Optimization Letters (2021) 15:2529–2540
https://doi.org/10.1007/s11590-021-01760-x

ORIGINAL PAPER

Approximating the product knapsack problem

Ulrich Pferschy[1] · Joachim Schauer[2] · Clemens Thielen[3]

Received: 9 April 2020 / Accepted: 27 May 2021 / Published online: 4 June 2021

Abstract
We consider the product knapsack problem, which is the variant of the classical 0-1 knapsack problem where the objective consists of maximizing the product of the profits of the selected items. These profits are allowed to be positive or negative. We present the first fully polynomial-time approximation scheme for the product knapsack problem, which is known to be weakly NP-hard. Moreover, we analyze the approximation quality achieved by a natural extension of the classical knapsack greedy procedure to the product knapsack problem.

Keywords Knapsack problem · Approximation scheme · Greedy procedure

1 Introduction

The 0-1 knapsack problem (KP) is a well-studied combinatorial optimization problem that has been treated extensively in the literature, with two monographs [3,4] devoted to KP and its relatives. Given a positive knapsack capacity C and n items $j = 1, \ldots, n$ with positive weights w_j and profits p_j, the task in the classical 0-1 knapsack problem is to select a subset of items with maximum total profit subject to the constraint that the total weight of the selected items may not exceed the knapsack capacity. The 0-1 knapsack problem is (weakly) NP-hard, but it admits a fully polynomial-time

✉ Clemens Thielen
clemens.thielen@tum.de

Ulrich Pferschy
pferschy@uni-graz.at

Joachim Schauer
joachim.schauer@fh-joanneum.at

[1] Department of Statistics and Operations Research, University of Graz, Universitaetsstrasse 15, A-8010 Graz, Austria

[2] Institute of Internet Technologies & Applications, FH JOANNEUM, Werk-VI-Strasse 46, A-8605 Kapfenberg, Austria

[3] TUM Campus Straubing, Technical University of Munich, Essigberg 3, D-94315 Straubing, Germany

approximation scheme (FPTAS) and can be solved exactly in pseudopolynomial time by dynamic programming (cf. [3]).

The *product knapsack problem (PKP)* is a new addition to the knapsack family. It has recently been introduced in [1] and is formally defined as follows:

Definition 1 (Product knapsack problem (PKP))

INSTANCE: Items $j \in N := \{1, \ldots, n\}$ with weights $w_j \in \mathbb{Z}$ and profits $p_j \in \mathbb{Z}$, and a positive knapsack capacity $C \in \mathbb{N}_+$.

TASK: Find a subset $S \subseteq N$ with $\sum_{j \in S} w_j \leq C$ such that $\prod_{j \in S} p_j$ is maximized.

The solution $S = \emptyset$ is always feasible and is assumed to yield an objective value of zero. Note that the assumption that the knapsack capacity as well as all weights and profits are integers is without loss of generality. Indeed, any instance with rational input data can be transformed into an equivalent instance with integer input data in polynomial time by multiplying all numbers by their lowest common denominator.

D'Ambrosio et al. [1] list several application scenarios for PKP, in particular in the area of computational social choice, and also provide pointers to literature on other nonlinear knapsack problems. Furthermore, two different ILP formulations for PKP are presented and compared from both a theoretical and a computational perspective. In addition, D'Ambrosio et al. [1] develop an algorithm performing dynamic programming by weights with pseudopolynomial running time $\mathcal{O}(nC)$. A computational study exhibits the strengths and weaknesses of the dynamic program and the ILP approaches for determining exact solutions of PKP depending on the characteristics of the test instances.

Concerning the complexity of PKP, a short proof of weak NP-hardness is given as a side remark in [1]. This proof, however, uses a reduction from KP and requires an exponential blow-up of the profits of the given instance of KP (by putting them into the exponent of 2). Since KP is only weakly NP-hard, this does not prove the desired hardness result. However, a valid NP-hardness proof for PKP has recently been provided in [2], which shows that the problem is weakly NP-hard even when all profits are required to be positive.

Note that this proof requires concepts of advanced calculus. As a possibly useful alternative, we provide a simpler proof using only elementary operations in an extended version of this paper, which is available as a technical report [5].

1.1 Our contribution

In this paper, we provide an FPTAS for PKP based on dynamic programming by profits. Since PKP is weakly NP-hard, an FPTAS is the best approximation result possible for the problem unless P = NP. Moreover, the construction of an FPTAS deserves attention since standard greedy-type algorithms do not yield a constant approximation ratio for PKP. We demonstrate this in Sect. 4 by providing a tight analysis of the greedy algorithm obtained by extending the classical greedy procedure for KP to PKP in the natural way.

We do not report computational experiments on the FPTAS or the greedy algorithm. The presented pseudocode descriptions are provided in order to illustrate the algorithms and allow a rigorous analysis of the obtained approximation ratios, but are not optimized for practical efficiency.

2 Preliminaries

In contrast to KP, both the item weights w_j and the item profits p_j are allowed to be negative in PKP. However, one can exclude certain weight-profit combinations that yield "useless" items, which leads to the following assumption used throughout the paper:

Assumption 1 *Any instance of PKP satisfies:*

(a) Any single item fits into the knapsack, i.e., $w_j \leq C$ *for all* $j \in N$.
(b) All profits are nonzero, i.e., $p_j \in \mathbb{Z} \setminus \{0\}$ *for all* $j \in N$.
(c) For each item $j \in N$ *with negative profit* $p_j < 0$*, there exists another item* $j' \in N \setminus \{j\}$ *with negative profit* $p_{j'} < 0$ *such that* $w_j + w_{j'} \leq C$.
(d) All weights are nonnegative, i.e., $w_j \in \mathbb{N}_0$ *for all* $j \in N$.
(e) All items with weight zero have negative profit, i.e., $p_j < 0$ *if* $w_j = 0$.

We note that Assumption 1 imposes no loss of generality and can easily be checked in polynomial time. Indeed, items $j \in N$ violating (a), (b), or (c) can never be part of any feasible solution with positive objective value and may, thus, be removed from the instance. The nonnegativity of the weights w_j demanded in (d) has been shown to impose no loss of generality in [1]. For (e), we note that items j with $w_j = 0$ can always be assumed to be packed if their profit is positive (but items j with $w_j = 0$ and negative profit remain part of the optimization).

Using Assumption 1 (b), the item set N can be partitioned into $N^+ := \{j \in N \mid p_j \geq 1\}$ and $N^- := \{j \in N \mid p_j \leq -1\}$. For convenience, we define $p_{\max} := \max_{j \in N} |p_j|$, $p^+_{\max} := \max_{j \in N^+} p_j$, and $p^-_{\max} := \max_{j \in N^-} |p_j|$.

Throughout the paper, we denote an optimal solution set for a given instance of PKP by S^* and the optimal objective value by z^*. Note that we must always have $z^* \geq 1$ since packing any item from N^+ or any feasible pair of items from N^- yields an objective value of at least 1.

Definition 2 For $0 < \alpha \leq 1$, an algorithm A that computes a feasible solution set $S \subseteq N$ with $\prod_{j \in S} p_j \geq \alpha \cdot z^*$ in polynomial time for every instance of PKP is called an *α-approximation algorithm* for PKP. The value α is then called the *approximation ratio* of A.

A *polynomial-time approximation scheme (PTAS)* for PKP is a family of algorithms $(A_\varepsilon)_{\varepsilon > 0}$ such that, for each $\varepsilon > 0$, the algorithm A_ε is a $(1-\varepsilon)$-approximation algorithm for PKP. A PTAS $(A_\varepsilon)_{\varepsilon > 0}$ for PKP is called a *fully polynomial-time approximation scheme (FPTAS)* if the running time of A_ε is additionally polynomial in $\frac{1}{\varepsilon}$.

Throughout the paper, $\log(x)$ always refers to the base 2 logarithm of x and $\ln(x)$ refers to the natural logarithm of x.

3 A fully polynomial-time approximation scheme

We now derive a fully polynomial-time approximation scheme (FPTAS) for PKP based on dynamic programming.

The most common approach for the exact solution of knapsack-type problems in pseudo-polynomial time applies dynamic programming by weights. This means that, for every capacity value $d = 0, 1, \ldots, C$, the largest profit value reachable by a feasible solution is determined, which yields a running time polynomial in C (see [3, Sec. 2.3]). However, for obtaining fully polynomial-time approximation schemes, one usually performs dynamic programming by profits. In this case, for every profit value p up to some upper bound U on the objective function value, the smallest weight required for a feasible solution with profit p is sought, which leads to a running time polynomial in U (see [3, Lemma 2.3.2]). Then, the profit space is simplified in some way, e.g., by scaling (cf. [3, Sec. 2.6]), such that the running time of the dynamic program becomes polynomial and the incurred loss of accuracy remains bounded. D'Ambrosio et al. [1] provide an algorithm solving PKP with dynamic programming by weights, where each entry of the dynamic programming array contains the objective value of a subproblem. However, exchanging the roles of profits and weights (as it is done, e.g., for KP, see [3, Sec. 2.3]), would require a dynamic programming array of length $\mathcal{O}(p_{\max}^n)$, which is exponential and does not permit a suitable scaling procedure.

An obvious way out of this dilemma would be the application of the logarithm to the profits. In fact, such an approach is suggested as a side remark in [1, Sec. 3] for dynamic programming by weights (without commenting on the details of the rounding process). For dynamic programming by profits, however, the profit values must be mapped to integers as indices of the dynamic programming array and there seems to be no way to preserve optimality in such a process. It should also be noted that applying any k-approximation algorithm for KP to the instance resulting from logarithmization would only yield a $(1/z^*)^{1/k}$-approximation for PKP. Thus, constant-factor approximations for PKP require different approaches.

We now construct a scaled profit space that actually yields a $(1-\varepsilon)$–approximation for PKP. Our scaling construction is based on a parameter $K > 0$ depending on ε, which will be defined later. For every item j, we define an integer scaled profit value in the logarithmized space as

$$\tilde{p}_j := \left\lfloor \frac{\log(|p_j|)}{K} \right\rfloor. \tag{1}$$

Since $|p_j| \geq 1$, we have $\tilde{p}_j \geq 0$, and we obtain $\tilde{p}_j = 0$ if and only if $|p_j| = 1$. Note that an item j with $p_j = -1$ and $\tilde{p}_j = 0$ might still be useful for changing the sign of the solution of PKP. Analogous to $p_{\max}$, we define $\tilde{p}_{\max} := \left\lfloor \frac{\log(p_{\max})}{K} \right\rfloor$. Ruling out trivial instances, we can assume without loss of generality that $p_{\max} \geq 2$, so $\log(p_{\max}) \geq 1$.

Regarding the computability of the scaled profits $\tilde{p}_j$, we observe that their definition involves logarithms $\log(|p_j|)$, which cannot be computed exactly in polynomial time.

However, these logarithms only appear in expressions that are rounded to integers, so we do not have to compute these values exactly.

We define the following dynamic programming arrays for profit values $\tilde{p} = 0, 1, \ldots, n \cdot \tilde{p}_{\max}$:

$$W_j^+(\tilde{p}) := \min_{S \subseteq \{1,\ldots,j\}} \left\{ \sum_{i \in S} w_i \mid \sum_{i \in S} \tilde{p}_i = \tilde{p},\ |S \cap N^-| \text{ is even} \right\},$$

$$W_j^-(\tilde{p}) := \min_{S \subseteq \{1,\ldots,j\}} \left\{ \sum_{i \in S} w_i \mid \sum_{i \in S} \tilde{p}_i = \tilde{p},\ |S \cap N^-| \text{ is odd} \right\}.$$

Note that the empty set has even cardinality. For convenience, we set the minimum over the empty set equal to $+\infty$.

The computation of these arrays can be done by the following recursion, which is related to Algorithm $\mathrm{DP}_{\mathrm{PKP}}$ in [1, Fig. 1]:

If $p_j \geq 1$, then:

$$W_j^+(\tilde{p}) := \min\{W_{j-1}^+(\tilde{p}),\ W_{j-1}^+(\tilde{p} - \tilde{p}_j) + w_j\}$$
$$W_j^-(\tilde{p}) := \min\{W_{j-1}^-(\tilde{p}),\ W_{j-1}^-(\tilde{p} - \tilde{p}_j) + w_j\}$$

If $p_j \leq -1$, then:

$$W_j^+(\tilde{p}) := \min\{W_{j-1}^+(\tilde{p}),\ W_{j-1}^-(\tilde{p} - \tilde{p}_j) + w_j\}$$
$$W_j^-(\tilde{p}) := \min\{W_{j-1}^-(\tilde{p}),\ W_{j-1}^+(\tilde{p} - \tilde{p}_j) + w_j\}$$

The obvious initialization is given by $W_0^+(0) := 0$ and setting all other entries (including the hypothetical ones with $\tilde{p} < 0$) to $+\infty$.

The approximate solution set S^A is represented by the array entry with $\max\{\tilde{p} \mid W_n^+(\tilde{p}) \leq C\}$. It follows by construction that S^A maximizes the total profit in the associated instance of KP with scaled profits $\tilde{p}_j$ among all subsets of N that fulfill the weight restriction and contain an even number of items from N^-. In the following, we show that, by choosing

$$K := \frac{\varepsilon}{n^2} > 0\,, \tag{2}$$

the set S^A yields a $(1-\varepsilon)$-approximation for PKP and can be computed in polynomial time via the above dynamic programming procedure. To this end, we use the following two lemmas:

Lemma 1 *For $\varepsilon \in (0, 1)$, we have $\varepsilon \leq -\log(1-\varepsilon)$.*

Proof The statement follows since, for any $x \in (0, 1)$, we have

$$-\log(1-x) = -\ln(1-x)/\ln(2) \geq -\ln(1-x) = \sum_{k=1}^{\infty} \frac{x^k}{k} \geq x.$$

□

Lemma 2 *Any optimal solution set S^* for PKP satisfies*

$$\sum_{j \in S^*} \log(|p_j|) \geq \log(p_{\max}).$$

Proof Let $j_{\max} \in N$ denote an item with $|p_{j_{\max}}| = p_{\max}$. If $p_{j_{\max}} > 0$, then the set $\{j_{\max}\}$, which is feasible for PKP by Assumption 1 (a), has objective value $p_{\max}$. If $p_{j_{\max}} < 0$, Assumption 1 (c) implies that there exists another item $j' \neq j_{\max}$ with $p_{j'} < 0$ such that $\{j_{\max}, j'\}$ is feasible for PKP, and this set has objective value $p_{j'} \cdot p_{j_{\max}} \geq p_{\max}$ since $p_{j'} \leq -1$ by Assumption 1 (b). Thus, in both cases, the optimality of S^* for PKP implies that

$$\prod_{j \in S^*} p_j \geq p_{\max} \Leftrightarrow \log\left(\prod_{j \in S^*} p_j\right) \geq \log(p_{\max})$$

$$\Leftrightarrow \log\left(\prod_{j \in S^*} |p_j|\right) \geq \log(p_{\max}) \Leftrightarrow \sum_{j \in S^*} \log(|p_j|) \geq \log(p_{\max}).$$

□

Proposition 1 *The running time for computing S^A is in $\mathcal{O}(\frac{n^4}{\varepsilon} \log(p_{\max}))$, which is polynomial in $1/\varepsilon$ and encoding length of the input of PKP.*

Proof Clearly, for each of the n items, one has to pass through the whole length of the two dynamic programming arrays. Therefore, the total running time is in

$$\mathcal{O}(n^2\, \tilde{p}_{\max}) = \mathcal{O}\left(n^2 \frac{\log(p_{\max})}{K}\right) = \mathcal{O}\left(n^4 \frac{\log(p_{\max})}{\varepsilon}\right).$$

□

Proposition 2 *The set S^A yields a $(1-\varepsilon)$–approximation for PKP.*

Proof The proof consists of two parts. First, we analyze the effect of scaling by K and rounding down in (1) by showing that S^A yields an objective value close to the value of an optimal solution set S^* for PKP in the associated instance of KP with profits $\log(|p_j|)$. The argumentation closely follows the standard FPTAS for KP (see [3, Sec. 2.6]):

$$\sum_{j \in S^A} \log(|p_j|) \geq \sum_{j \in S^A} K \cdot \left\lfloor \frac{\log(|p_j|)}{K} \right\rfloor \geq \sum_{j \in S^*} K \cdot \left\lfloor \frac{\log(|p_j|)}{K} \right\rfloor \tag{3}$$

$$\geq \sum_{j \in S^*} K \cdot \left(\frac{\log(|p_j|)}{K} - 1\right) \geq \sum_{j \in S^*} \log(|p_j|) - n \cdot K \tag{4}$$

Table 1 Profits p_j, weights w_j, and scaled profits $\tilde{p}_j$ of the items in Example 1

item j	1	2	3	4	5
p_j	1	$2^{10}-1$	$-(2^{10}+1)$	2^{10}	-1
w_j	1	5	5	5	4
$\tilde{p}_j$	0	9998	10001	10000	0

To obtain the second inequality in (3), we exploited the optimality of S^A for the KP instance with profits $\tilde{p}_j$. We now set

$$\varepsilon' := \frac{-\log(1-\varepsilon)}{n \cdot \log(p_{\max})} > 0. \tag{5}$$

Then, using the definition of K in (2) and that $\varepsilon \leq -\log(1-\varepsilon)$ for $\varepsilon \in (0,1)$, we obtain

$$n \cdot K = \frac{\varepsilon}{n} \leq \frac{-\log(1-\varepsilon)}{n} = \varepsilon' \cdot \log(p_{\max}),$$

and using that $\sum_{j \in S^*} \log(|p_j|) \geq \log(p_{\max})$ by Lemma 2, the chain of inequalities in (3)–(4) yields that

$$\sum_{j \in S^A} \log(|p_j|) \geq \sum_{j \in S^*} \log(|p_j|) - \varepsilon' \cdot \log(p_{\max}) \geq (1-\varepsilon') \sum_{j \in S^*} \log(|p_j|).$$

In the second part of the proof, we simply raise two to the power of both sides of this inequality, i.e., $2^{\sum_{j \in S^A} \log(|p_j|)} \geq \left(2^{\left(\sum_{j \in S^*} \log(|p_j|)\right)}\right)^{1-\varepsilon'}$, so

$$\prod_{j \in S^A} |p_j| \geq \left(\prod_{j \in S^*} |p_j|\right)^{1-\varepsilon'} = z^* \cdot (1/z^*)^{\varepsilon'} \geq z^* \cdot \left(\frac{1}{(p_{\max})^n}\right)^{\varepsilon'} \tag{6}$$

$$= z^* \cdot 2^{-\varepsilon' n \log(p_{\max})} = z^* \cdot 2^{\log(1-\varepsilon)} = (1-\varepsilon) z^*. \tag{7}$$

Here, the right inequality in (6) is derived from the trivial bound $z^* \leq (p_{\max})^n$, and the second equality in (7) from the definition of ε' in (5). Recalling that S^A contains an even number of items from N^-, the claim follows. □

Propositions 1 and 2 immediately yield the following theorem:

Theorem 1 *There exists an FPTAS with running time in* $\mathcal{O}(\frac{n^4}{\varepsilon} \log(p_{\max}))$ *for PKP.* □

We conclude this section with an example illustrating how the FPTAS works.

Example 1 Consider the instance of PKP given by the $n = 5$ items with profits and weights as shown in Table 1 and a knapsack capacity of $C := 9$. We choose $\varepsilon = 0.025$ so that $K = \frac{\varepsilon}{n^2} = 0.001$.

The resulting scaled profits $\tilde{p}_j$ are shown in the last row of Table 1 and we have $\tilde{p}_{\max} = 10001$, so $n \cdot \tilde{p}_{\max} = 50005$. Thus, the FPTAS computes the relevant dynamic programming arrays $W_j^+(\tilde{p})$ and $W_j^-(\tilde{p})$ for all profit values $\tilde{p} = 0, 1, \ldots, 50005$. Note that $N^+ = \{1, 2, 5\}$ and $N^- = \{3, 5\}$.

For this instance, the FPTAS finds the optimal solution set $S^* = \{3, 5\}$ during the computation of $W_5^+(10001)$, which is given as follows:

$$W_5^+(10001) = \min\left\{W_4^+(10001),\ W_4^-(10001 - 0) + 4\right\} = \min\{+\infty, 5 + 4\} = 9$$

Here, $W_4^+(10001) = +\infty$ since a scaled profit of 10001 cannot be obtained by any subset $S \subseteq \{1, 2, 3, 4\}$ containing an even number of items from N^-, and $W_4^-(10001) = 5$ since a scaled profit of 10001 is reachable by the subset $S = \{3\} \subseteq \{1, 2, 3, 4\}$ that contains an odd number of items from N^-. Thus, the solution set corresponding to the array entry $W_5^+(10001)$ is $\{3, 5\} = S^*$, and since $\tilde{p} = 10001$ is indeed the highest value of $\tilde{p}$ for which $W_5^+(\tilde{p}) \leq C = 9$, this is also the set S^A returned by the FPTAS.

4 A greedy-type algorithm

For KP, the classical greedy procedure is probably one of the most obvious first attempts for anybody confronted with the problem. Hence, it is interesting to evaluate the performance of a variant of this greedy procedure also for PKP.

It is known that, for obtaining a bounded approximation ratio for KP in the classical greedy procedure, one has to take into account also the item with largest profit as a singleton solution (cf. [3, Sec. 2.1]). Extending this requirement to the negative profits allowed in PKP, we additionally determine, among all items with negative profits, a feasible pair of items with largest profit product. Moreover, if the greedy solution contains an odd number of items from N^-, we simply remove the negative-profit item whose profit has the smallest absolute value. This leads to the following natural greedy algorithm for PKP, which we refer to as PRODUCT GREEDY:

Algorithm 1 Algorithm PRODUCT GREEDY

1: Sort and renumber the items in nonincreasing order of $\frac{\log(|p_j|)}{w_j}$.
(Items j with $w_j = 0$ are put to the beginning of the ordering.)
2: Perform the classical greedy procedure with this ordering yielding solution set $\bar{S} \subseteq N$.
3: **if** $|\bar{S} \cap N^-|$ is odd **then**
4: $\quad j^- := \text{argmin}\{|p_j| \mid j \in \bar{S} \cap N^-\}$
5: $\quad S := \bar{S} \setminus \{j^-\}$
6: **else**
7: $\quad S := \bar{S}$
8: **end if**
9: Let $\{j_1, j_2\} \subseteq N^-$ be a pair of items with $w_{j_1} + w_{j_2} \leq C$ maximizing the profit product $p_{j_1} \cdot p_{j_2}$ over all such pairs.
10: Let $j_{\max}^+ := \text{argmax}\{p_j \mid j \in N^+\}$ be an item with largest positive profit.
11: **return** the best among the three solutions S, $\{j_1, j_2\}$, and $\{j_{\max}^+\}$.

We note that, since $\log(|p_j|)/w_j = \log\left(|p_j|^{1/w_j}\right)$ and the logarithm is a strictly increasing function, the sorting and renumbering of the items in step 1 of PRODUCT GREEDY can equivalently be done by sorting the items in nonincreasing order of $|p_j|^{1/w_j}$, which means that the values $\log(|p_j|)/w_j$ do not have to be computed in the algorithm.

Let $j^+_{\max} := \operatorname{argmax}\{p_j \mid j \in N^+\}$ denote an item with largest positive profit (i.e., with $p_{j^+_{\max}} = p^+_{\max}$) as in PRODUCT GREEDY. Similarly, we let $j^-_{\max} := \operatorname{argmax}\{|p_j| \mid j \in N^-\}$ denote an item with smallest negative profit (i.e., with $-p_{j^-_{\max}} = p^-_{\max}$). Then, by Assumption 1 (c), there exists another item in N^- that can be packed into the knapsack together with $j^-_{\max}$. This implies that the profits of the items j^- and j_1, j_2 considered in PRODUCT GREEDY satisfy

$$p_{j_1} \cdot p_{j_2} \geq -p_{j^-_{\max}} \geq -p_{j^-}. \tag{8}$$

In the following analysis, we denote the objective value obtained by PRODUCT GREEDY by z^H.

Theorem 2 *(a)* PRODUCT GREEDY *is a* $1/(z^*)^{2/3}$*-approximation algorithm for PKP.*
(b) PRODUCT GREEDY *is a* $1/(p_{\max})^2$*-approximation algorithm for PKP.*

Proof The algorithm clearly runs in polynomial time. In order to analyze its approximation ratio, let $s \in N$ be the *split item*, i.e., the first item in the given order that cannot be packed into the knapsack anymore during the greedy procedure performed in step 2. Similar to the analysis of the greedy procedure for KP, the analysis concentrates on the *split solution*, i.e., the set of items $\bar{S} = \{j \in N \mid j \leq s-1\}$ produced in step 2 of PRODUCT GREEDY.

We distinguish two cases depending on the number of items with negative profits in $\bar{S}$ and, for each of the two cases, two subcases depending on the sign of the profit p_s of the split item s:
Case 1: $|\bar{S} \cap N^-|$ is even.
In this case, the solution $S = \bar{S}$ is considered when choosing the best solution in step 11. Consider the sign of the split item's profit. If $p_s > 0$, then

$$\begin{aligned} 2 \cdot \log(z^H) &\geq 2 \cdot \max\left\{\sum_{j\in\bar{S}} \log(|p_j|),\ \log(p^+_{\max})\right\} \\ &\geq \sum_{j\in\bar{S}} \log(|p_j|) + \log(p^+_{\max}) \ \geq \ \sum_{j=1}^{s} \log(|p_j|). \end{aligned}$$

Obviously, we also have $\log(z^H) + \log(p^+_{\max}) \geq \sum_{j=1}^{s} \log(|p_j|)$.

Similarly, if $p_s < 0$, then

$$\begin{aligned} 2 \cdot \log(z^H) &\geq 2 \cdot \max \left\{ \sum_{j \in \bar{S}} \log(|p_j|),\ \log(|p_{j_1}|) + \log(|p_{j_2}|) \right\} \\ &\geq \sum_{j \in \bar{S}} \log(|p_j|) + \log(|p_{j_1}|) + \log(|p_{j_2}|) \\ &\geq \sum_{j \in \bar{S}} \log(|p_j|) + \log(|p_s|) \ = \ \sum_{j=1}^{s} \log(|p_j|), \end{aligned}$$

where the third inequality follows from (8). Moreover, we have $\log(z^H) + \log(p^-_{\max}) \geq \sum_{j=1}^{s} \log(|p_j|)$.

Case 2: $|\bar{S} \cap N^-|$ is odd.

In this case, the solution $S = \bar{S} \setminus \{j^-\}$ is considered when choosing the best solution in step 11. If $p_s > 0$, we obtain

$$\begin{aligned} 3 \cdot \log(z^H) &\geq 3 \cdot \max \left\{ \sum_{j \in \bar{S} \setminus \{j^-\}} \log(|p_j|),\ \log(|p_{j_1}|) + \log(|p_{j_2}|),\ \log(p^+_{\max}) \right\} \\ &\geq \sum_{j \in \bar{S} \setminus \{j^-\}} \log(|p_j|) + \log(|p_{j_1}|) + \log(|p_{j_2}|) + \log(p^+_{\max}) \\ &\geq \sum_{j \in \bar{S} \setminus \{j^-\}} \log(|p_j|) + \log(|p_{j^-}|) + \log(p_s) = \sum_{j=1}^{s} \log(|p_j|), \end{aligned}$$

by invoking (8) again. In this case, we also have $\log(z^H) + \log(p^-_{\max}) + \log(p^+_{\max}) \geq \sum_{j=1}^{s} \log(|p_j|)$.

Similarly, if $p_s < 0$, then

$$\begin{aligned} 3 \cdot \log(z^H) &\geq 3 \cdot \max \left\{ \sum_{j \in \bar{S} \setminus \{j^-\}} \log(|p_j|),\ \log(|p_{j_1}|) + \log(|p_{j_2}|) \right\} \\ &\geq \sum_{j \in \bar{S} \setminus \{j^-\}} \log(|p_j|) + 2 \left(\log(|p_{j_1}|) + \log(|p_{j_2}|) \right) \\ &\geq \sum_{j \in \bar{S} \setminus \{j^-\}} \log(|p_j|) + \log(|p_{j^-}|) + \log(|p_s|) = \sum_{j=1}^{s} \log(|p_j|). \end{aligned}$$

Moreover, we have $\log(z^H) + 2\log(p^-_{\max}) \geq \sum_{j=1}^{s} \log(|p_j|)$.

Summarizing all four cases, we always have $3 \cdot \log(z^H) \geq \sum_{j=1}^{s} \log(|p_j|)$.

Then, since $\sum_{j=1}^{s} \log(|p_j|)$ is an upper bound on the optimal objective value of the LP relaxation of the associated instance of KP with profits $\log(|p_j|)$ (see, e.g., [3]),

Table 2 Profits p_j and weights w_j of the items in Example 2 with items indexed in nonincreasing order of $\log(|p_j|)/w_j$

item j	1	2	3	4	5	6
p_j	2	$M+2$	$-(M+1)$	M	M	-1
w_j	1	M	M	M	M	M

we have $\sum_{j=1}^{s} \log(|p_j|) \geq \sum_{j \in S^*} \log(|p_j|) = \log(\prod_{j \in S^*} p_j) = \log(z^*)$ (clearly, $|S^* \cap N^-|$ must be even). This yields

$$3 \cdot \log(z^H) \geq \log(z^*) \Longleftrightarrow z^H \geq (z^*)^{1/3}$$

and proves the approximation ratio in (a).

Moreover, in all four cases the additive error in the logarithmic space can be bounded by $\max\{\log(p^+_{\max}),\ \log(p^-_{\max})\} + \log(p^-_{\max}) \leq 2 \cdot \log(p_{\max})$, which yields the approximation ratio in (b). □

The approximation ratios obtained by PRODUCT GREEDY are rather disappointing. The following example, however, shows that the analysis in the proof of Theorem 2 is asymptotically tight and that a considerable deviation from the greedy principle would be necessary to improve upon the obtained approximation ratios:

Example 2 Consider the instance of PKP given by the item profits and weights shown in Table 2 and a knapsack capacity of $C := 3M$ for some large, positive integer M.

Algorithm PRODUCT GREEDY first finds $\bar{S} = \{1, 2, 3\}$ in step 2, but has to remove item 3 in step 5 since $|\bar{S} \cap N^-| = 1$, which yields $S = \{1, 2\}$ with an objective value of $2(M+2)$. The best negative pair found in step 9 is given by $j_1 = 3$ and $j_2 = 6$, and has profit product $M + 1$. Finally, $j^+_{\max} = 2$ with $p_{j^+_{\max}} = p^+_{\max} = M + 2$ in step 10. Therefore, PRODUCT GREEDY returns the solution $\{1, 2\}$ with an objective value of $z^H = 2(M+2)$, while the optimal solution consists of items 2, 4, and 5 with objective value $z^* = (M+2)M^2$.

Acknowledgements The work of Ulrich Pferschy has been supported by the Field of Excellence "COLIBRI" at the University of Graz

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare that there is no conflict of interest.

References

1. D'Ambrosio, C., Furini, F., Monaci, M., Traversi, E.: On the product knapsack problem. Optim. Lett. **12**(4), 691–712 (2018)
2. Halman, N., Kovalyov, M., Quilliot, A., Shabtay, D., Zofi, M.: Bi-criteria path problem with minimum length and maximum survival probability. OR Spectr. **41**, 469–489 (2019)
3. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
4. Martello, S., Toth, P.: Knapsack problems: Algorithms and computer implementations. Wiley, New Jersey (1990)
5. Pferschy, U., Schauer, J., Thielen, C.: The product knapsack problem: Approximation and complexity. arXiv: 1901.00695. (2019)