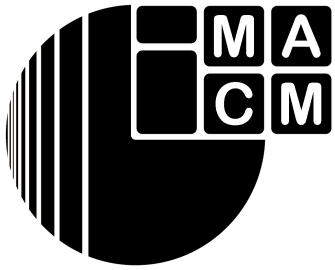

Bergische Universität Wuppertal

Fachbereich Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 18/26

B. Schulze, M. Stiglmayr, L. Paquete, C. M. Fonseca, D. Willems und S. Ruzika

On the Rectangular Knapsack Problem - Approximation of a Specific Quadratic Knapsack Problem

January 7, 2019

http://www.math.uni-wuppertal.de

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

On the Rectangular Knapsack Problem

Approximation of a Specific Quadratic Knapsack Problem

Britta Schulze · Michael Stiglmayr ·
Luís Paquete · Carlos M. Fonseca ·
David Willems · Stefan Ruzika

Abstract In this article, we introduce the *rectangular knapsack problem* as a special case of the quadratic knapsack problem consisting in the maximization of the product of two separate knapsack profits subject to a cardinality constraint. We propose a polynomial time algorithm for this problem that provides a constant approximation ratio of 4.5. Our experimental results on a large number of artificially generated problem instances show that the average ratio is far from theoretical guarantee. In addition, we suggest refined versions of this approximation algorithm with the same time complexity and approximation ratio that lead to even better experimental results.

Keywords quadratic knapsack problem, approximation algorithm, multiobjective combinatorial optimization, hypervolume

1 Introduction

In contrast to classical (linear) knapsack problems, the profit of a collection of items in a quadratic knapsack problem is not only determined by the sum of individual profits, but also by profits generated by pairwise combinations of items. This can be used to model the fact that two items may complement each other such that their profit is increased if both of them are selected.

B. Schulze · M. Stiglmayr
Department of Mathematics and Natural Sciences, University of Wuppertal, Gaußstr. 20, 42119 Wuppertal, Germany
E-mail: schulze@math.uni-wuppertal.de

L. Paquete · C. M. Fonseca
CISUC, Department of Informatics Engineering, University of Coimbra, Portugal

D. Willems
Mathematical Institute, University of Koblenz-Landau, Koblenz, Germany

S. Ruzika
Department of Mathematics, Technical University of Kaiserslautern, Kaiserslautern, Germany

The quadratic objective still allows to model that the profit of two items is independent of each other by setting the combined profit to 0. In this case, including both items does not increase the profit over the sum of the individual profits. Furthermore, a negative combined profit value can model the fact that both items together are less profitable than the sum of individual profits. This might be the case, if both items are substitutes for each other and including both items is as profitable as including one.

The formulation of quadratic knapsack problems (QKP) is very general and, therefore, its range of applications is quite wide. For example, Johnson et al. [1993] present a problem in the context of compiler construction that can be formulated as a quadratic knapsack problem. Moreover, QKP have been discussed in the context of the location of airports, freight handling terminals, railway stations, and satellite stations [Rhys, 1970, Witzgall, 1975].

We present a variant of QKP which we call the *rectangular knapsack problem* (RKP). The profit matrix is built by the product of two vectors and the constraint is a cardinality constraint.

The main motivation for problem RKP arises when solving the *cardinality constrained bi-objective knapsack problem* (2oKP)

$$\begin{aligned} \max \quad & \left(\sum_{i=1}^{n} a_i\, x_i,\ \sum_{i=1}^{n} b_i\, x_i \right) \\ \text{s. t.} \quad & \sum_{i=1}^{n} x_i \leq k \\ & x_i \in \{0,1\}, \qquad i = 1, \ldots, n \end{aligned} \tag{2oKP}$$

where $a, b \in \mathbb{N}^n$, with $a, b \neq \mathbf{0}_n = (0, 0, \ldots, 0)^\top \in \mathbb{N}^n$, and $k \in \mathbb{N}$, $k < n$. Instead of computing the set of efficient solutions for this bi-objective optimization problem, we want to find one (or several) representative nondominated point(s). Originally proposed by Zitzler and Thiele [1998] in the context of evolutionary algorithms, the hypervolume indicator is often used as a versatile quality measure of representation of the efficient set in multiobjetive optimization [c.f. Kuhn et al., 2016]. The problem of finding one solution of 2oKP that maximizes the hypervolume, considering $(0, 0)^\top$ as reference point, is equivalent to RKP.

The structure of RKP allows to formulate a polynomial time 4.5-approximation algorithm. An algorithm is called a *polynomial time ρ-approximation algorithm*, if it computes a feasible solution in run time being polynomial in the encoding length of the input such that

$$\rho \geq \max \left\{ \frac{\mathrm{OPT}}{\mathrm{ALG}}, \frac{\mathrm{ALG}}{\mathrm{OPT}} \right\}.$$

Here, OPT denotes the optimal objective function value of the maximization problem and ALG the objective function value of the solution which is the output of the algorithm [Cormen et al., 2001].

The remainder of this article is organized as follows: In Section 2, we give an introduction to quadratic knapsack problems. We introduce the rectangular knapsack problem in Section 3 and present upper and lower bounds. These bounds motivate an approximation algorithm that is formulated in Section 4, for which a constant approximation ratio ρ is proven. Furthermore, we also introduce improved implementations of this approximation method. In Section 5 we present a computational study of these algorithms and compare the realized approximation ratios to the theoretical bound of 4.5. Section 6 concludes this article.

2 Quadratic knapsack problems

Gallo et al. [1980] first introduced the *binary quadratic knapsack problem* (QKP). It is a variant of the classical knapsack problem and can be concisely stated as follows: Given n items, the profit for including item i is given by the coefficient p_{ii}. Additionally, a profit $p_{ij} + p_{ji}$ is generated if both items i and j are selected. The values p_{ij} (which are often assumed to be non-negative integers) can be compactly written in a profit matrix

$$P := (p_{ij})_{\substack{i=1,\dots,n \\ j=1,\dots,n}}.$$

The profits p_{ij} and p_{ji} are either both realized, i. e., if items i and j are selected, or both not realized, i. e., if item i or item j is not selected. Hence, p_{ij} and p_{ji} can be assumed to be equally valued, which results in a symmetric matrix P.

As for the classical knapsack problem, each item i has a positive integral weight w_i and the goal is to select a subset of items that maximizes the overall profit while the sum of weights does not exceed the given capacity W. As usual, the binary decision variable x_i indicates if item i is selected, $x_i = 1$, or not, $x_i = 0$. Thus, QKP can be defined as follows:

$$\begin{aligned} \max \quad & x^\top P x = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij}\, x_i\, x_j \\ \text{s. t.} \quad & \sum_{i=1}^{n} w_i\, x_i \leq W \\ & x_i \in \{0,1\}, \qquad i = 1, \dots, n. \end{aligned} \tag{QKP}$$

An illustrative interpretation of QKP can be given based on graphs. We define a complete undirected graph $G = (V, E)$ with vertex set $V = \{1, \dots, n\}$, i. e., each vertex corresponds to one item of QKP. Each vertex has assigned a profit value p_{ii} and a weight value w_i and each edge (i, j) has assigned a profit value $p_{ij} + p_{ji}$. The task is to select a clique $S \subset V$ with maximal profit which does not exceed the capacity W. The overall profit of S consists of the profit of the vertices in S and edges (i, j) connecting two vertices $i, j \in S$. It is well known that the quadratic knapsack problem is $\mathcal{NP}$-complete in the strong

sense, which can be shown by a polynomial reduction from the *Clique*-problem [Garey and Johnson, 1979, Pisinger, 2007].

The quadratic knapsack problem has been widely studied in the literature, see Pisinger [2007] for a comprehensive survey. Exact solution algorithms are mainly based on branch-and-bound (B&B) schemes. Besides the model of QKP, Gallo et al. [1980] also presented the first B&B algorithm for this optimization problem. Their approach makes use of upper bounds that are computed by a relaxed version of QKP. The objective function is replaced by an upper plane, that is a linear function g such that $g(x) \geq x^\top Px$ for any feasible solution x of QKP. Solving this new optimization problem, which is a classical knapsack problem due to the linear objective function, yields an upper bound on QKP.

Billionnet and Calmels [1996] linearize the quadratic problem to compute upper bounds for a B&B algorithm. Caprara et al. [1999] use upper planes for computing upper bounds. In addition, they present a reformulation to an equivalent instance of QKP using Lagrangian relaxation. The reformulated instance provides a tight upper bound at the root node of the B&B algorithm. Billionnet et al. [1999] use upper bounds based on Lagrangian decomposition and Billionnet and Soutif [2004] introduce algorithms for fixing variables based on these bounds. The aim of these algorithms is to reduce the size of the problem before applying the B&B. Helmberg et al. [2000] apply several semidefinite relaxation techniques to QKP and include cutting planes to improve their basic approaches. Pisinger et al. [2007] introduce an aggressive reduction algorithm for large instances of QKP, where *aggressive* means that a large effort is spent to fix as many of the decision variables as possible at their optimal value such that the final optimization is done rather easy. The authors apply the bounds of Caprara et al. [1999] and Billionnet et al. [1999] for the reduction of QKP. Rodrigues et al. [2012] present a linearization scheme for QKP that provides tight upper bounds for a B&B algorithm.

However, few results are known about the approximation of QKP. Since the problem is strongly $\mathcal{NP}$-hard, a fully polynomial time approximation scheme (FPTAS) cannot be expected unless $\mathcal{P} = \mathcal{NP}$. Furthermore, it is unknown whether there exists an approximation with a constant approximation ratio for QKP. Taylor [2016] present an approximation algorithm based on an approach for the densest k-subgraph problem. They show that for $\varepsilon > 0$, QKP can be approximated with an approximation ratio in $\mathcal{O}(n^{2/5+\varepsilon})$ and a run time of $\mathcal{O}(n^{9/\varepsilon})$. Rader and Woeginger [2002] prove that for a variant of QKP, where positive as well as negative profit coefficients p_{ij} are considered, there does not exist any polynomial time approximation algorithm with finite worst case guarantee unless $\mathcal{P} = \mathcal{NP}$.

Other approximation results concentrate on special cases of QKP where the underlying graph $G = (V, E)$, with $E = \{(i, j) : i, j \in V, i \neq j, p_{ij} \neq 0\}$, has a specific structure. Pferschy and Schauer [2016] present an FPTAS for QKP on graphs of bounded tree width, which includes series-parallel graphs [see Bodlaender and Koster, 2008]. Furthermore, the authors introduce a polynomial time approximation scheme (PTAS) for graphs that do not contain any

fixed graph H as a minor, which includes planar graphs. As negative results, Pferschy and Schauer [2016] show that QKP on 3-book embeddable graphs is strongly $\mathcal{NP}$-hard and Rader and Woeginger [2002] prove that QKP on vertex series-parallel graphs is strongly $\mathcal{NP}$-hard.

Kellerer and Strusevich [2010] introduce a very special variant of QKP: the symmetric quadratic knapsack problem. In addition to assigning a profit to pairs of items that both have been selected, this variant also assigns a profit to pairs of items that both have not been selected. Furthermore, the profits p_{ij} are built as a multiplicative of two coefficients of which one also defines the weight of the constraint. Kellerer and Strusevich [2010] introduce an FPTAS to solve the problem, which is further improved by Xu [2012].

3 Rectangular knapsack problems

The (cardinality constrained) rectangular knapsack problem (RKP) is a variant of QKP which can be written as follows:

$$\begin{aligned} \max \quad & f(x) = x^\top a\, b^\top x = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j\, x_i x_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i \le k \\ & x_i \in \{0,1\}, \qquad i = 1,\dots,n \end{aligned} \tag{RKP}$$

where $a, b \in \mathbb{N}^n$, with $a, b \neq \mathbf{0}_n$, and $k \in \mathbb{N}$, $k < n$. Note, that $P = a\, b^\top$, i. e., $\operatorname{rank}(P) = 1$, with $p_{ij} = a_i b_j$ and $p_{ji} = a_j b_i$, i. e., in general, P is not symmetric. We assume that $k \geq 2$. Otherwise, i. e., if $k = 1$, the problem reduces to finding the largest coefficient $a_i b_i$, for $i \in \{1,\dots,n\}$.

The rectangular objective function is formulated in analogy to the so-called Koopmans-Beckmann form of the quadratic assignment problem, see Burkard et al. [1998], which is also a particular case of the more general Lawler formulation. In both cases, the two respective four dimensional arrays of profit/cost coefficients are given as a product of two lower dimensional parameter matrices or vectors, respectively.

3.1 Illustrative interpretation

The denotation *rectangular* knapsack problem is motivated by the special structure of P given by the coefficients $a_i\, b_j$. Each coefficient can be interpreted as the area of a rectangle. Accordingly, for fixed item $\hat{\imath} \in \{1,\dots,n\}$, all rectangles corresponding to coefficients $a_{\hat{\imath}} b_j$, $j = 1,\dots,n$, have the same width, and all rectangles corresponding to coefficients $a_j\, b_{\hat{\imath}}$, $j = 1,\dots,n$, have the same height. Note that the objective function can be rewritten as

$$f(x) = x^\top a\, b^\top x = (a^\top x) \cdot (b^\top x) = \sum_{i=1}^{n} a_i\, x_i \cdot \sum_{i=1}^{n} b_i\, x_i,$$

which can be interpreted as choosing a subset $S \subset \{1, \ldots, n\}$ of items such that the area of the rectangle with width $\sum_{i \in S} a_i$ and height $\sum_{i \in S} b_i$ is maximized.

Example 1 We consider the following instance of RKP:

$$\begin{aligned} \max \quad & \left((4,5,2,12,7)^\top x\right) \cdot \left((6,3,8,5,10)^\top x\right) \\ \text{s.t.} \quad & \sum_{i=1}^{5} x_i \leq 2 \\ & x_i \in \{0,1\}, \qquad i = 1, \ldots, 5 \end{aligned}$$

The corresponding rectangles are plotted in Figure 1. Each rectangle has the same position in the overall rectangle as the corresponding coefficient $p_{ij} = a_i b_j$ in the profit matrix P. The optimal solution $x = (0,1,0,0,1)^\top$ generates an objective function value that corresponds to the highlighted area in the figure.

	b_1	b_2	b_3	b_4	b_5
a_1	4 · 6	4 · 3	4 · 8	4 · 5	4 · 10
a_2	5 · 6	5 · 3	5 · 8	5 · 5	5 · 10
a_3	2 · 6	2 · 3	2 · 8	2 · 5	2 · 10
a_4	12 · 6	12 · 3	12 · 8	12 · 5	12 · 10
a_5	7 · 6	7 · 3	7 · 8	7 · 5	7 · 10

Fig. 1 Visualization of coefficients $p_{ij} = a_i\, b_j$, interpreted as areas of rectangles.

3.2 Bounds

The structure of the profit matrix P implies an easy computation of bounds for RKP. In the following, we assume that all instances are defined or reordered such that

$$a_1 \geq \ldots \geq a_n$$

and, in case of ties, i. e., if $a_i = a_{i+1}$ for $i \in \{1, \ldots, n-1\}$, such that

$$b_i \geq b_{i+1}.$$

Let $\mathcal{S}_n$ denote the symmetric group of order n and $\pi \in \mathcal{S}_n$ denote a permutation of $\{1, \ldots, n\}$. More specifically, consider π such that

$$b_{\pi(1)} \geq \ldots \geq b_{\pi(n)}$$

and in case of ties, i. e., if $b_{\pi(j)} = b_{\pi(j+1)}$ for $j \in \{1, \ldots, n-1\}$, such that

$$a_{\pi(j)} \geq a_{\pi(j+1)}.$$

Using the sorted coefficients a_i, $b_{\pi(j)}$ of the objective function, one can compute an upper bound for RKP in a straightforward way.

Lemma 1 *For every feasible solution $x \in \{0,1\}^n$ of RKP, the following inequality holds:*

$$f(x) \leq \sum_{i=1}^{k} a_i \cdot \sum_{j=1}^{k} b_{\pi(j)} := \mathcal{U}$$

This bound is tight, if

$$\{\pi(j) : 1 \leq j \leq k\} = \{1, \ldots, k\}. \tag{1}$$

Note that, in general, this upper bound does not correspond to a solution of RKP since the value of a variable x_i may be differently defined w. r. t. the respective sorting of the coefficients. As soon as Equation (1) holds, the upper bound $\mathcal{U}$ corresponds to a feasible solution of RKP and this solution is optimal.

Proof We consider the objective function of RKP:

$$f(x) = \sum_{i=1}^{n} a_i\, x_i \cdot \sum_{i=1}^{n} b_i\, x_i = \sum_{i=1}^{n} a_i\, x_i \cdot \sum_{j=1}^{n} b_{\pi(j)}\, x_{\pi(j)}.$$

The cardinality constraint restricts the number of selected items to k. Due to the ordering of coefficients a_i, it is

$$0 \leq \sum_{i=1}^{n} a_i\, x_i \leq \sum_{i=1}^{k} a_i$$

for every feasible solution x of RKP. Analogously, due to the definition of the permutation π, we know that

$$0 \leq \sum_{j=1}^{n} b_{\pi(j)}\, x_{\pi(j)} \leq \sum_{j=1}^{k} b_{\pi(j)}$$

for every feasible solution x of RKP. Thus,

$$f(x) = \sum_{i=1}^{n} a_i\, x_i \cdot \sum_{j=1}^{n} b_{\pi(j)}\, x_{\pi(j)} \leq \sum_{i=1}^{k} a_i \cdot \sum_{j=1}^{k} b_{\pi(j)} = \mathcal{U}.$$

Furthermore, if $\{\pi(j) : 1 \leq j \leq k\} = \{1, \ldots, k\}$, the upper bound is based on the selection of the k items $1, \ldots, k$:

$$\sum_{i=1}^{k} a_i \cdot \sum_{j=1}^{k} b_{\pi(j)} = \sum_{i=1}^{k} a_i \cdot \sum_{i=1}^{k} b_i = \sum_{i=1}^{n} a_i \, x_i \cdot \sum_{i=1}^{n} b_i \, x_i$$

with

$$x_i = \begin{cases} 1 & \text{, for } i \in \{1, \ldots, k\} \\ 0 & \text{, otherwise} \end{cases}$$

The solution x is feasible and realizes $\mathcal{U}$. Hence, x is optimal and $\mathcal{U}$ is a tight upper bound. □

A lower bound on RKP can also be obtained by using the sorting of the coefficients. Let $\tilde{x}$ and $\hat{x} \in \{0,1\}^n$ be defined as follows:

$$\tilde{x}_i = \begin{cases} 1 & \text{, for } i \in \left\{1, \ldots, \left\lceil \frac{k}{2} \right\rceil\right\} \cup \left\{\pi(1), \ldots, \pi\left(\left\lfloor \frac{k}{2} \right\rfloor\right)\right\} \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

$$\hat{x}_i = \begin{cases} 1 & \text{, for } i \in \left\{1, \ldots, \left\lfloor \frac{k}{2} \right\rfloor\right\} \cup \left\{\pi(1), \ldots, \pi\left(\left\lceil \frac{k}{2} \right\rceil\right)\right\} \\ 0 & \text{, otherwise} \end{cases} \tag{3}$$

For notational convenience, let $\kappa := \frac{k}{2}$, $\overline{\kappa} := \left\lceil \frac{k}{2} \right\rceil$, and $\underline{\kappa} := \left\lfloor \frac{k}{2} \right\rfloor$. If k is even, the equality $\overline{\kappa} = \underline{\kappa} = \frac{k}{2}$ holds, i. e., $\tilde{x}$ and $\hat{x}$ are identical.

Remark 1 The definition of $\tilde{x}$ guarantees that at least the product

$$\sum_{i=1}^{\overline{\kappa}} a_i \cdot \sum_{j=1}^{\underline{\kappa}} b_{\pi(j)}$$

is realized in the objective function. Due to the ordering of the coefficients a_i and $b_{\pi(j)}$, this is the maximal possible value that a product of $\overline{\kappa}$ coefficients a_i and $\underline{\kappa}$ coefficients b_j can achieve. The same holds analogously for $\hat{x}$. This property is important to prove an approximation quality in the following, see the proof of Theorem 1.

Lemma 2 *For an optimal solution x^* of RKP, the following inequality holds:*

$$f(x^*) \geq \max\left\{f(\tilde{x}), f(\hat{x})\right\} := \mathcal{L}.$$

Proof The solutions $\tilde{x}$ and $\hat{x}$ are both elements of $\{0,1\}^n$. The sets $\{1, \ldots, \overline{\kappa}\}$ and $\{\pi(1), \ldots, \pi(\overline{\kappa})\}$ have cardinality $\overline{\kappa}$ and the sets $\{\pi(1), \ldots, \pi(\underline{\kappa})\}$ and $\{1, \ldots, \underline{\kappa}\}$ have cardinality $\underline{\kappa}$. Therefore, it holds:

$$\left.\begin{aligned} \sum_{i=1}^{n} \tilde{x}_i \leq \overline{\kappa} + \underline{\kappa} \\ \sum_{i=1}^{n} \hat{x}_i \leq \underline{\kappa} + \overline{\kappa} \end{aligned}\right\} = k. \tag{4}$$

Both solutions $\tilde{x}$ and $\hat{x}$ are feasible for RKP and the corresponding objective function values are lower bounds on the optimal objective function value. □

Remark 2 Note that equality is obtained in Equation 4 if the sets $\{1,\dots,\overline{\kappa}\}$ and $\{\pi(1),\dots,\pi(\underline{\kappa})\}$ ($\{1,\dots,\underline{\kappa}\}$ and $\{\pi(1),\dots,\pi(\overline{\kappa})\}$, respectively) are disjoint. If the sets are not disjoint, the bound can be improved by including more items. We discuss this in Section 4.1.

We define $\widetilde{\mathcal{L}} := f(\tilde{x})$ and $\widehat{\mathcal{L}} := f(\hat{x})$. The following example shows a connection between the bound computation and the visualization of RKP as a selection of a subset of rectangular areas.

Example 2 Consider the following instance of RKP:

$$\begin{aligned} \max \quad & \left((6,5,5,4,3,3,2,1)^\top x\right) \cdot \left((6,11,4,10,6,9,1,8)^\top x\right) \\ \text{s.t.} \quad & \sum_{i=1}^{8} x_i \le 5 \\ & x_i \in \{0,1\}, \qquad i = 1,\dots,8. \end{aligned}$$

Thus, the permutation π is $\pi = (2,4,6,8,1,5,3,7)^\top$ and the permuted vector b_π is given by $b_\pi = (11,10,9,8,6,6,4,1)^\top$.

As described above, the coefficients $a_i \cdot b_{\pi(j)}$, for $i,j = 1,\dots,8$, can be interpreted as rectangles with width a_i and height $b_{\pi(j)}$ and, consequently, with area $a_i \cdot b_{\pi(j)}$. We arrange the rectangles line by line according to the index i, and column by column according to the index $\pi(j)$ (see Figure 2). In doing so, the rectangles representing the coefficients are sorted in non-increasing manner from top to bottom and from left to right. Feasible solutions of RKP correspond to 5^2 rectangles, which have to be part of intersections of rows and columns with equal sets of indices $\mathcal{I}$, i. e., a set of indices $\mathcal{I} \subset \{1,\dots,8\}$ with $|\mathcal{I}| \le 5$.

- The upper bound computation chooses the 5 largest rows and columns, i. e., a_1 to a_5 and $b_{\pi(1)}$ to $b_{\pi(5)}$. In our example, we obtain:

$$\mathcal{U} = \sum_{i=1}^{5} a_i \cdot \sum_{j=1}^{5} b_{\pi(j)} = (6{+}5{+}5{+}4{+}3)\cdot(11{+}10{+}9{+}8{+}6) = 23\cdot 44 = 1012.$$

  This corresponds to the area of the 5^2 largest rectangles in the upper left part of the overall rectangle in Figure 2.
- For the lower bound computation at most 5 variables corresponding to the first three and two (two and three, respectively) indices of rows and columns are selected. In doing so, the largest $2{\cdot}3$ rectangles in the upper left part of the overall rectangle in Figure 3 (lower bound $\widehat{\mathcal{L}}$) are included in the

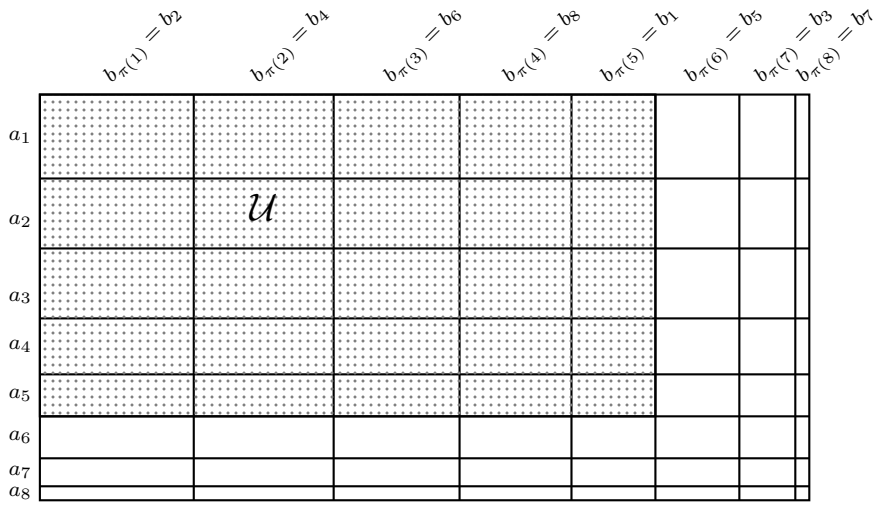

Fig. 2 Area that defines the upper bound $\mathcal{U}$ (⬚) for Example 2.

solution and, in addition, feasibility is guaranteed. In the example, the candidate solutions are $\tilde{x} = (1,1,1,1,0,0,0,0)^\top$ and $\hat{x} = (1,1,0,1,0,1,0,0)^\top$. The lower bound is computed as:

$$\begin{aligned}\mathcal{L} &= \max\{\widetilde{\mathcal{L}}, \widehat{\mathcal{L}}\} \\ &= \max\big\{(6+5+5+4)\cdot(6+11+4+10), \\ &\qquad\qquad (6+5+4+3)\cdot(6+11+10+9)\big\} \\ &= \max\{620, 648\} = 648.\end{aligned}$$

The optimal solution of this instance is $x^* = (1,1,1,1,0,1,0,0)^\top$ with $f(x^*) = 920$. We can verify that indeed: $\mathcal{U} = 1012 \geq 920 \geq 648 = \mathcal{L}$.

In this context, we show that the following inequality holds:

$$\mathcal{L} \geq \sum_{i=1}^{\underline{\kappa}} \sum_{j=1}^{\overline{\kappa}} a_i b_{\pi(j)}. \tag{5}$$

Referring to the description of Example 2, the right-hand side of this inequality corresponds to the area of the $\overline{\kappa} \cdot \underline{\kappa}$ largest rectangles in the left upper part of the overall rectangle (see also Remark 1). We partition the area corresponding

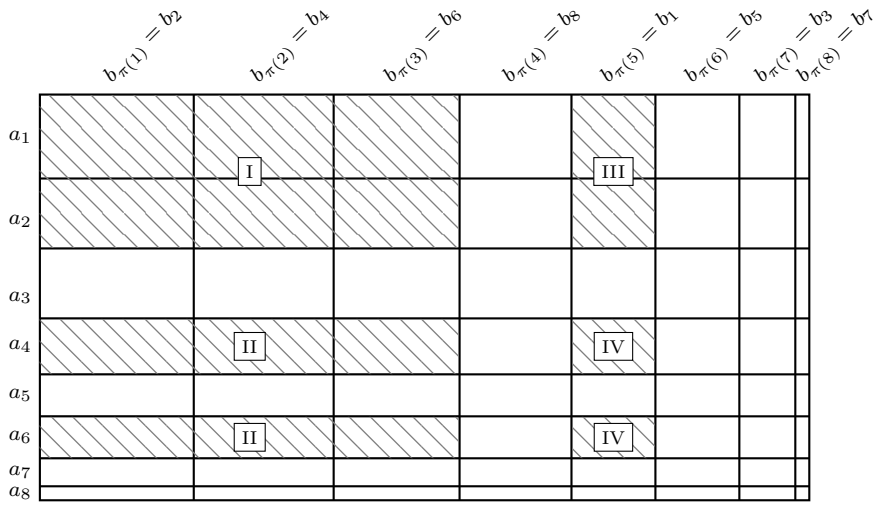

Fig. 3 Area that defines the lower bound $\mathcal{L} = \widehat{\mathcal{L}}$ (▨) for Example 2. The assignment of labels I to IV is relevant for the proof of Equality (5).

to the lower bound into four distinct areas to show that the inequality holds:

$$
\begin{aligned}
\mathcal{L} \geq \widehat{\mathcal{L}} &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i\, b_j\, \hat{x}_i\, \hat{x}_j \\
&= \underbrace{\sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\overline{\kappa}} a_i\, b_{\pi(j)}}_{\text{I}} + \underbrace{\sum_{i=1}^{\overline{\kappa}}\sum_{\pi(i)\notin\{1,\ldots,\underline{\kappa}\}}^{\overline{\kappa}} a_{\pi(i)}\, b_{\pi(j)}}_{\text{II}} \\
&\quad + \underbrace{\sum_{i=1}^{\underline{\kappa}}\sum_{j\notin\{\pi(1),\ldots,\pi(\overline{\kappa})\}}^{\underline{\kappa}} a_i\, b_j}_{\text{III}} + \underbrace{\sum_{i=1}^{\overline{\kappa}}\sum_{\pi(i)\notin\{1,\ldots,\underline{\kappa})\}}^{\underline{\kappa}}\sum_{j\notin\{\pi(1),\ldots,\pi(\overline{\kappa})\}} a_{\pi(i)}\, b_j}_{\text{IV}} \\
&\geq \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\overline{\kappa}} a_i\, b_{\pi(j)}
\end{aligned}
$$

In the context of Example 2, the four terms resulting from this partition correspond, in this order, to the four areas (I to IV) in Figure 3.

Analogously, using the definition of $\tilde{x}$, it holds that:

$$\mathcal{L} \geq \sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)}. \tag{6}$$

4 Approximation algorithms

The results of Section 3.2 naturally motivate an approximation algorithm, see Algorithm 1. It computes the solutions $\tilde{x}$ and $\hat{x}$ and outputs the better alternative as an approximate solution.

Algorithm 1 Approximation algorithm for RKP

Input: coefficients $a = (a_1, \ldots, a_n)^\top$ sorted in non-increasing order, $b = (b_1, \ldots, b_n)^\top$, capacity k

$\tilde{x} := \mathbf{0}_n$, $\hat{x} := \mathbf{0}_n$, $\overline{\kappa} := \left\lceil \frac{k}{2} \right\rceil$ and $\underline{\kappa} := \left\lfloor \frac{k}{2} \right\rfloor$

compute permutation $\pi \in \mathcal{S}_n$ such that

$$\begin{cases} b_{\pi(j)} > b_{\pi(j+1)}, \text{ or} \\ b_{\pi(j)} = b_{\pi(j+1)} \text{ and } a_{\pi(j)} \geq a_{\pi(j+1)} \end{cases} \quad \text{for } j = 1, \ldots, n-1$$

for $i := 1, \ldots, \underline{\kappa}$ **do** *//set $\tilde{x}$ and $\hat{x}$ analogous to (2) and (3)*
    $\tilde{x}_i := 1$, $\tilde{x}_{\pi(i)} := 1$
    $\hat{x}_i := 1$, $\hat{x}_{\pi(i)} := 1$
end for
$\tilde{x}_{\overline{\kappa}} := 1$
$\hat{x}_{\pi(\overline{\kappa})} := 1$
$\widetilde{\mathcal{L}} := (a^\top \tilde{x}) \cdot (b^\top \tilde{x})$
$\widehat{\mathcal{L}} := (a^\top \hat{x}) \cdot (b^\top \hat{x})$
if $\widetilde{\mathcal{L}} \geq \widehat{\mathcal{L}}$ **then**
    $\mathcal{L} := \widetilde{\mathcal{L}}$, $x := \tilde{x}$
else
    $\mathcal{L} := \widehat{\mathcal{L}}$, $x := \hat{x}$
end if

Output: lower bound $\mathcal{L}$ for RKP and corresponding solution x

The computation of $\tilde{x}$ and $\hat{x}$ and of their objective function values $\widetilde{\mathcal{L}}$ and $\widehat{\mathcal{L}}$ can be realized in time $\mathcal{O}(n)$. Therefore, with a time complexity of $\mathcal{O}(n \log n)$, the sorting of the coefficients determines the time complexity of Algorithm 1.

Theorem 1 *Algorithm 1 is a polynomial time* 4.5*-approximation algorithm for the rectangular knapsack problem.*

Proof Algorithm 1 returns a feasible solution in polynomial time $\mathcal{O}(n \log n)$.

Case 1: k even

Since the coefficients $a_i, b_{\pi(j)}$ are in non-increasing order, it holds that

$$\begin{aligned} \mathcal{U} &= \sum_{i=1}^{k} \sum_{j=1}^{k} a_i \, b_{\pi(j)} \\ &= \sum_{i=1}^{\kappa} \sum_{j=1}^{\kappa} a_i \, b_{\pi(j)} + \sum_{i=\kappa+1}^{k} \sum_{j=1}^{\kappa} a_i \, b_{\pi(j)} + \sum_{i=1}^{\kappa} \sum_{j=\kappa+1}^{k} a_i \, b_{\pi(j)} + \sum_{i=\kappa+1}^{k} \sum_{j=\kappa+1}^{k} a_i \, b_{\pi(j)} \\ &\leq 4 \cdot \sum_{i=1}^{\kappa} \sum_{j=1}^{\kappa} a_i \, b_{\pi(j)} \leq 4 \, \mathcal{L} \end{aligned}$$

Case 2: k odd

In analogy to case 1 we again use the fact that the coefficients a_i, $b_{\pi(j)}$ are in non-increasing order. We can assume without loss of generality that:

$$\sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\overline{\kappa}} a_i\, b_{\pi(j)} \leq \sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)}.$$

This inequality is equivalent to:

$$\begin{aligned}
& \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} + \sum_{i=1}^{\underline{\kappa}} a_i\, b_{\pi(\overline{\kappa})} \leq \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} + \sum_{j=1}^{\underline{\kappa}} a_{\overline{\kappa}}\, b_{\pi(j)} \\
\iff \quad & \sum_{i=1}^{\overline{\kappa}} a_i\, b_{\pi(\overline{\kappa})} - a_{\overline{\kappa}}\, b_{\pi(\overline{\kappa})} \leq \sum_{j=1}^{\overline{\kappa}} a_{\overline{\kappa}}\, b_{\pi(j)} - a_{\overline{\kappa}}\, b_{\pi(\overline{\kappa})} \\
\iff \quad & \sum_{i=1}^{\overline{\kappa}} a_i\, b_{\pi(\overline{\kappa})} \leq \sum_{j=1}^{\overline{\kappa}} a_{\overline{\kappa}}\, b_{\pi(j)}. \qquad (7)
\end{aligned}$$

Thus, the following inequality holds. Note that we use Equations 5 and 6 to bound several terms.

$$\begin{aligned}
\mathcal{U} &= \sum_{i=1}^{k}\sum_{j=1}^{k} a_i\, b_{\pi(j)} \\
&= \sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} + \sum_{i=\overline{\kappa}+1}^{k}\sum_{j=\underline{\kappa}+1}^{k} a_i\, b_{\pi(j)} + \sum_{i=1}^{\overline{\kappa}}\sum_{j=\underline{\kappa}+1}^{k} a_i\, b_{\pi(j)} + \sum_{i=\overline{\kappa}+1}^{k}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&\leq \mathcal{L} + \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\overline{\kappa}} a_i\, b_{\pi(j)} + \sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\overline{\kappa}} a_i\, b_{\pi(j)} + \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&\leq 2\,\mathcal{L} + \left(\sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} + \sum_{i=1}^{\overline{\kappa}} a_i\, b_{\pi(\overline{\kappa})}\right) + \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&\overset{(7)}{\leq} 3\,\mathcal{L} + \sum_{j=1}^{\overline{\kappa}} a_{\overline{\kappa}}\, b_{\pi(j)} + \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&\leq 3\,\mathcal{L} + a_{\overline{\kappa}}\, b_{\pi(\overline{\kappa})} + \sum_{j=1}^{\underline{\kappa}} a_{\overline{\kappa}}\, b_{\pi(j)} + \sum_{i=1}^{\underline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&= 3\mathcal{L} + a_{\overline{\kappa}}\, b_{\pi(\overline{\kappa})} + \sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&\leq 4\,\mathcal{L} + a_{\overline{\kappa}}\, b_{\pi(\overline{\kappa})} \quad \text{// worst case: } a_{\overline{\kappa}}\, b_{\pi(\overline{\kappa})} = a_i\, b_{\pi(j)},\ i = 1,\ldots,\overline{\kappa},\ j = 1,\ldots,\underline{\kappa} \\
&\leq 4\,\mathcal{L} + \frac{1}{\overline{\kappa}} \cdot \frac{1}{\underline{\kappa}} \cdot \sum_{i=1}^{\overline{\kappa}}\sum_{j=1}^{\underline{\kappa}} a_i\, b_{\pi(j)} \\
&\leq 4\,\mathcal{L} + \frac{1}{\overline{\kappa}} \cdot \frac{1}{\underline{\kappa}} \cdot \mathcal{L} \qquad (8) \\
&\leq 4.5 \cdot \mathcal{L}
\end{aligned}$$

In summary, this yields the approximation factor:

$$\max\left(\frac{\mathcal{L}}{OPT}, \frac{OPT}{\mathcal{L}}\right) \leq \max\left(\frac{\mathcal{L}}{\mathcal{U}}, \frac{\mathcal{U}}{\mathcal{L}}\right) \leq \frac{4.5 \cdot \mathcal{L}}{\mathcal{L}} = 4.5.$$

□

As presented in the proof of Theorem 1, we can guarantee better results for even values of k. Also, if k is odd the quality of the approximation increases for increasing values of k.

Remark 3

- If k is even, the result of Theorem 1 improves to a 4-approximation algorithm.
- For fixed odd values of k, Algorithm 1 is a polynomial time ρ-approximation algorithm for RKP with (cf. Equation (8)):

k	3	5	7	9	11	13	15	17	19
$\underline{\kappa}$	1	2	3	4	5	6	7	8	9
$\overline{\kappa}$	2	3	4	5	6	7	8	9	10
$\rho = 4 + \frac{1}{\overline{\kappa}} \cdot \frac{1}{\underline{\kappa}}$	$\frac{9}{2}$	$\frac{25}{6}$	$\frac{49}{12}$	$\frac{81}{20}$	$\frac{121}{30}$	$\frac{169}{42}$	$\frac{225}{56}$	$\frac{289}{72}$	$\frac{361}{90}$

However, in the worst case the approximation ratio is tight as is shown in the following example.

Example 3 Consider an instance of RKP with $n \geq 3k$, $M \in \mathbb{R}$, and coefficients

$$\begin{aligned} &a_1 = \ldots = a_k = M \quad a_{k+1} = \ldots = a_{n-k} = M-1 \quad a_{n-k+1} = \ldots = a_n = 1 \\ &b_1 = \ldots = b_k = 1 \quad\;\; b_{k+1} = \ldots = b_{n-k} = M-1 \quad b_{n-k+1} = \ldots = b_n = M, \end{aligned}$$

with

$$b_{\pi(i)} = \begin{cases} b_{n-k+i} & \text{for } i = 1, \ldots, k \\ b_i & \text{for } i = k+1, \ldots, n-k \\ b_{i-(n-k)} & \text{for } i = n-k+1, \ldots, n. \end{cases}$$

Algorithm 1 computes a lower bound solution with

$$\mathcal{L}_{\text{even}} = (\kappa \cdot M + \kappa \cdot 1)^2 = \frac{k^2}{4}(M+1)^2$$

for even values of k and

$$\begin{aligned} \mathcal{L}_{\text{odd}} &= (\overline{\kappa} \cdot M + \underline{\kappa} \cdot 1)(\underline{\kappa} \cdot M + \overline{\kappa} \cdot 1) \\ &= \frac{1}{4}\big((k^2-1)M^2 + 2(k^2+1)M + k^2 - 1\big). \end{aligned}$$

for odd values of k, respectively.

As one can easily see, one optimal solution is given by x^* with $x^*_{k+1} = \ldots = x^*_{2k} = 1$ and $x^*_1 = \ldots = x^*_k = x^*_{2k+1} = \ldots = x^*_n = 0$ and $f(x^*) = (k \cdot (M-1))^2 = k^2(M-1)^2$.

Thus, for increasing values of M the approximation ratio tends towards

$$\lim_{M\to\infty} \rho_{\text{even}} = \lim_{M\to\infty} \frac{f(x^*)}{\mathcal{L}_{\text{even}}} = \frac{k^2}{\frac{k^2}{4}} = 4$$

for even values of k and

$$\lim_{M\to\infty} \rho_{\text{odd}} = \lim_{M\to\infty} \frac{f(x^*)}{\mathcal{L}_{\text{odd}}} = \frac{k^2}{\frac{k^2-1}{4}} \leq 4.5$$

for odd values of $k \leq 3$, respectively. Note that, for fixed values of k, ρ_{odd} exactly matches the approximation ratios given in Remark 3.

4.1 Improvements of the approximation algorithm

In practice, Algorithm 1 can be improved in two different ways. A first observation is that, due to the definition of the lower bound solution $\tilde{x}$ (c.f. (2)), we do not use the full capacity of RKP, if the sets $\{1,\ldots,\overline{\kappa}\}$ and $\{\pi(1),\ldots,\pi(\underline{\kappa})\}$ are not disjoint, i. e., if $\sum_{i=1}^n \tilde{x}_i < k$. Hence, it is possible to increase the lower bound value by including further items. Algorithm 2 demonstrates a possible procedure to compute an improved lower bound $\mathcal{L}_{\text{impr}}$ that takes this into account.

An additional parameter k', which we call *adaptive capacity*, is introduced to increase the sets $\{1,\ldots,\overline{\kappa}\}$ and $\{\pi(1),\ldots,\pi(\underline{\kappa})\}$, and, therefore, increase the number of selected items, without violating the constraint. In the beginning, k' is set to k. After computing the lower bound solution $\tilde{x}$ as defined in (2), the algorithm tests whether k items are selected or not. In the latter case, the adaptive capacity k' is increased by the difference $k - \sum_{i=1}^n \tilde{x}_i$. A re-computation of $\tilde{x}$, using k' as capacity, allows to include more items in accordance with the ordering of the respective coefficients a_i or $b_{\pi(i)}$ which compensates for the fact that the original sets are not disjoint. Subsequently, it is tested again if the constraint is satisfied with equality. If not, the adaptive capacity k' is further increased. Otherwise, the algorithm continues by computing $\hat{x}$ using the current value of the parameter k' as capacity and testing which of the lower bound values is larger.

Lemma 3 *If the solution $\tilde{x}$ allows to increase the adaptive capacity k' to $k' + (k - \sum_{i=1}^n \tilde{x}_i)$ (in Step 10 of Algorithm 2), then this increase is also feasible for the computation of $\hat{x}$.*

Proof For ease of notation, we assume that we are examining the iteration where the adaptive capacity k' is increased for the first time from the capacity k to $k' = k + (k - \sum_{i=1}^n \tilde{x}_i)$. The following discussion can be applied in an

Algorithm 2 Improved approximation algorithm for RKP with adaptive capacity

Input: coefficients $a = (a_1, .., a_n)^\top$ sorted in non-increasing order, $b = (b_1, \dots, b_n)^\top$, capacity k

$\tilde{x} := \mathbf{0}_n$, $\hat{x} := \mathbf{0}_n$, stop$:= 0$, $k' := k$, $\overline{\kappa}' := \left\lceil \frac{k'}{2} \right\rceil$, $\underline{\kappa}' := \left\lfloor \frac{k'}{2} \right\rfloor$, $a := 1$

compute permutation $\pi \in \mathcal{S}_n$ such that

$$\begin{cases} b_{\pi(j)} > b_{\pi(j+1)}, \text{ or} \\ b_{\pi(j)} = b_{\pi(j+1)} \text{ and } a_{\pi(j)} \geq a_{\pi(j+1)} \end{cases} \quad \text{for } j = 1, \dots, n-1$$

while stop $= 0$ **do**
  for $i := a, \dots, \underline{\kappa}'$ **do** *//include further items*
    $\tilde{x}_i := 1$, $\tilde{x}_{\pi(i)} := 1$
  end for
  $\tilde{x}_{\overline{\kappa}'} := 1$
  if $\sum_{i=1}^{n} \tilde{x}_i < k$ **then** *//no equality in constraint*
    $a := \underline{\kappa}' + 1$
    $k' := k' + \left(k - \sum_{i=1}^{n} \tilde{x}_i\right)$ *//increase adaptive capacity k'*
    $\overline{\kappa}' := \left\lceil \frac{k'}{2} \right\rceil$, $\underline{\kappa}' := \left\lfloor \frac{k'}{2} \right\rfloor$
  else *//equality obtained*
    stop $:= 1$
  end if
end while
for $i := 1, \dots, \underline{\kappa}'$ **do** *//compute $\hat{x}$*
  $\hat{x}_i := 1$, $\hat{x}_{\pi(i)} := 1$
end for
$\hat{x}_{\pi(\overline{\kappa}')} := 1$
$\widetilde{\mathcal{L}} := (a^\top \tilde{x}) \cdot (b^\top \tilde{x})$
$\widehat{\mathcal{L}} := (a^\top \hat{x}) \cdot (b^\top \hat{x})$
if $\widetilde{\mathcal{L}} \geq \widehat{\mathcal{L}}$ **then**
  $\mathcal{L} := \widetilde{\mathcal{L}}$, $x := \tilde{x}$
else
  $\mathcal{L} := \widehat{\mathcal{L}}$, $x := \hat{x}$
end if

Output: lower bound $\mathcal{L}$ and corresponding solution x

analogous manner to all further iterations by adapting the notation accordingly.

If k is even, we know that $\tilde{x} = \hat{x}$ and the statement is trivially true. Otherwise, i. e., if k is odd, we can take advantage of the fact that the solution $\tilde{x}$ or $\hat{x}$ uses less than k items if:

- for $\tilde{x}$: $\{1, \dots, \overline{\kappa}\} \cap \{\pi(1), \dots, \pi(\underline{\kappa})\} \neq \emptyset$.
- for $\hat{x}$: $\{1, \dots, \underline{\kappa}\} \cap \{\pi(1), \dots, \pi(\overline{\kappa})\} \neq \emptyset$.

Therefore, we define

$$\begin{aligned} \tilde{\mathcal{I}} &:= \{1, \dots, \overline{\kappa}\} \cup \{\pi(1), \dots, \pi(\underline{\kappa})\}, \\ \hat{\mathcal{I}} &:= \{1, \dots, \underline{\kappa}\} \cup \{\pi(1), \dots, \pi(\overline{\kappa})\} \text{ and} \\ \mathcal{J} &:= \tilde{\mathcal{I}} \cap \hat{\mathcal{I}} = \{1, \dots, \underline{\kappa}\} \cup \{\pi(1), \dots, \pi(\underline{\kappa})\}. \end{aligned}$$

It holds that $\sum_{i=1}^{n} \tilde{x}_i = |\tilde{\mathcal{I}}|$ and that $\sum_{i=1}^{n} \hat{x}_i = |\hat{\mathcal{I}}|$. Furthermore, we know that

$$|\tilde{\mathcal{I}}| = \begin{cases} |\mathcal{J}| & \text{, if } \overline{\kappa} \in \{\pi(1), \ldots, \pi(\underline{\kappa})\}, \text{i, e., if } \tilde{\mathcal{I}} = \mathcal{J} \\ |\mathcal{J}| + 1 & \text{, else} \end{cases}.$$

Furthermore, we know that

$$|\hat{\mathcal{I}}| = \begin{cases} |\mathcal{J}| & \text{, if } \pi(\overline{\kappa}) \in \{1, \ldots, \underline{\kappa}\}, \text{i, e., if } \hat{\mathcal{I}} = \mathcal{J} \\ |\mathcal{J}| + 1 & \text{, else} \end{cases}$$

Considering these relations, we distinguish four cases:

Case 1: $|\tilde{\mathcal{I}}| = |\mathcal{J}|$ and $|\hat{\mathcal{I}}| = |\mathcal{J}|$
Thus, k' can be set to $k + (k - |\mathcal{J}|) = k + (k - |\tilde{\mathcal{I}}|)$ for $\tilde{x}$ and for $\hat{x}$.

Case 2: $|\tilde{\mathcal{I}}| = |\mathcal{J}| + 1$ and $|\hat{\mathcal{I}}| = |\mathcal{J}| + 1$
Thus, k' can be set to $k + (k - (|\mathcal{J}| + 1)) = k + (k - |\tilde{\mathcal{I}}|)$ for $\tilde{x}$ and for $\hat{x}$.

Case 3: $|\tilde{\mathcal{I}}| = |\mathcal{J}| + 1$ and $|\hat{\mathcal{I}}| = |\mathcal{J}|$
For $\hat{x}$, k' can be set to $k + (k - |\mathcal{J}|) = k + (k - |\hat{\mathcal{I}}|)$. The use of $\tilde{x}$ in Step 10 of Algorithm 2 leads to $k' = k + (k - |\tilde{\mathcal{I}}|) = k + (k - (|\mathcal{J}| + 1)) < k + (k - |\mathcal{J}|)$ which is feasible for $\hat{x}$.

Case 4: $|\tilde{\mathcal{I}}| = |\mathcal{J}|$ and $|\hat{\mathcal{I}}| = |\mathcal{J}| + 1$
Since $|\tilde{\mathcal{I}}| = |\mathcal{J}|$, we know that $\overline{\kappa} \in \{\pi(1), \ldots, \pi(\underline{\kappa})\}$ $(*)$. In a first iteration we examine the consequences of setting $k' := k + 1$. Thus, k' is even and we define the corresponding solution as:

$$x'_i = \begin{cases} 1 & \text{, for } i \in \{1, \ldots, \overline{\kappa}\} \cup \{\pi(1), \ldots, \pi(\overline{\kappa})\} \\ 0 & \text{, otherwise} \end{cases},$$

where $\{1, \ldots, \overline{\kappa}\} \cup \{\pi(1), \ldots, \pi(\overline{\kappa})\} \overset{(*)}{=} \{1, \ldots, \underline{\kappa}\} \cup \{\pi(1), \ldots, \pi(\overline{\kappa})\} = \hat{\mathcal{I}}$. Thus, setting the adaptive capacity k' to $k + 1$ does not change $\hat{x}$, i. e., $x' = \hat{x}$.
Hence, k' can be set to

$$k + 1 + (k - (|\hat{\mathcal{I}}| + 1)) = k + (k - |\mathcal{J}|) = k + (k - |\tilde{\mathcal{I}}|)$$

for $\tilde{x}$ and for $\hat{x}$.

□

Lemma 4 *Let n be the number of items and let k be the capacity of RKP. Algorithm 2 terminates, has a worst case time complexity of $\mathcal{O}(n \log n)$, and a worst case approximation ratio of* 4.5.

Proof The while-loop for computing the solution $\tilde{x}$ with $\sum_{i=1}^{n} \tilde{x}_i = k$ is critical for the termination of Algorithm 2. In the first iteration, at least $\overline{\kappa}$ variables are set to 1. The parameter k' is increased by at least 1 in each consecutive iteration and, thus, in at least every second iteration an additional entry of $\tilde{x}$

is set to 1. Hence, after at most $2 \cdot \underline{\kappa} + 1 = k$ iterations k variables have been selected for $\tilde{x}$ and the loop terminates.

We take advantage of the ordering of the coefficients to set only new variables to 1 if the adaptive capacity is increased. Thus, the execution of the while loop requires $\mathcal{O}(k)$. The complexity of Algorithm 2 is determined by the sorting algorithm (cf. Algorithm 1), i. e., Algorithm 2 has a worst case time complexity of $\mathcal{O}(n \log n)$.

The approximation ratio is at most 4.5, since Algorithm 2 computes the same optimal objective function value for the RKP instance of Example 3 as Algorithm 1. □

Example 4 We apply the improved approximation algorithm, Algorithm 2, on the instance of RKP of Example 2. The solution $\tilde{x}$ is defined by the set

$$\tilde{\mathcal{I}} = \{1,2,3\} \cup \{\pi(1), \pi(2)\} = \{1,2,3\} \cup \{2,4\} = \{1,2,3,4\}$$

with $|\tilde{\mathcal{I}}| = \sum_{i=1}^{n} \tilde{x}_i = 4 < 5$. Thus, the adaptive capacity can be set to $k' := 5 + (5-4) = 6$. The re-computation of $\tilde{x}$ leads to

$$\tilde{\mathcal{I}} = \{1,2,3\} \cup \{\pi(1), \pi(2), \pi(3)\} = \{1,2,3\} \cup \{2,4,6\} = \{1,2,3,4,6\}$$

with $|\tilde{\mathcal{I}}| = \sum_{i=1}^{n} \tilde{x}_i = 5$. Hence, the cardinality constraint is tight and, since k' is even, the solution $x = (1,1,1,1,0,1,0,0)^\top$ generates the improved lower bound $\mathcal{L} = f(x) = 920$. The optimal solution of the instance x^* is identical to the improved lower bound solution $x^* = x$.

As proven above, setting the adaptive capacity to $k' := 6$ is also feasible for $\hat{x}$. The solution $\hat{x}$ defined by $k' = 5$ corresponds to the set

$$\hat{\mathcal{I}} = \{1,2\} \cup \{\pi(1), \pi(2), \pi(3)\} = \{1,2\} \cup \{2,4,6\} = \{1,2,4,6\}$$

with $|\hat{\mathcal{I}}| = \sum_{i=1}^{n} \hat{x}_i = 4 < 5$. Hence, one additional item can be included, resulting again in the same lower bound solution $x = (1,1,1,1,0,1,0,0)^\top$ (with $k' = 6$). The areas of rectangles corresponding to the lower bounds $\widetilde{\mathcal{L}}$ and $\widehat{\mathcal{L}}$ based on the first computations of $\tilde{x}$ and $\hat{x}$ (Algorithm 1), respectively, and the improved lower bound $\mathcal{L}$ (Algorithm 2) are shown in Figure 4.

A second improvement of Algorithm 1 is motivated differently: Without an analysis of the input data, the distribution of the entries of the coefficient vectors a and b is unknown, and, thus, there might be better selections then deciding equally according to both of the orderings. One possible approach is to compute various alternative solutions, still based on the sorting of the coefficients, and select the best solution. The alternatives can be defined by setting the variables only corresponding to the sorting of a, to the sorting of b, and by all alternatives in between, i. e., $x_1 = \ldots = x_j = 1$, $x_{\pi(1)} = \ldots = x_{\pi(k-j)} = 1$ and $x_i = 0$ for all remaining indices, for $0 \leq j \leq k$.

This approach is formalized in Algorithm 3. Following the idea of Algorithm 2, it includes a test if k items have been selected in the current solution. If not, further items are included according to the ordering of the coefficients $b_{\pi(i)}$. Other strategies are possible: include further items according to

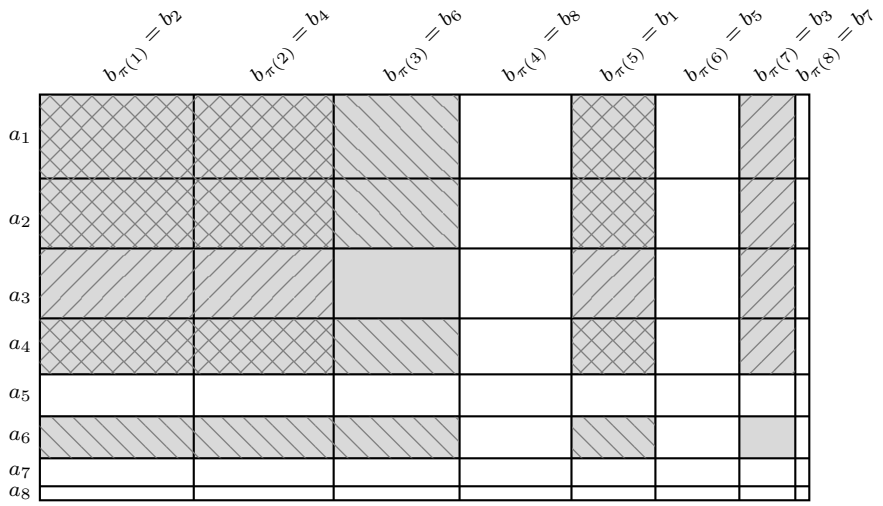

Fig. 4 Lower bounds $\widetilde{\mathcal{L}}$ (▨) , $\widehat{\mathcal{L}}$ (▧) and $\mathcal{L}$ (▩) for Examples 2 and 4.

the ordering of the coefficients a_i, alternate between both orderings or include arbitrarily chosen items. The quality of those strategies strongly depends on the given problem instance.

In practice, it is quite intuitive to assume that Algorithm 3 leads to better approximation results than that of the basic version of Algorithm 1. However, the theoretical approximation ratio is the same.

Theorem 2 *Algorithm 3 is a polynomial time* 4.5*-approximation algorithm for the rectangular knapsack problem.*

Proof Algorithm 3 returns a feasible solution in polynomial time, since each alternative solution $\hat{x}$ and the corresponding objective function value can be computed in linear time and there are linearly many alternatives (*c. f.* Theorem 1).

The approximation ratio is at least 4.5, since the solutions $\tilde{x}$ and $\hat{x}$ of Algorithm 1 are included in the set of alternatives. The approximation ratio is at most 4.5, since Algorithm 3 computes the same optimal objective function value for the RKP instance of Example 3 as Algorithm 1. □

5 Computational experiments

In this section, the quality of the several variants of the approximation algorithm is evaluated experimentally on a wide range of RKP instances. We implemented the basic variant of the approximation algorithm (Algorithm 1), the improved variants with adaptive capacity (Algorithm 2), the shifted selection, and the combined version of these two improvements (Algorithm 3); we will refer to the solution quality returned by these variants as $\mathcal{L}_{\text{basic}}$, $\mathcal{L}_{\text{impr}}$,

Algorithm 3 Improved approximation algorithm for RKP with shifted selection wrt. a and b and adaptive capacity

Input: coefficients $a = (a_1, \ldots, a_n)^\top$ sorted in non-increasing order, $b = (b_1, \ldots, b_n)^\top$, capacity k

$x := \mathbf{0}_n$ and $\mathcal{L} := 0$
compute permutation $\pi \in \mathcal{S}_n$ such that

$$\begin{cases} b_{\pi(j)} > b_{\pi(j+1)}, \text{ or} \\ b_{\pi(j)} = b_{\pi(j+1)} \text{ and } a_{\pi(j)} \geq a_{\pi(j+1)} \end{cases} \quad \text{for } j = 1, \ldots, n-1$$

for $j := 0, \ldots, k$ **do**
  $\hat{x} := \mathbf{0}_n$
  for $i := 1, \ldots, k - j$ **do**
    $\hat{x}_i := 1$
  end for
  for $i := 1, \ldots, j$ **do**
    $\hat{x}_{\pi(i)} := 1$
  end for
  $j' := j + 1$
  while $\sum_{i=1}^{n} \hat{x}_i < k$ **do**
    $\hat{x}_{\pi(j')} := 1$
    $j' := j' + 1$
  end while
  $\widehat{\mathcal{L}} := (a^\top \hat{x}) \cdot (b^\top \hat{x})$
  if $\widehat{\mathcal{L}} > \mathcal{L}$ **then**
    $\mathcal{L} := \widehat{\mathcal{L}}$, $x := \hat{x}$
  end if
end for

Output: lower bound $\mathcal{L}$ for RKP and corresponding solution x

$\mathcal{L}_{\text{shift}}$, $\mathcal{L}_{\text{comb}}$, respectively. All algorithm variants were implemented in C. The QKP solver by Caprara et al. [1999] was used to compute the optimal solutions of RKP [see Pisinger, 2016]. The computational experiments were performed on an Intel Quadcore 3.2 GHz with 4 GB RAM running Linux compiled with gcc 4.8.

Three different classes of instances were generated to test the algorithms:

Uncorrelated instances The coefficients a_i, b_i are generated according to a uniform distribution within the range $[0, 100]$.

Positively correlated instances The coefficients a_i are generated according to a uniform distribution within the range $[0, 100]$ and $b_i = a_i + n(i)$ where $n(i)$ is a value generated according to a uniform distribution within the range $[-5, 5]$.

Negatively correlated instances The coefficients a_i are generated according to a uniform distribution within the range $[0, 100]$ and $b_i = \max\{100 - a_i + n(i),\ 0\}$ where $n(i)$ is a value generated according to a uniform distribution within the range $[-5, 5]$.

For each type of instances, four different constraint slacknesses c_k, with $k = \lfloor c_k \cdot n \rfloor$, were chosen: $c_k = 0.1$, $c_k = 0.25$, $c_k = 0.5$, and $c_k = 0.75$. The instance sizes were $n = 100, 200, 300, 400$, except for the negatively correlated instances, with $n = 25, 50, 75$. For the latter instance class, the QKP solver was not able to solve instances with $n \geq 75$ and $k \geq 14$ within one hour of CPU-time. For each combination of instance class, size and constraint slackness, 10 instances were generated. Noteworthy, all approximation algorithms required at most 0.01 seconds for all instances tested.

n	c_k	$z^*/\mathcal{L}_{\text{basic}}$	$z^*/\mathcal{L}_{\text{impr}}$	$z^*/\mathcal{L}_{\text{shift}}$	$z^*/\mathcal{L}_{\text{comb}}$	$\mathcal{U}/\mathcal{L}^*$
100	0.10	1.37	1.32	1.24	1.22	1.50
	0.25	1.34	1.18	1.18	1.17	1.52
	0.50	1.37	1.06	1.14	1.12	1.32
	0.75	1.36	1.02	1.08	1.08	1.14
200	0.10	1.41	1.32	1.28	1.26	1.56
	0.25	1.35	1.17	1.24	1.20	1.52
	0.50	1.33	1.07	1.16	1.14	1.34
	0.75	1.37	1.02	1.09	1.09	1.14
300	0.10	1.38	1.32	1.27	1.26	1.55
	0.25	1.34	1.17	1.25	1.20	1.52
	0.50	1.35	1.06	1.15	1.14	1.32
	0.75	1.40	1.02	1.10	1.09	1.14
400	0.10	1.45	1.33	1.35	1.30	1.61
	0.25	1.35	1.18	1.26	1.20	1.55
	0.50	1.33	1.06	1.16	1.15	1.33
	0.75	1.38	1.01	1.10	1.10	1.14

Table 1 Results for uncorrelated instances.

n	c_k	$z^*/\mathcal{L}_{\text{basic}}$	$z^*/\mathcal{L}_{\text{impr}}$	$z^*/\mathcal{L}_{\text{shift}}$	$z^*/\mathcal{L}_{\text{comb}}$	$\mathcal{U}/\mathcal{L}^*$
100	0.10	2.85	1.00	1.00	1.00	1.00
	0.25	2.74	1.00	1.00	1.00	1.00
	0.50	2.69	1.00	1.00	1.00	1.00
	0.75	2.28	1.00	1.00	1.00	1.00
200	0.10	2.78	1.00	1.00	1.00	1.00
	0.25	2.91	1.00	1.00	1.00	1.00
	0.50	2.74	1.00	1.00	1.00	1.00
	0.75	2.29	1.00	1.00	1.00	1.00
300	0.10	2.50	1.00	1.00	1.00	1.00
	0.25	2.95	1.00	1.00	1.00	1.00
	0.50	2.79	1.00	1.00	1.00	1.00
	0.75	2.29	1.00	1.00	1.00	1.00
400	0.10	2.83	1.00	1.00	1.00	1.00
	0.25	2.97	1.00	1.00	1.00	1.00
	0.50	2.71	1.00	1.00	1.00	1.00
	0.75	2.28	1.00	1.00	1.00	1.00

Table 2 Results for positively correlated instances.

Tables 1 to 3 present the average results obtained for the three classes of instances where columns $z^*/\mathcal{L}_\bullet$ refer to the average approximation ratios obtained by the four algorithm variants and column $\mathcal{U}/\mathcal{L}^*$ gives an upper bound on the approximation ratio. In general, the results indicate that the approximation quality of all algorithm variants is much better than the guaranteed approximation ratio of 4.5 and that the improved versions yield even better

n	c_k	$z^*/\mathcal{L}_{\text{basic}}$	$z^*/\mathcal{L}_{\text{impr}}$	$z^*/\mathcal{L}_{\text{shift}}$	$z^*/\mathcal{L}_{\text{comb}}$	$\mathcal{U}/\mathcal{L}^*$
25	0.10	1.06	1.06	1.06	1.06	3.62
	0.25	1.06	1.06	1.06	1.06	3.13
	0.50	1.04	1.04	1.04	1.04	2.33
	0.75	1.02	1.02	1.02	1.02	1.62
50	0.10	1.08	1.08	1.08	1.08	3.56
	0.25	1.06	1.06	1.06	1.06	3.06
	0.50	1.04	1.04	1.04	1.04	2.26
	0.75	1.02	1.02	1.02	1.02	1.57
75	0.10	1.08	1.08	1.08	1.08	3.56

Table 3 Results for negatively correlated instances.

results than the basic variant, except on negatively correlated instances, for which all versions presented a similar performance. Moreover, the instance size does not play a strong role on the approximation ratio. However, the performance of the four variants seem to be affected by the instance type. In the following, we discuss the results in more detail for each instance type.

Uncorrelated instances The experimental results in Table 1 suggest that the improved variant performs better as the constraint slackness increases and that a larger capacity value k improves the approximation (see Remark 3). Differently, the basic variant does not seem to be affected by c_k and presents the worst approximation ratio in all cases. The shifted and combined variants present the best approximation ratio for small c_k. Both variants also improve the approximation for larger c_k but not as much as for the improved variant, which gives the best approximation ratio.

Positively correlated instances Table 2 shows that the basic variant has the worst approximation ratio, although still far from the theoretical bound. For this variant, many items seem to be selected twice due to both orderings, which can be expected for positive correlated instances since the orderings of the coefficients to both objective functions should be rather similar. Noteworthy, all the three improved variants are close to an approximation ratio of 1.0. We observed that as soon as the equality in the constraint is ensured, all improved variants solve the problem to optimality.

Negatively correlated instances For this instance type, all variants present a similar approximation ratio; see Table 3. In fact, the basic variant generates very good approximation results with approximation ratios close to 1.0 for the tested instances. This is especially good, since the exact algorithm could not solve larger instances in less then one hour of CPU-time whereas the approximation algorithm can compute a high quality approximation in less then 0.01 seconds.

6 Conclusion

We presented a geometric interpretation of the rectangular knapsack problem. Upper and lower bounds for the problem can be computed directly by sorting the coefficients of the objective function.

Based on these bound computations, we introduced a polynomial time approximation algorithm for RKP that provides an approximation ratio of 4.5. In practice, however, the algorithm can be further improved by selecting additional items if the cardinality constraint is not met with equality. Furthermore, the selection strategy for items can be modified

We tested all algorithm variants on knapsack instances with three different correlation structures, up to 400 items, and four different constraint slacknesses. The approximations were computed in 0.01 seconds or less per instance. We observed that in practice the approximation ratios of all algorithms are much better than the theoretical ratio of 4.5. Thus, our approximation algorithms are an efficient tool to compute approximations of good quality for RKP.

In the future it would be interesting to integrate the bound computations in a branch-and-bound procedure to formulate an exact algorithm for RKP. Furthermore, the results seem to be transferable to higher dimensions, where we think of problems of the form

$$\begin{aligned} \max \quad & f(x) = \prod_{j=1}^{m} \sum_{i=1}^{n} p_i^j \, x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i \leq k \\ & x_i \in \{0,1\}, \qquad i = 1, \ldots, n. \end{aligned}$$

The bound computations and algorithm formulations should be convertible without problems, whereas the proof of an approximation ratio may become more complicated due to more possible cases that may occur.

We also suggested a field of application for RKP. Finding a representative solution of the bi-objective cardinality constrained knapsack problem that maximizes the hypervolume with the origin as reference point is modeled by the rectangular knapsack problem. It is, therefore, very interesting for future research.

Acknowledgements This work was supported by the bilateral cooperation project *Multiobjective Network Optimization for Engineering and Management Support* funded by the Deutscher Akademischer Austauschdienst (DAAD, Project-ID 57128839) and Fundação para a Ciência e Tecnologia. Stefan Ruzika gratefully acknowledges support by DFG grant RU 1524/4-1.

References

A. Billionnet and F. Calmels. Linear programming for the 01 quadratic knapsack problem. *European Journal of Operational Research*, 92(2):310–325, 1996.

A. Billionnet and E. Soutif. An exact method based on Lagrangian decomposition for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 157:565–575, 2004.

A. Billionnet, A. Faye, and E. Soutif. A new upper bound for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 112:664–672, 1999.

H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

R. Burkard, E. Çela, P. Pardalos, and L. Pitsoulis. The quadratic assignment problem. *Handbook of Combinatorial Optimization*, 3, 1998.

A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, second edition, 2001.

G. Gallo, P. Hammer, and B. Simeone. Quadratic knapsack problems. In M. Padberg, editor, *Combinatorial Optimization*, volume 12 of *Mathematical Programming Studies*, pages 132–149. Springer, Berlin Heidelberg, 1980.

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

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

E. L. Johnson, A. Mehrotra, and G. L. Nemhauser. Min-cut clustering. *Mathematical Programming*, 62(1):133–151, 1993.

H. Kellerer and V. A. Strusevich. Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica*, 57(4):769–795, 2010.

T. Kuhn, C. M. Fonseca, L. Paquete, S. Ruzika, M. M. Duarte, and J. R. Figueira. Hypervolume subset selection in two dimensions: Formulations and algorithms. *Evolutionary computation*, 24:411–425, 2016. ISSN 1530-9304. doi: 10.1162/EVCO_a_00157.

U. Pferschy and J. Schauer. Approximation of the quadratic knapsack problem. *INFORMS Journal on Computing*, 28(2):308–318, 2016.

D. Pisinger. The quadratic knapsack problem – a survey. *Discrete Applied Mathematics*, 155(5):623–648, 2007.

D. Pisinger. Exact algorithm for the quadratic knapsack problem and instance generator, October 2016. URL `http://www.diku.dk/~pisinger/codes.html`.

D. W. Pisinger, A. B. Rasmussen, and R. Sandvik. Solution of large quadratic knapsack problems through aggressive reduction. *INFORMS Journal on*

Computing, 19(2):280–290, 2007.

D. J. Rader, Jr. and G. J. Woeginger. The quadratic 0–1 knapsack problem with series-parallel support. *Operations Research Letters*, 30(3):159–166, 2002.

J. M. W. Rhys. A selection problem of shared fixed costs and network flows. *Management Science*, 17(3):200–207, 1970.

C. D. Rodrigues, D. Quadri, P. Michelon, and S. Gueye. 0–1 quadratic knapsack problems: An exact approach based an a t-linearization. *SIAM Journal of Optimization*, 22(4):1449–1468, 2012.

R. Taylor. Approximation of the quadratic knapsack problem. *Operations Research Letters*, 44(4):495–497, 2016.

C. Witzgall. Mathematical models of site selection of electronic message systems (EMS). Technical report, National Bureau of Standards, Washington, DC., 1975.

Z. Xu. A strongly polynomial FPTAS for the symmetric quadratic knapsack problem. *European Journal of Operational Research*, 218(2):377–381, 2012.

E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V, 5th International Conference Amsterdam, The Netherlands, September 27-30, 1998, Proceedings*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer, Berlin Heidelberg, 1998.