Mathematical Methods of Operations Research
https://doi.org/10.1007/s00186-022-00788-8

ORIGINAL ARTICLE

On the rectangular knapsack problem

Fritz Bökler[1] · **Markus Chimani**[1] · **Mirko H. Wagner**[1]

Received: 17 November 2021 / Revised: 2 May 2022 / Accepted: 5 May 2022

Abstract
A recent paper by Schulze et al. (Math Methods Oper Res 92(1):107–132, 2020) presented the Rectangular Knapsack Problem (RKP) as a crucial subproblem in the study on the Cardinality-constrained Bi-objective Knapsack Problem (CBKP). To this end, they started an investigation into its complexity and approximability. The key results are an **NP**-hardness proof for a more general scenario than RKP, and a 4.5-approximation for RKP, raising the question of improvements for either result. In this note we settle both questions conclusively: we show that (a) RKP is indeed **NP**-hard in the considered setting (and even in more restricted settings), and (b) there exists both a pseudopolynomial algorithm and a fully-polynomial time approximation scheme (i.e., efficient approximability within any desired ratio $\alpha > 1$) for RKP.

Keywords Quadratic optimization · Knapsack problems · Multiobjective optimization · Approximation

1 Introduction

We mainly consider the Rectangular Knapsack Problem:

Definition 1 Given $a, b \in \mathbb{N}^n$ and $\kappa \in \mathbb{N}$, the *Rectangular Knapsack Problem* (RKP) is formulated as

$$\max \; f(x) := \sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j x_i x_j = \left(a^\mathsf{T} x\right)\left(b^\mathsf{T} x\right)$$

✉ Fritz Bökler
fboekler@uos.de
http://tcs.uos.de/staff/boekler

Markus Chimani
markus.chimani@uos.de
http://tcs.uos.de/

Mirko H. Wagner
mwagner@uos.de

[1] Department of Computer Science, Osnabrück University, Osnabrück, Germany

Published online: 09 June 2022

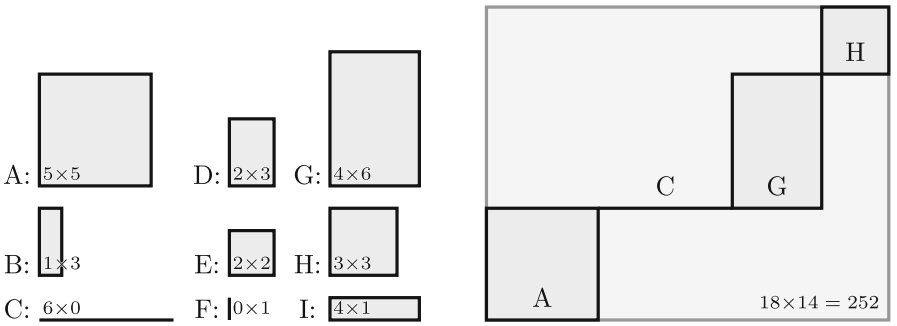

Fig. 1 A graphical representation of an RKP instance: Given a multiset of (possibly degenerated) rectangles and an integer κ (left), we ask for a subset of cardinality κ that spans the largest possible area (right). Here, we are given 9 rectangles (A,B,... I) with $\kappa = 4$

$$\begin{aligned} \text{s.t. } \mathbf{1}^\mathsf{T} x &\leq \kappa \\ x &\in \{0,1\}^n, \end{aligned}$$

where $\mathbf{1}$ is the all-one vector. We also define a decision version of this problem, where we are additionally given a $\theta \in \mathbb{N}$, and ask whether there exists a feasible $x \in \{0,1\}^n$ with $f(x) \geq \theta$. ◁

RKP allows a nice geometric interpretation, see Fig. 1: We are given an integer $\kappa \in \mathbb{N}$ and a multiset $R = \{(a_i, b_i)\}_{i=1,\dots,n}$ of rectangles, specified by their width and height. The rectangles are allowed to be degenerated to orthogonal line segments (i.e., have width or height 0). For a subset of rectangles, we may lay them out in the plane connectedly such that the (axis-parallel) bounding-box has maximum area. Clearly, this is achieved by placing the rectangles in a linear sequence such that the bottom-left corner of a rectangle is the top-right corner of its predecessor. In RKP we thus ask for a rectangle subset of cardinality at most κ that maximizes this area.

RKP arises naturally in the study of the Cardinality-constrained Bi-objective Knapsack problem (CBKP), as observed by Schulze et al. (2020):

Definition 2 Given $a, b \in \mathbb{N}^n$ and $\kappa \in \mathbb{N}$, the *Cardinality-constrained Bi-objective Knapsack Problem* (CBKP) is formulated as

$$\begin{aligned} \max\ & a^\mathsf{T} x \\ \max\ & b^\mathsf{T} x \\ \text{s.t. } \mathbf{1}^\mathsf{T} x &\leq \kappa \\ x &\in \{0,1\}^n. \end{aligned}$$

◁

In this problem, we ask for the Pareto-front of two linear profit functions, subject to a cardinality constraint. As the Pareto-front can be exponential in the size of the input, Schulze et al. (2020) consider approximative methods to find a good respresentation of it; this can be achieved by iteratively solving RKP instances. The approach has also been evaluated experimentally by Paquete et al. (2022). Schulze et al. (2020) give

a 4.5-approximation for RKP and conjecture that it is **NP**-hard. However, they can only show **NP**-hardness of two more general cases of the RKP, namely when the a and b vectors may consist of general integer values (i.e., negative components are allowed) or when the cardinality constraint is a more general Knapsack constraint. The complexity status of the actual RKP remained open.

RKP is a special case of the *Quadratic Knapsack Problem* (QKP) first introduced by Gallo et al. (1980); see (Pisinger 2007) for a comprehensive survey. Thereby we want to optimize a quadratic function over binary variables, i.e., $f^q(x) = x^\mathsf{T} Q x$, subject to a knapsack constraint. QKP is known to be *strongly* **NP**-complete even if the Knapsack constraint is a cardinality constraint, by reduction from CLIQUE (Garey and Johnson 1979). Recall that *strong* **NP**-hardness means that the problem remains **NP**-hard even when all *values* of the numbers in the input are bounded by a polynomial in the input size; thus, such problems do not allow a pseudopolynomial algorithm unless **P=NP**.

QKP with a general integral weight matrix Q does not allow a constant-factor approximation unless **P=NP** (Rader and Woeginger 2002). For non-negative Q, a constant-factor approximation can currently not be ruled out, but the best known approximation guarantees only a ratio of $\mathcal{O}\left(n^{2/5+\varepsilon}\right)$ in $\mathcal{O}\left(n^{9/\varepsilon}\right)$ time (Taylor 2016). Recall that, for a given quality requirement $\varepsilon > 0$, a (fully) polynomial-time approximation scheme, abbreviated as PTAS (FPTAS), achieves a $(1-\varepsilon)$-approximation while its running time is bounded by a polynomial in the input size and an arbitrary (polynomial, respectively) function in ε^{-1}. A weight-matrix Q induces an n-vertex graph G that has an edge (ij) if and only if $Q_{i,j} \neq 0$. There is an FPTAS if G has bounded tree-width and a PTAS if G is H-minor-free, for any fixed minor H (see (Pferschy and Schauer 2016) for both results). However, the problem remains strongly **NP**-hard even if G is guaranteed to be 3-book-embeddable (Pferschy and Schauer 2016) or vertex series-parallel (Rader and Woeginger 2002). Furthermore, an FPTAS exists for a special symmetric quadratic knapsack problem, where the knapsack constraint coefficients are dependent on the matrix Q (Kellerer and Strusevich 2010; Xu 2012).

Our contribution. Our first result is to settle **NP**-hardness for RKP in Sect. 3.

Next, we show that we can use an FPTAS from the literature to construct an FPTAS for RKP in Sect. 4, concluding that RKP is in fact only weakly **NP**-hard. Interestingly, we multi-objectivize the RKP, i.e., turn the RKP into a multi-objective optimization problem, to apply an FPTAS from the muliobjective literature.

In Sect. 5, we describe an exact pseudopolynomial time algorithm for RKP. Our algorithm can also be used to directly and exactly solve the original CBKP, the starting point of the investigation of RKP by Schulze et al. (2020). Finally, in Sect. 6 we show how to use our algorithm as a building block to develop an FPTAS for RKP and CBKP. We show that both our pseudopolynomial algorithm and FPTAS yield improved running times compared to the FPTAS used in Sect. 4.

2 Preliminaries

Let an RKP instance be given by $a, b \in \mathbb{N}^n$ and $\kappa \in \mathbb{N}$. For $x \in \mathbb{N}^n$, we have $\|x\|_\infty := \max_{i=1,\ldots,n} x_i$. Let $\alpha := \|a\|_\infty$, $\beta := \|b\|_\infty$, $W^\uparrow := \max\{\alpha, \beta\}$, and

$W^{\downarrow} := \min\{\alpha, \beta\}$. W.l.o.g., we assume $\alpha \geq \beta$ and, thus, $W^{\uparrow} = \alpha$ and $W^{\downarrow} = \beta$. Let N be the encoding length of the RKP instance; we observe in particular that $n \leq N$ and $\log \alpha \leq N$. We may use these observations when showing running time bounds depending solely on the input length. Whenever we compare two vectors, e.g. $x \leq y$ for $x, y \in \mathbb{N}^n$, this is understood to be component-wise.

Consider a more general version of the CBKP: the multiobjective Knapsack problem with general knapsack constraints and an arbitrary number of objective functions.

Definition 3 Given $C \in \mathbb{N}^{d \times n}$, $w \in \mathbb{N}^n$, and $\kappa \in \mathbb{N}$, the *Multiobjective Knapsack Problem* (MOKP) is formulated as

$$\begin{aligned} &\max\ Cx \\ \text{s.t.}\quad &w^{\mathsf{T}}x \leq \kappa \\ &x \in \{0,1\}^n. \end{aligned}$$

◁

The rows $c^1, \ldots, c^d \in \mathbb{N}^n$ of the objective function matrix C can be seen as individual objective functions $(c^i)^{\mathsf{T}}x$ meant to be maximized. A vector x *dominates* another vector y, if $x \geq y$ and $x \neq y$. Let $Y := \left\{Cx : w^{\mathsf{T}}x \leq \kappa, x \in \{0,1\}^n\right\}$ be the set of *value vectors* of an MOKP instance. A value vector $y \in Y$ is *non-dominated* if there is no other $y' \in Y$ that dominates it. We call the set of non-dominated value vectors the *non-dominated set* or Y_N. A solution $x \in \{0,1\}^n$ is *Pareto-optimal*, if $Cx \in Y_N$. Computing the non-dominated set for MOKP is known to be **NP**-hard, since it contains the single-objective knapsack problem as a special case.

An FPTAS for a multiobjective problem is an algorithm that computes, for any given quality guarantee $\varepsilon \in (0,1)$, a set $S \subseteq Y$, such that for every $y \in Y_N$, there is an $x \in S$ with $x \geq (1-\varepsilon)y$ in time polynomial in the input size and $1/\varepsilon$. While for many multiobjective optimization problems it is known that the size of Y_N can be exponential in the input size, e.g., multiobjective versions of the shortest path, spanning tree, or knapsack problems, there also always exists a set S as above of size bounded polynomially in the input size and $1/\varepsilon$ (cf. Papadimitriou and Yannakakis (2000)).

3 NP-Hardness

Before we proceed with the hardness proof, we start with an initial observation. Consider an arbitrary RKP instance and any two vectors $x, x' \in \{0,1\}^n$. If $x' \leq x$, we have $f(x') \leq f(x)$ since $a, b \geq \mathbf{0}$. We can therefore deduce:

Observation 4 *If there is an optimal solution (or a witness to a yes-instance) x' with $\mathbf{1}^T x' < \kappa$ for some* RKP *instance, then there also exists another optimal solution (yes-witness) x with $\mathbf{1}^T x = \kappa$. It thus suffices to search for solutions with the latter property.*

The next problem is known to be (weakly) **NP**-hard for decades (Garey and Johnson 1979). Intuitively, it asks whether we can partition the components of a vector c into two subsets of equal size such that the sum of the components of each subsets coincide.

Definition 5 (Cardinality- Constraint Partition (Ccp)) Given a vector $c \in \mathbb{N}^n$, is there a vector $x \in \{0, 1\}^n$ such that $\mathbf{1}^\mathsf{T} x = n/2$ and $c^\mathsf{T} x = c^\mathsf{T}(\mathbf{1} - x)$? ◁

Ccp was also used by Schulze et al. (2020) to show **NP**-hardness for the problem that arises from Rkp problem when a, b are allowed to include negative components. However, for the true Rkp (which was tackled by the algorithmic approximation by Schulze et al. (2020)), no hardness result was shown. By massaging Ccp-instances further, we can in fact show **NP**-hardness for strictly non-negative (or even strictly positive) vectors a, b.

Theorem 6 Rkp *is* ***NP****-hard.*

Proof We reduce from Ccp to the decision variant of Rkp, establishing that the latter is **NP**-complete as **NP** membership is trivial. Let $c \in \mathbb{N}^n$ be a Ccp instance. We define the sum of all c-components $C := \mathbf{1}^\mathsf{T} c$ and a sufficiently large constant $M := \|c\|_\infty \cdot n/2$. Let $\mathbf{C} := C \cdot \mathbf{1}$ and $\mathbf{M} := M \cdot \mathbf{1}$ be the n-dimensional vectors with uniform components C and M, respectively. We construct an Rkp instance (a, b, κ, θ) with

$$\begin{aligned} \kappa &:= n/2, \\ a &:= \mathbf{M} + \kappa \cdot c, \\ b &:= \mathbf{M} + \mathbf{C} - \kappa \cdot c, \\ \theta &:= \kappa^2 \cdot \left(M^2 + CM + (C/2)^2 \right). \end{aligned}$$

We can assume w.l.o.g. that n and C are even numbers (as c would otherwise be a trivial no-instance for Ccp) and non-zero. Thus we have integrality for all our constructed numbers. Since $c \in \mathbb{N}^n$, all constructed numbers (except for perhaps the components of b) are trivially positive and thus elements of $\mathbb{N}_+$. Since $M = \kappa \|c\|_\infty$ is at least as large as any component of $\kappa \cdot c$, also every component of b is in fact positive and thus from $\mathbb{N}_+$. The tuple (a, b, κ, θ) is hence a legal Rkp instance.

We now show that (a, b, κ, θ) is an Rkp yes-instance if and only if the original $c \in \mathbb{N}^n$ is a Ccp yes-instance by proving the equivalence directly. Assume (a, b, κ, θ) is a yes-instance for Rkp, and consider a solution vector $\bar{x} \in \{0, 1\}^n$. By Observation 4, we can assume w.l.o.g. that $\mathbf{1}^\mathsf{T} \bar{x} = \kappa$ and thus $\mathbf{M}^\mathsf{T} \bar{x} = \kappa M$ and $\mathbf{C}^\mathsf{T} \bar{x} = \kappa C = \kappa \cdot c^\mathsf{T} \mathbf{1}$. We examine the objective $f(\bar{x})$:

$$\begin{aligned} f(\bar{x}) &= a^\mathsf{T} \bar{x} \cdot b^\mathsf{T} \bar{x} \\ &= (\mathbf{M} + \kappa \cdot c)^\mathsf{T} \bar{x} \cdot (\mathbf{M} + \mathbf{C} - \kappa \cdot c)^\mathsf{T} \bar{x} \\ &= (\mathbf{M}^\mathsf{T} \bar{x} + \kappa \cdot c^\mathsf{T} \bar{x}) \cdot (\mathbf{M}^\mathsf{T} \bar{x} + \mathbf{C}^\mathsf{T} \bar{x} - \kappa \cdot c^\mathsf{T} \bar{x}) \\ &= (\kappa M + \kappa \cdot c^\mathsf{T} \bar{x}) \cdot (\kappa M + \kappa \cdot c^\mathsf{T} (\mathbf{1} - \bar{x})) \\ &= \kappa^2 \cdot \left(M^2 + M \cdot c^\mathsf{T} \bar{x} + M \cdot c^\mathsf{T} (\mathbf{1} - \bar{x}) + (c^\mathsf{T} \bar{x}) \big(c^\mathsf{T} (\mathbf{1} - \bar{x}) \big) \right) \\ &= \kappa^2 \cdot \left(M^2 + MC + (c^\mathsf{T} \bar{x}) \big(c^\mathsf{T} (\mathbf{1} - \bar{x}) \big) \right) \end{aligned}$$

Therefore, $f(\bar{x}) \geq \theta$ reduces to $\left(c^\mathsf{T}\bar{x}\right)\left(c^\mathsf{T}(\mathbf{1}-\bar{x})\right) \geq (C/2)^2$. This inequality holds if and only if $c^\mathsf{T}\bar{x} = c^\mathsf{T}(\mathbf{1}-\bar{x}) = C/2$. Thus (a, b, κ, θ) is a yes-instance for RKP if and only if c is a yes-instance for CCP (both witnessed by the identical solution vector $\bar{x}$). □

CCP remains **NP**-hard even if all components of c are positive (in contrast to only being non-negative) and distinct (Garey and Johnson 1979). By the above construction we naturally obtain:

Observation 7 RKP *remains* ***NP****-hard even if all components of* a *and all components of* b *are positive and distinct.*

4 An FPTAS via MOKP

The MOKP admits an FPTAS proposed by Erlebach et al. (2002). To apply this FPTAS, we first establish a connection between RKP and MOKP. To this end, we define for a given RKP instance with $a, b \in \mathbb{N}^n$ and $\kappa \in \mathbb{N}$ a CBKP instance and thus a MOKP instance of the following form:

$$\begin{aligned} &\max\ a^\mathsf{T}x \\ &\max\ b^\mathsf{T}x \\ \text{s.t. } &\mathbf{1}^\mathsf{T}x \leq \kappa \\ &x \in \{0,1\}^n. \end{aligned}$$

Lemma 8 *Any optimal solution to* RKP *is a Pareto-optimal solution to the respective* MOKP *instance.*

Proof Let $\hat{x}$ be an optimal solution to RKP and assume its value vector is dominated in the MOKP instance by some other value vector. Let x be a solution corresponding to the latter. Clearly, $f(x) = \left(a^\mathsf{T}x\right)\left(b^\mathsf{T}x\right) > \left(a^\mathsf{T}\hat{x}\right)\left(b^\mathsf{T}\hat{x}\right) = f(\hat{x})$. As both solutions are feasible w.r.t. the cardinality constraint, $\hat{x}$ cannot be an optimal solution to RKP. □

Algorithm 1 Given an RKP instance (a, b, κ) and $\varepsilon > 0$, we solve it as an MOKP instance using the FPTAS by Erlebach et al. (2002). The latter algorithm is thereby started with a quality requirement $\varepsilon' := \varepsilon/2$ to compute an approximate solution set S. Our RKP solution is an $x \in S$ with maximum $\left(a^\mathsf{T}x\right)\left(b^\mathsf{T}x\right)$. ◁

Theorem 9 *Algorithm 1 is an FPTAS for* RKP *with running time bounded by* $\mathcal{O}(\varepsilon^{-2}n^3\log(n\alpha)\log(n\beta)) \subseteq \mathcal{O}(\varepsilon^{-2}n^3\log(nW^\uparrow)^2) \subseteq \mathcal{O}(\varepsilon^{-2}N^5)$.

Proof We first prove the approximation ratio. Let $\hat{x}$ be an optimal solution to RKP. We also know that $(a, b)^\mathsf{T}\hat{x}$ is a non-dominated point to the MOKP instance by Lemma 8. Thus, the FPTAS computes at least one solution $\mathring{x}$ with $a^\mathsf{T}\mathring{x} \geq (1-\varepsilon')a^\mathsf{T}\hat{x}$ and $b^\mathsf{T}\mathring{x} \geq (1-\varepsilon')b^\mathsf{T}\hat{x}$. The chosen solution $\bar{x} \in S$ has maximum $\left(a^\mathsf{T}\bar{x}\right)\left(b^\mathsf{T}\bar{x}\right)$ and thus we have

$$f(\bar{x}) = \left(a^\mathsf{T}\bar{x}\right)\left(b^\mathsf{T}\bar{x}\right) \geq \left(a^\mathsf{T}\mathring{x}\right)\left(b^\mathsf{T}\mathring{x}\right)$$
$$\geq (1-\varepsilon')^2\left(a^\mathsf{T}\hat{x}\right)\left(b^\mathsf{T}\hat{x}\right) \geq (1-\varepsilon)\left(a^\mathsf{T}\hat{x}\right)\left(b^\mathsf{T}\hat{x}\right) = (1-\varepsilon)f(\hat{x}).$$

The running time of the FPTAS by Erlebach et al. (2002) is in $\Theta\Big(\varepsilon^{-d}n^{d+1}(\log U_1)\cdots(\log U_d)\Big)$, where U_i is an upper bound on the respective objective function values. Since $\varepsilon/2 \in \Theta(\varepsilon)$, we can bound the running time in the above algorithm by $\mathcal{O}\Big(\varepsilon^{-2}n^3(\log n\alpha)(\log n\beta)\Big)$. □

Corollary 10 *Algorithm 1 is an FPTAS for* CBKP *with running time bounded by* $\mathcal{O}(\varepsilon^{-2}n^3\log(n\alpha)\log(n\beta)) \subseteq \mathcal{O}(\varepsilon^{-2}n^3\log(nW^{\uparrow})^2) \subseteq \mathcal{O}(\varepsilon^{-2}N^5)$.

While these results, based on using the FPTAS from the literature as a black box, already improve on the known 4.5-approximation for RKP, we show below that we can further improve the results in terms of running time by attacking the problem more directly.

5 Exact pseudopolynomial algorithm

We describe an exact pseudopolynomial algorithm for RKP. It is also based on a multi-objective optimization decomposition. Given a solution $x \in \{0,1\}^n$, we map it to its *evaluation tuple* via

$$x \mapsto \left\langle a^\mathsf{T}x, b^\mathsf{T}x, \mathbf{1}^\mathsf{T}x\right\rangle.$$

These tuples form the corner stone of our enumeration procedure below. Observe that multiple solutions may attain the same evaluation tuple. A tuple $t = \langle t_1, t_2, t_3\rangle$ *dominates* a tuple $t' = \langle t'_1, t'_2, t'_3\rangle$ if $t_1 \geq t'_1$, $t_2 \geq t'_2$, $t_3 \leq t'_3$, and $t \neq t'$. Recall that we can assume w.l.o.g. that $\beta \leq \alpha$.

Algorithm 2 We maintain a set of tuples T. It is initialized with $T = \{\langle 0,0,0\rangle\}$, the corresponding solution would be the n-dimensional all-zeros vector. The algorithm now runs in n iterations. In iteration i, we consider every current tuple $t = \langle t_1, t_2, t_3\rangle \in T$ and obtain a new tuple $t' = \langle t'_1, t'_2, t'_3\rangle := \langle t_1 + a_i, t_2 + b_i, t_3 + 1\rangle$. In solution space, this corresponds to setting the i-th component of the associated solution to 1. We discard t' if $t'_3 > \kappa$, or there is another tuple $\langle s, t'_2, t'_3\rangle \in T$ with $s \geq t'_1$. We denote these pruning strategies by P1 and P2, respectively.

After all n iterations, we attain the optimal objective value as $\max_{\langle t_1,t_2,t_3\rangle\in T}\{t_1 \cdot t_2\}$. ◁

If we are also interested in optimal solutions, we can use standard techniques to keep track of one solution per tuple. This incurs an additional factor of $\Theta(n)$ in the running time later.

We observe that the computed solution is feasible. Moreover, the final set T contains every possible non-dominated tuple—and possibly some more:

Lemma 11 *Algorithm 2 finds all non-dominated evaluation tuples of a given instance.*

Proof The algorithm is essentially a brute-force enumeration, computing all possible evaluation tuples. However, the algorithm uses two pruning strategies.

Pruning P1 is correct since the algorithm never decreases the number of non-zero components in any considered solution. Pruning P2 only deletes (some) dominated evaluation tuples: The tuples in iteration i correspond to subsolutions only considering the first i components. It is well-known, see e.g. (Nemhauser and Ullmann 1969), that any non-dominated solution x cannot contain any dominated subsolutions. If it contained a dominated subsolution x', let y' denote the subsolution dominating x'; we could substitute x' by y' in x to achieve a new solution that then dominates x. □

We now establish how the non-dominated tuples given by Algorithm 2 help us in solving RKP. We have to be a bit more careful than in the proof of Lemma 8.

Lemma 12 *The evaluation tuple of any optimal solution to* RKP *is non-dominated.*

Proof Let $\hat{x}$ be an optimal solution to RKP and assume its evaluation tuple is dominated by some other evaluation tuple. Let x be a solution corresponding to the latter. Then $a^\mathsf{T}x \geq a^\mathsf{T}\hat{x}$, $b^\mathsf{T}x \geq b^\mathsf{T}\hat{x}$, and $\mathbf{1}^\mathsf{T}x \leq \mathbf{1}^\mathsf{T}\hat{x}$, with at least one of these inequalities being strict. If one of the first two is strict, x attains a better objective value than $\hat{x}$—a contradiction. If only the thirds is strict, we have $\mathbf{1}^\mathsf{T}x < \kappa$ and could set a further component in x to 1, obtaining a yet better objective value—again a contradiction. □

Theorem 13 *Algorithm 2 is a pseudopolynomial exact algorithm for* RKP. *Its running time is bounded by* $\mathcal{O}(n^3\beta) = \mathcal{O}(n^3 W^{\downarrow})$.

Proof The fact that Algorithm 2 computes all non-dominated evaluation tuples together with Lemma 12 establishes correctness, and it remains to discuss the running time. We can encode T as a two-dimensional array A with $\kappa\beta$ rows and κ columns. A tuple $\langle t_1, t_2, t_3\rangle$ is stored as $A[t_2, t_3] = t_1$; we have $A[t_2', t_3'] = -\infty$ if there is no tuple $\langle \cdot, t_2', t_3'\rangle \in T$. Thus each pruning test can be trivially performed in constant time. For later reference, let τ denote the maximum number of tuples ever in T. We have $\tau \leq \kappa^2\beta \leq n^2\beta$. Over the n iterations this yields a running time of $\mathcal{O}(n\tau) \subseteq \mathcal{O}(n^3\beta)$. □

Corollary 14 *Algorithm 2 yields a pseudopolynomial exact algorithm for* CBKP. *Its running time is bounded by* $\mathcal{O}(n^3\beta) = \mathcal{O}(n^3 W^{\downarrow})$.

6 FPTAS

Algorithm 3 For a given $\varepsilon \in (0, 1)$, let

$$\delta := \sqrt[n]{1/(1-\varepsilon)} > 1$$

and for $y \in \mathbb{N}$ let

$$\Delta(y) := \begin{cases} \lceil \log_\delta y \rceil & \text{if } y \geq 1 \\ -1 & \text{else.} \end{cases}$$

Given a solution $x \in \{0, 1\}^n$, we map it to its *scaled* evaluation tuple via

$$x \mapsto \left\langle a^\mathsf{T} x, \Delta\left(b^\mathsf{T} x\right), \mathbf{1}^\mathsf{T} x \right\rangle.$$

We reuse Algorithm 2 but modify it slightly: Instead of working on evaluation tuples, we let Algorithm 2 now work on scaled tuples. Observe that we thus initialize T with the tuple $\langle 0, -1, 0\rangle$ that corresponds to $x = \mathbf{0}$.

The only algorithmic part we need to change is the computation of new scaled evaluation tuples from a predecessor. In Algorithm 2, we were able to deduce a new candidate tuple t' using only the predecessor tuple $t \in T$ and not an actual solution x, since only linear functions were involved. To also achieve the same running time in the new algorithm, we also shall not store a full solution for each tuple. Instead, for each scaled tuple $t \in T$, we additionally store a single value $B(t)$: Let x be a solution that yields tuple t; we want $B(t) := b^\mathsf{T} x$ to be the value we would store as the second entry in an unscaled evaluation tuple of x. The initial tuple has $B(\langle 0, -1, 0\rangle) = 0$. For a scaled evaluation tuple $t = \langle t_1, t_2, t_3\rangle \in T$ in iteration i, we can then efficiently generate a new tuple $t' := \langle t_1 + a_i, \Delta(B(t) + b_i), t_3 + 1\rangle$ with $B(t') := B(t) + b_i$. Tuple t' is added to T subject to the same pruning strategies as described in Algorithm 2, working purely on the scaled evaluation tuples.

The final objective value is naturally computed as $\max_{t=\langle t_1,t_2,t_3\rangle\in T} t_1 \cdot B(t)$. ◁

Lemma 15 *The running time of Algorithm 3 is bounded by* $\mathcal{O}(\varepsilon^{-1} n^3 \log(n\beta)) \subseteq \mathcal{O}(\varepsilon^{-1} n^3 \log(nW^{\downarrow})) \subseteq \mathcal{O}(\varepsilon^{-1} N^4)$.

Proof The second entry in any scaled evaluation tuple is comprised by the Δ function that can attain at most $\lceil \log_\delta \kappa\beta \rceil + 2$ different values. Thus, we can let our array A have $\lceil \log_\delta \kappa\beta \rceil + 2$ rows and κ columns. Again, let τ denote the size of A. We have:

$$\tau \leq \kappa(2 + \lceil \log_\delta \kappa\beta \rceil) \leq n(3 + \log_\delta n\beta) \in \mathcal{O}\left(\frac{n \log n\beta}{\log \delta}\right) = \mathcal{O}\left(\frac{n^2 \log n\beta}{\log {}^1\!/_{(1-\varepsilon)}}\right).$$

The $B(t)$ values can be perceived as an additional entry in every cell of A and thus contributing as a constant factor to the size of A. Considering ${}^1\!/_\varepsilon \to \infty$, we have

$$\frac{1}{\log {}^1\!/_{(1-\varepsilon)}} = \frac{-1}{\log(1-\varepsilon)} = \frac{1}{\varepsilon + {}^{\varepsilon^2}\!/_2 + {}^{\varepsilon^3}\!/_3 + \ldots} \in \Theta\left(\frac{1}{\varepsilon}\right)$$

by Taylor expansion. Thus, $\tau \in \mathcal{O}(\varepsilon^{-1} n^2 \log n\beta)$. The algorithm's overall running time is thus understood to be bounded by $\mathcal{O}(n\tau) = \mathcal{O}(\varepsilon^{-1} n^3 \log n\beta)$. □

Theorem 16 *Algorithm 3 is an FPTAS for* RKP.

Proof Consider an RKP instance (a, b, κ) with an optimal solution $\hat{x}$. Lemma 15 settles the running time requirement. It remains to show that for every $\varepsilon > 0$, Algorithm 3 finds a feasible solution $\mathring{x}$ with $f(\mathring{x}) \geq (1 - \varepsilon) f(\hat{x})$.

Let X_i be a set of solutions corresponding to T after iteration $i \in \{1, \ldots n\}$, and $X_0 = \{\mathbf{0}\}$. Recall that in any solution $x \in X_i$, only the first i components of x may be

non-zero. At any point in the algorithm, we may consider a final best solution $\check{x}$ that we could still hope to find; initially set $\check{x}{:=}\hat{x}$.

If the algorithm finds an optimal solution, the claim is true. Suppose an optimal solution is not found; let ℓ be the smallest iteration index, such that there is no solution in X_ℓ that has the same first ℓ components as $\check{x}$. We define $\check{x}^\ell \in \{0,1\}^n$ such that $\check{x}^\ell_i{:=}\check{x}_i$ for $1 \leq i \leq \ell$ and $\check{x}^\ell_i{:=}0$ for $\ell < i \leq n$. That is, $\check{x}^\ell$ is equal to $\check{x}$ on the first ℓ components and 0 in all further components. Since $\check{x}^\ell \notin X_\ell$, there must be some solution $\bar{x}^\ell \in X_\ell$ that dominates $\check{x}^\ell$.

Since $\bar{x}^\ell$ dominates $\check{x}^\ell$ we have

$$\begin{aligned} a^\mathsf{T}\bar{x}^\ell &\geq a^\mathsf{T}\check{x}^\ell, \\ \Delta\left(b^\mathsf{T}\bar{x}^\ell\right) &\geq \Delta\left(b^\mathsf{T}\check{x}^\ell\right), \text{ and} \\ 1^T\bar{x}^\ell &\leq 1^T\check{x}^\ell. \end{aligned}$$

Focusing on the second inequality, if $\Delta\left(b^\mathsf{T}\bar{x}^\ell\right) = -1$, we have $b^\mathsf{T}\bar{x}^\ell = b^\mathsf{T}\check{x}^\ell = 0$. Otherwise, if $\Delta\left(b^\mathsf{T}\check{x}^\ell\right) = -1$, clearly $b^\mathsf{T}\bar{x}^\ell \geq b^\mathsf{T}\check{x}^\ell$. If neither Δ-evaluation yields -1, we have for $\nu{:=}\log_\delta b^\mathsf{T}\bar{x}^\ell$ and $\mu{:=}\log_\delta b^\mathsf{T}\check{x}^\ell$, that $\nu \geq \mu - 1$ and thus

$$b^\mathsf{T}\bar{x}^\ell = \delta^\nu \geq \delta^{\mu-1} = 1/\delta \cdot b^\mathsf{T}\check{x}^\ell.$$

Consequently, in all cases the inequality

$$b^\mathsf{T}\bar{x}^\ell \geq 1/\delta \cdot b^\mathsf{T}\check{x}^\ell$$

holds.

We now define $\check{x}^r{:=}\check{x} - \check{x}^\ell$, i.e., $\check{x}^r$ matches $\check{x}$ on the last $n-\ell$ components and is 0 on the first ℓ components. Let $\tilde{x}{:=}\bar{x}^\ell + \check{x}^r$. We see that

$$\begin{aligned} a^\mathsf{T}\tilde{x} = a^\mathsf{T}\bar{x}^\ell + a^\mathsf{T}\check{x}^r &\geq a^\mathsf{T}\check{x}^\ell + a^\mathsf{T}\check{x}^r = a^\mathsf{T}\check{x}, \\ b^\mathsf{T}\tilde{x} = b^\mathsf{T}\bar{x}^\ell + b^\mathsf{T}\check{x}^r &\geq 1/\delta \cdot b^\mathsf{T}\check{x}^\ell + b^\mathsf{T}\check{x}_i \geq 1/\delta \cdot b^\mathsf{T}\check{x}, \text{ and} \\ 1^\mathsf{T}\tilde{x} = 1^\mathsf{T}\bar{x}^\ell + 1^\mathsf{T}\check{x}^r &\leq 1^\mathsf{T}\check{x}^\ell + 1^\mathsf{T}\check{x}^r = 1^\mathsf{T}\check{x}. \end{aligned}$$

Intuitively, while $\check{x}$ is no longer attainable after iteration ℓ, solution $\tilde{x}$ is still attainable as it can arise from $\bar{x}$. At the same time, $\tilde{x}$ is a solution with an objective value that is at most a factor of $1/\delta$ worse than $\check{x}$ in the second component.

We can iterate the above consideration with $\tilde{x}$ assuming the role of $\check{x}$. We again look for an iteration index ℓ, such that the new $\check{x}$ does not agree with some solution in X_ℓ on its first ℓ components. If such an index is not present then the new $\check{x}$ is actually computed by the algorithm. If such an index is present, this new index ℓ is now strictly larger than the index considered before. As there are only n possible indices, we repeat this argument at most n times. For each repetition, we lose a factor of at most $1/\delta$ in the second component.

Let $\mathring{x}$ be the final solution by the algorithm. We conclude that $b^\mathsf{T}\mathring{x} \geq {}^1/_{\delta^n} \cdot b^\mathsf{T}\hat{x} = (1-\varepsilon)b^\mathsf{T}\hat{x}$, while $a^\mathsf{T}\mathring{x} \geq a^\mathsf{T}\hat{x}$ and $1^\mathsf{T}\mathring{x} \leq 1^\mathsf{T}\hat{x} \leq \kappa$. Consequently, $\mathring{x}$ is feasible and $f(\mathring{x}) = \left(a^\mathsf{T}\mathring{x}\right)\left(b^\mathsf{T}\mathring{x}\right) \geq \left(a^\mathsf{T}\hat{x}\right)(1-\varepsilon)\left(b^\mathsf{T}\hat{x}\right) = (1-\varepsilon)f(\hat{x})$. □

Corollary 17 *Algorithm 3 yields an FPTAS for* CBKP *with running time in* $\mathcal{O}(\varepsilon^{-1} n^3 \log n\beta) \subseteq \mathcal{O}(\varepsilon^{-1} n^3 \log(nW^\downarrow)) \subseteq \mathcal{O}(\varepsilon^{-1} N^4)$.

7 Conclusion

We answered all open questions from Schulze et al. (2020) regarding the complexity and approximability of RKP: while the problem is indeed **NP**-hard, it allows not only a pseudopolynomial exact algorithm, but also an FPTAS—the theoretically strongest kind of approximation algorithm. Furthermore, our techniques in fact also allow us to directly tackle the CBKP, achieving the equivalent algorithmic results.

Comparing Algorithms 1 and 3, our approach achieves a better running time in all cases: $\mathcal{O}(\varepsilon^{-2} n^3 \log(nW^\downarrow) \log(nW^\uparrow)) \subseteq \mathcal{O}(\varepsilon^{-2} N^5)$ vs. $\mathcal{O}(\varepsilon^{-1} n^3 \log\left(nW^\downarrow\right)) \subseteq \mathcal{O}(\varepsilon^{-1} N^4)$. Especially, our pseudopolynomial algorithm shows that RKP can be solved in polynomial time even if only $W^\downarrow$ (but not $W^\uparrow$) is bounded by a polynomial in the input size.

Furthermore, it should be understood that our algorithms (and proofs) can trivially be extended to any fixed arbitrary number of objective functions.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Erlebach T, Kellerer H, Pferschy U (2002) Approximating multiobjective knapsack problems. Manag Sci 48(12):1603–1612

Gallo G, Hammer P, Simeone B (1980) Quadratic knapsack problems. In: Padberg M (ed) Combinatorial optimization, mathematical programming studies, vol 12. Springer, Berlin

Garey MR, Johnson DS (1979) Computers and intractability; a guide to the theory of NP-completeness. Series of books in the mathematical sciences. W. H. Freeman & Co, New York

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

Nemhauser GL, Ullmann Z (1969) Discrete dynamic programming and capital allocation. Manag Sci 15(9):494–505

Papadimitriou CH, Yannakakis M (2000) On the approximability of trade-offs and optimal access of web sources. In: FOCS. IEEE Computer Society, pp 86–92

Paquete L, Schulze B, Stiglmayr M et al (2022) Computing representations with hypervolume scalarizations. Comput Oper Res 137:105349

Pferschy U, Schauer J (2016) Approximation of the quadratic knapsack problem. INFORMS J Comput 28(2):308–318
Pisinger D (2007) The quadratic knapsack problem–a survey. Discrete Appl Math 155(5):623–648
Rader DJJ, Woeginger GJ (2002) The quadratic $0-1$ knapsack problem with series-parallel support. Oper Res Lett 30(3):159–166
Schulze B, Stiglmayr M, Paquete L et al (2020) On the rectangular knapsack problem: approximation of a specific quadratic knapsack problem. Math Methods Oper Res 92(1):107–132
Taylor R (2016) Approximation of the quadratic knapsack problem. Oper Res Lett 44(4):495–497
Xu Z (2012) A strongly polynomial FPTAS for the symmetric quadratic knapsack problem. Eur J Oper Res 218(2):377–381