for each $j' \in N$, $j' \neq j$, must have the same objective value as (x, s) has according to lemma 2. □

According to lemma 5 we first fix $x_j = 1$ for each $j \in N^+ \cup N^=$, which leads to an objective value of $P^{+=} = \sum_{j \in N^+ \cup N^=} p_j$. Then, if $W^+ + W^= - b = s \geq l$, an optimal solution can be constructed by additionally letting $x_j = 0$ for each $j \in N^-$. Otherwise, we solve the problem represented by the following mixed integer program where $b' = b - W^+ - W^=$, named reduced KPC, rKPC for short:

$$P^{+=} + \max \sum_{j \in N^-} p_j x_j - c \cdot s \tag{12}$$

$$\text{s.t.} \sum_{j \in N^-} w_j x_j \leq b' + s \tag{13}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in N^- \tag{14}$$

$$s \geq l \tag{15}$$

Regarding the structure of rKPC, the solution space can be split into two subsets S^l and S^+ of solutions:

- S^l contains all solutions where an capacity adjustment equaling the lower bound l suffices. Hence, $S^l = \{(x,s) \mid s = l\}$ and, obviously, $\sum_{j \in N} w_j x_j \leq b + l$ for each solution $(x,s) \in S^l$.
- S^+ contains all solutions where the used capacity is not lower than $b + l$. Hence, $S^+ = \left\{(x,s) \mid \sum_{j \in N} w_j x_j \geq b + l\right\}$.

Note that S^l and S^+ do not form a complete neither a disjunct partition of the feasible solutions to rKPC: $S^l \cup S^+$ does not contain each single solution (x,s) where $\sum_{j \in N} w_j x_j < b + s$ and $s > l$. Obviously, for each of these solutions a better solution (x,s') exists which justifies the reduction of solution space to $S^l \cup S^+$. Moreover, solutions having $\sum_{j \in N} w_j x_j = b + l$ are contained in both parts due to formal reasons when constructing standard KPs below. Next, we propose how to find an optimal solution for both subsets by solving KPs. Choosing the better solution leads to the optimal solution of the KPC.

First, restricting solutions to S^l, we fix $s = l$ and trivially, terms (16) to (18) form a standard KP.

$$P^{+=} - c \cdot l + \max \sum_{j \in N^-} p_j x_j \tag{16}$$

$$\text{s.t.} \sum_{j \in N^-} w_j x_j \leq b' + l \tag{17}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in N^- \tag{18}$$

Next, restricting the underlying rKPC to S^+, we can use the KPC_{eq} according to lemma 1. Regrouping the objective function we yield:

$$P^{+=} - c \cdot b' + \max \sum_{j \in N^-} (p_j - c w_j)\, x_j \tag{19}$$

$$\text{s.t.} \sum_{j \in N^-} w_j x_j \geq b' + l \tag{20}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in N^- \tag{21}$$

Then, we introduce binary variable $x'_j = 1 - x_j$ and replace x_j by $1 - x'_j$:

$$P^{+=} - c \cdot b' + \sum_{j \in N^-} (p_j - cw_j) + \max \sum_{j \in N^-} (cw_j - p_j)x'_j \tag{22}$$

$$\text{s.t.} \sum_{j \in N^-} w_j x'_j \leq W^- - b' - l \tag{23}$$

$$x'_j \in \{0,1\} \qquad j \in N^- \tag{24}$$

Obviously, this reformulation is a standard KP. It can be interpreted as a problem complementary to the original one. Based on a knapsack containing all items, binary variable $x'_j = 1 - x_j$ is equal to 1 if and only if item j is removed from the knapsack. A knapsack containing all items corresponds to a feasible solution in S^+ having objective value $c \cdot b + \sum_{j \in N} (p_j - c \cdot w_j)$. The optimization problem is to remove items from the knapsack such that capacity of at least $b + l$ is used and profit is maximized.

Summarizing, given a lower bound for s we first check in $O(n)$ time whether an optimal solution in line with lemma 5 exists. If not, then we reduce the KPC to rKPC, which can be optimized by solving two standard KPs: After obtaining optimal solutions of S^l and S^+ we identify the better one as the optimal solution to rKPC and adding $N^= \cup N^+$ we obtain an optimal solution to KPC.

4 Capacity Adjustment with Lower and Upper Bound

Assume, that the possible capacity adjustment s is now limited in both directions.

$$-b < l \leq s \leq u < W - b$$

Note that a feasible solution can still have a capacity consumption, which is less than $b + l$. In general our solution approaches introduced in section 3 can not be used directly. Therefore, we first propose an approach to use the techniques presented in section 3 for two special cases. Second, we reformulate the problem as a single standard KP accepting a reduction of the solution space.

4.1 Employing Relaxations

We propose to solve two relaxations of (1) to (4) which correspond to the problems treated in section 3.1 and 3.2. First, we drop the capacity adjustment's lower bound and obtain the KPC where s is only limited by u, namely KPC$_{up}$. If the optimal solution has $s \geq l$ the optimal solution to the original problem is reached. Otherwise we can vary the optimal solution to the relaxation without changing its objective value by adding items out of $N^=$ according to lemma 2 as long as s does not overshoot u. If no feasible solution can be reached we solve the relaxation where the capacity adjustment's upper bound is dropped, namely KPC$_{lo}$. If its solution implies $s \leq u$ we have found an optimal solution to the original problem. Again, we can modify an optimal solution by removing items out of $N^=$ without changing the objective value, as long as s does not undershoot l.

Theorem 1. *If*

1i $b + l \leq W^+ \leq b + u$ *or*

1ii $W^+ < b + l$ *and* $b + l \leq W^+ + W^= \leq b + u$ *or*

1iii $W^+ < b + l$, $W^+ + W^= \geq b + u$, *and* $\max_{j \in N^=} w_j \leq u - l$

an optimal solution to KPC can be found in $O(n)$ *time.*

Proof.

1i Setting $x_j = 1$ for $j \in N^+$ yields an optimal solution to KPC_{up} due to lemma 4. Obviously, it is feasible and, therefore, optimal to the KPC at hand, as well.

1ii Here, by setting $x_j = 1$ for $j \in N^+ \cup N^=$ an optimal solution to KPC_{up} is obtained due to lemmas 2 and 4. Obviously, it is feasible and, therefore, optimal to the KPC under consideration, as well.

1iii By setting $x_j = 1$ for $j \in N^+$ an optimal solution (x, s) to KPC_{up} is found due to lemma 4. Note, that $s < l$ and, therefore, (x, s) is not feasible to the KPC at hand. Adding items belonging to set $N^=$ in arbitrary order we preserve optimality of (x, s) according to KPC_{up}, see lemma 2. Since $\max_{j \in N^=} w_j \leq u - l$ we reach at least one solution (x', s') to KPC_{up} with $s' \geq l$. Solution (x', s') is optimal to KPC_{up} and feasible to KPC.

□

Theorem 2. *If*

2i $W^+ > b + u$ *and* $\max_{j \in N^+} w_j \leq u - l$ *or*

2ii $W^+ + W^= < b + l$ *and* $\max_{j \in N^-} w_j \leq u - l$

an optimal solution to KPC can be found by solving one or two standard KPs, respectively.

Proof.

2i The optimal solution (x, s) to KPC_{up} has $s \geq l$. Suppose $s < l$, then, there is an item $j \in N^+$ with $x_j = 0$. Since $w_j \leq u - l$ adding j to (x, s) yields a better solution. Hence, solving KPC_{up} according to section 3.1 yields an optimal solution to the KPC at hand.

2ii The optimal solution (x, s) to KPC_{lo} has $s \leq u$. Suppose $s > u$, then, there is an item $j \in N^-$ with $x_j = 1$. Since $w_j \leq u - l$ removing j from (x, s) yields a better solution. Hence, we solve KPC_{lo} according to section 3.2 and obtain an optimal solution to the KPC at hand.

□

Figure 1 illustrates the cases considered in theorems 1 and 2.

Summarizing, we identify three cases 1i to 1iii where solving KPC with effective lower and upper bounds for s can be done in linear time. Given cases 2i or 2ii we can reduce KPC to KP using the ideas presented in section 3.

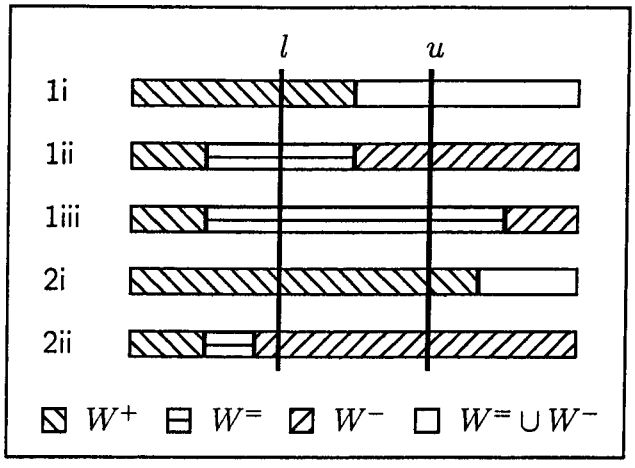

Figure 1: Illustration of cases of theorems 1 and 2

4.2 Binary Coding of the Adjustment

Since the technique presented in section 4.1 can not be applied in general we develop a reformulation of the underlying problem where the adjustment is represented by discrete values. Naturally, this means a restriction of solution space. Therefore, a proof of the solution's quality is presented.

First, we replace the capacity adjustment s by $s = u - \bar{s}$, see Nauss [8]. The new variable $\bar{s}$ represents the unused capacity adjustment and, thus, it is bounded by $0 \leq \bar{s} \leq u - l$. For illustration one can imagine, that in advance the capacity is extended at cost $c \cdot u$ and then, during the optimization, the superfluous capacity can be sold which would be represented by $\bar{s} > 0$. Now, the continuous variable $\bar{s}$ is substituted by binary code $\tilde{s} = \sum_{i \in M} a_i y_i$, with $M = \{1, \ldots, m\}$. Here, binary variable y_i equals one if and only if a_i contributes to the sold capacity. The reformulation, namely KPC_{bin}, can be represented as follows:

$$-c \cdot u + \max \sum_{j \in N} p_j x_j + c \cdot \sum_{i \in M} a_i y_i \tag{25}$$

$$\text{s.t.} \sum_{j \in N} w_j x_j + \sum_{i \in M} a_i y_i \leq b + u \tag{26}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in N \tag{27}$$

$$y_i \in \{0,1\} \qquad \forall\, i \in M \tag{28}$$

Obviously, terms (25) to (28) form a standard KP. A major drawback regarding this reformulation is that $\tilde{s}$ can not represent each value in $[0, u - l]$ being the domain of $\bar{s}$. Hence, we focus on the choice of m and a_i, $i \in M$, in the following.

First, we detail our requirements regarding m and a_i, $i \in M$:

i Values of $\tilde{s}$ must not undershoot 0 or exceed $u - l$, respectively, that is $0 \leq \sum_{i \in M} a_i y_i \leq u - l$ for each binary vector $y = \{y_1, \ldots, y_m\}$.

ii Given a precision $p \in \mathbb{N}$ for each value $v \in [0, u-l]$ there must be a binary vector y such that $|v - \sum_{i \in M} a_i y_i| \leq 10^{-p}$.

iii Regarding i and ii the number of additional binary variables m should be chosen as small as possible.

In the following we present an approach in order to ensure fulfillment of each condition. The idea is closely related to standard binary encoding. First, we focus on pure integer values of $\tilde{s}$ and, therefore, $p = 0$. If $u - l = 2^k - 1$ with $k \in \mathbb{N}$ then $m = \log_2(u - l + 1)$ and $a_i = 2^{i-1}$ for each $i \in M$ is the obvious choice. In general, if $u - l \neq 2^k - 1$ for all $k \in \mathbb{N}$ we choose $m = \lceil \log_2(u - l + 1) \rceil$, $a_i = 2^{i-1}$ for each $i \in \{1, \ldots, m-1\}$ and $a_m = u - l - 2^{m-1} + 1$. Condition i is fulfilled since $\sum_{i \in M} a_i \cdot 0 = 0$ and $\sum_{i \in M} a_i \cdot 1 = u - l$. It is well–known, that any integer value can be stated by a binary representation. Thus, every integer up to $2^{m-1} - 1$ can be expressed by a vector $y' = \{y_1, \ldots, y_{m-1}\}$. Since $a_m \leq \sum_{i=1}^{m-1} a_i + 1$, we can express any integer $v \in [0, u - l]$ and condition ii is satisfied. Moreover, we obtain the minimum possible value for m since we can not have $m < \lceil \log_2(u - l + 1) \rceil$ if we aim at binary codes. Therefore, condition iii is fulfilled as well.

Now, we turn to the representation of real numbers. Regarding the required precision $p \in \mathbb{N}$ we apply the procedure described above to the value $\lfloor 10^p \cdot (u - l) \rfloor$. After obtaining values a_i, $i \in M$, we divide them by 10^p. Obviously, these transformed values suffice condition ii, see Müller-Bungart [7]. The number of additional variables is now $m = \lceil \log_2(\lfloor 10^p (u - l) \rfloor + 1) \rceil$.

In order to illustrate the choice of m and a_i, $i \in M$, consider the following example: $u - l = 8.35$ and $p = 1$ and, hence, $\lfloor 10^p \cdot (u - l) \rfloor = 83$. Consequently, $m = \lceil \log_2(83 + 1) \rceil = 7$ and $a_i = \frac{2^{i-1}}{10}$ for $i \in \{1, \ldots, 6\}$. Now, for each $v \in [0, 6.3]$ we can represent a value $v' = \sum_{i=1}^{6} a_i y_i$ with $|v - v'| \leq 10^{-1}$. Adding $a_7 = \frac{83 - 2^6 + 1}{10} = 2$ we cover $[0, 8.35]$ with binary codes and given precision $p = 1$. Note that we create a certain degree of redundancy since specific values of $\tilde{s}$ can be expressed by several binary codes, for example 3.1 can be represented either by $(y_1 = y_2 = y_3 = y_4 = y_5 = 1, y_6 = y_7 = 0)$ or by $(y_1 = y_2 = y_4 = y_7 = 1, y_3 = y_5 = y_6 = 0)$.

Note, if s is just limited by an effective lower bound, see section 3.2, our reformulation KPC_{bin} can be used as well by setting $u \geq W - b$. The advantage in comparison to the proposed algorithm in section 3.2 is, that always only one knapsack problem has to be solved instead of two.

Since we reduce the solution space to discrete values of $\tilde{s}$ within $[0, u-l]$, where the domain of $\tilde{s}$ is a subset of the domain of $\bar{s}$, the solution obtained via KPC_{bin} is a feasible solution for the original model KPC, too. Thus, the solution to KPC_{bin} is a lower bound for the optimal solution to KPC. Given an optimal solution $(x, \tilde{s})$ to KPC_{bin} the lower bound can be strengthened if $b + l < \sum_{j \in N} w_j x_j < b + u - \tilde{s}$. Then, $\tilde{s}$ can be increased to $\tilde{s} = b + u - \sum_{j \in N} w_j x_j$, denoted by $\lceil \tilde{s} \rceil^{=}$, according to lemma 1. In the following we analyze the gap between the lower bound by KPC_{bin} and the optimal solution to KPC.

Theorem 3. *The gap Δ between the lower bound obtained by KPC_{bin} and the optimal objective value according to KPC can not be larger than $c \cdot 10^{-p}$.*

Proof. Let $\lfloor \bar{s} \rfloor_{\tilde{s}}$ denote the binary code which is closest to but not larger than $\bar{s}$. Then, given the optimal solution $(x, \bar{s})$ to KPC and the optimal solution $(x', \tilde{s})$ to KPC_{bin} the following

holds.

$$\sum_{j\in N} p_j x_j + c\cdot \bar{s} \;\geq\; \sum_{j\in N} p_j x'_j + c\cdot \lceil \tilde{s} \rceil^{=} \tag{29}$$

$$\geq \sum_{j\in N} p_j x'_j + c\cdot \tilde{s} \tag{30}$$

$$\geq \sum_{j\in N} p_j x_j + c\cdot \lfloor \bar{s} \rfloor_{\tilde{s}} \tag{31}$$

where the constant $-c\cdot u$ has been left out for simplification.

Relation (29) expresses that the improved optimal solution to KPC_{bin} is a lower bound to the optimal solution of KPC. Obviously, this improved solution has no smaller value than the optimal solution to KPC_{bin} itself, giving (30). Representation of $(x, \bar{s})$ in terms of KPC_{bin} can not have a larger value than $(x', \tilde{s})$, as otherwise this solution would have been found with KPC_{bin} as well, therefore (31) is valid.

Apparently, regarding requirement ii

$$\Delta \leq \sum_{j\in N} p_j x_j + c\cdot \bar{s} - \sum_{j\in N} p_j x_j + c\cdot \lfloor \bar{s} \rfloor_{\tilde{s}} = c\,(\bar{s} - \lfloor \bar{s} \rfloor_{\tilde{s}}) \leq c\cdot 10^{-p}.$$

□

As we can see from theorem 3 we can always obtain a lower bound (and a corresponding heuristic solution) which is arbitrarily close to the optimal objective value by increasing p. This leads to the question whether we can choose p such that we obtain an optimal solution by solving KPC_{bin}.

5 Computational results

After giving the theoretical background in the previous sections, we now present some computational results to show the efficiency of the developed algorithms. The algorithms for all presented models, the KPC, the KPC_{lb} and the KPC_{bin}, are coded in C^{++} and executed on an Intel Pentium D processor with 1GB RAM and 2.80 GHz clockpulse using the operating system Windows XP.

We employ the test instance generator designed by Martello et al. [5] in order to generate KPs with 1000 items according to three of those classes evaluated as hard in Pisinger [9]: in instance class I_{sc} items' profit and capacity consumption are strongly correlated, in instance class I_{sic} items' profit and capacity consumption are strongly inverse correlated, and in instance class I_{asc} items' profit and capacity consumption are almost strongly correlated. Note, that all generated coefficients are integer numbers.

As shown in the previous sections there are two crucial factors for the usage of additional capacity: capacity b determining how many and which items can be chosen without using additional capacity and cost c determining W^+, $W^=$, and W^-. Therefore, we focus on variation of both factors for the three classes mentioned above. We successively increase the capacity given from $\frac{1}{11}W$ to $\frac{10}{11}W$ by steps amounting to $\frac{1}{11}W$. As outlined in sections 3 and 4,

if c is quite large or quite small KPC is equivalent to KP or trivial to solve, respectively. Both cases are not interesting for our study. Let $\bar{e} = \frac{1}{n}\sum_{j\in N}\frac{p_j}{w_j}$ be the average items' efficiency. Then, we vary c from $\frac{1}{10}e$ to $\frac{15}{10}e$ by steps of $\frac{1}{10}e$ and restrict s to be at least 0.

In order to show efficiency of our approaches we solve these instances using the commercial solver ILOG CPLEX 10.1, where the solution process is aborted if optimality is not proven within 600 seconds. Using the reduction to the standard KP and the binary coding enables us to use combo.

First of all, we observe that in analogy to the standard KP, see Pisinger [9] for example, most instances are easy to solve resulting in an optimal solution. However, depending on capacity b and cost c there are instances leading to enormous run times. For I_{sc}, instances are hard if c is close to $\bar{e}$ which means that for many items it is not obvious whether to choose them or not and if additional capacity is needed. We observe the same effect for I_{asc} but for c being wider spread around $\bar{e}$. Instead, for I_{sic}, hard instances seem not to obey this pattern but to exist for arbitrary c. For about 10% (84 out of 870) of all instances CPLEX reached the run time limit of 600 sec. For about 3% (27 out of 870) of the instances CPLEX proofed optimality within 600 sec. but needed more than 1 sec. The remaining instances were solved in less than 1 sec. Both of our approaches always solved each single instance in less than 1 sec. Clearly, the approach using binary coding needs approximately half of the time needed by the approach solving two standard KPs to the cost of obtaining a heuristic solution, where the quality depends on the coefficients, see section 4.2. In this study, as all coefficients are integers, the obtained solution is always optimal.

Summarizing, as known for the standard KP there are few hard instances for KPC. Both of our approaches allow to carry the efficiency of algorithm combo over from KP to KPC. Impressively, each single instance was solved within seconds.

6 Conclusions and Outlook

In our work, we examine the KPC with all possible limitations of capacity adjustment s. For each case we present at least one solution approach. The underlying idea is to reduce the problem at hand to the well-known standard KP, for which the algorithm combo is known to be very efficient. We employ two ideas to do so: splitting solution space where exploration of each part is equivalent to a standard KP and replacing the continuous variable by a binary code where each flag can be seen as an item.

Considering the four possible restrictions to s we can conclude the following:

- If s is not restricted at all KPC is solvable in linear time.
- If s is restricted only from above KPC can be reduced to KP and solution of KPC requires solving a single KP.
- If s is restricted only from below KPC can be reduced to KP by splitting solution space and, then, solution of KPC requires solving two KPs. Furthermore, a heuristic solution can be obtained via binary coding which requires solving a single KP.

- If s is restricted from below as well as from above KPC we identify several cases where solving KPC can be done in linear time. Furthermore, we provide sufficient conditions allowing solving the KPC by splitting solution space and solving no more than two KPs. Our heuristic approach of binary coding can be employed unconditionally and requires solving a single KP.

As for KP most instances of KPC are solved within seconds by means of standard commercial solver CPLEX. For those we cannot provide any progress. However, depending on the combination of scarceness of capacity b and cost c of additional capacity there are instances which cannot be solved by CPLEX in short time. Therefore, in line with the reasoning for employing combo to solve KP our approach enables us to solve KPC by using combo and avoid high run times for specific instances.

For future research we can think of two branches: First, the question is whether combo can be adapted to solve KPC itself without loosing its high efficiency. Second, whether the concept of replacing a continuous variable by a binary representation can be transferred e.g. to the elastic generalized assignment problem, see Nauss [8].

References

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[2] M. F. Gorman and S. Ahire. A major appliance manufacturer rethinks its inventory policies for service vehicles. *Interfaces*, 36(5):407–419, 2006.

[3] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer Berlin, 2004.

[4] H. Marchand and A. Wolsey. The 0-1 knapsack problem with a single continuous variable. *Mathematical Programming*, 85:15–33, 1997.

[5] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, March 1999.

[6] S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123:325–332, 2000.

[7] M. Müller-Bungart. Revenue management with flexible products. Lecture Notes in Economics and Mathematical Systems. Springer Berlin, 2007.

[8] R. M. Nauss. The elastic generalized assignment problem. *Journal of the Operational Research Society*, 55:1333–1341, 2004.

[9] D. Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

[10] L. A. Wolsey. Strong formulations for mixed integer programs: valid inequalities and extended formulations. *Mathematical Programming*, 97(1-2):423–447, 2003.