UNIVERSIDADE ESTADUAL DE CAMPINAS
SISTEMA DE BIBLIOTECAS DA UNICAMP
REPOSITÓRIO DA PRODUÇÃO CIENTIFICA E INTELECTUAL DA UNICAMP

Versão do arquivo anexado / Version of attached file:

Versão do Editor / Published Version

Mais informações no site da editora / Further information on publisher's website:
https://www.sciencedirect.com/science/article/pii/S1877050923010335

DOI: https://doi.org/10.1016/j.procs.2023.08.260

Direitos autorais / Publisher's copyright statement:
©2023 by Elsevier. All rights reserved.

DIRETORIA DE TRATAMENTO DA INFORMAÇÃO

Cidade Universitária Zeferino Vaz Barão Geraldo
CEP 13083-970 – Campinas SP
Fone: (19) 3521-6493
http://www.repositorio.unicamp.br

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 223 (2023) 400–402

Procedia Computer Science

www.elsevier.com/locate/procedia

XII Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 2023)

Positional Knapsack Problem: NP-hardness and approximation scheme

(Brief Announcement)

Lehilton L. C. Pedrosa[a,1], Mauro R. C. da Silva[a,1], Rafael C. S. Schouery[a,1]

[a]*Institute of Computing, University of Campinas Av. Albert Einstein, 1251 – Cidade Universitária, Campinas – SP, Brazil, 13083-852*

Abstract

We present the Positional Knapsack Problem (PKP), show that it is NP-hard and admits a Fully Polynomial-Time Approximation Scheme (FPTAS). This problem is a variant of the classical Binary Knapsack Problem (KP) in which the contribution of an item to the objective function varies according to the position in which it is added. The change in the valuation adds new properties to the problem that do not hold for KP as PKP is not a generalization of KP. Our FPTAS is based on a dynamic programming algorithm and uses a recursive rounding approach, which is necessary since the objective function depends on each item's value and position.

© 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the XII Latin-American Algorithms, Graphs and Optimization Symposium

Keywords: Approximation Algorithms; NP-Hardness; FPTAS

1. Introduction

Packing problems are recurrent in the advertising layout of sites, such as Google and Mercado Livre (a large Latin American marketplace), where ads are displayed on a banner. The probability of receiving clicks and, thus, the expected revenue for displaying an ad varies with its proximity to the top. This leads to the problem of arranging these advertisements in the banner in a way similar to Binary Knapsack Problem (KP), with the difference that an ad's value is higher if placed near the top. To model this task, we introduce the Positional Knapsack Problem (PKP), a variant of the classical KP in which the value of an item varies with the position in which it is placed.

E-mail addresses: lehilton@ic.unicamp.br (Lehilton L. C. Pedrosa)., mauro.silva@ic.unicamp.br (Mauro R. C. da Silva)., rafael@ic.unicamp.br (Rafael C. S. Schouery).

[1] Supported by São Paulo Research Foundation (FAPESP) grants #2015/11937-9, #2016/23552-7, and #2020/13162-2 and National Council for Scientific and Technological Development (CNPq) grants #425340/2016-3, #312186/2020-7, and #311039/2020-0.
E-mail address: mauro.silva@ic.unicamp.br

1877-0509 © 2023 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the XII Latin-American Algorithms, Graphs and Optimization Symposium
10.1016/j.procs.2023.08.260

i	1	2	3	4	5	6
$s_i = v_i$	5	4	3	3	3	2

Table 1: Instance for the example of Figure 2.

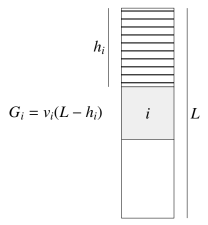

Fig. 1: Gain G_i of an item i with value v_i when packed at position h_i of a knapsack with capacity L.

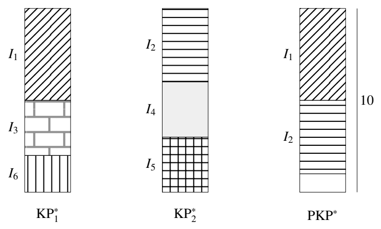

Fig. 2: Examples of optimal solutions for KP and PKP considering the items in Table 1.

Formally, an instance of PKP is composed of a knapsack of capacity L and a set of items $I = \{1, 2, \ldots, n\}$, where each item i has value $v_i > 0$ and size $s_i > 0$. The objective is to find a subset $S \subseteq I$ and the positions to pack S that maximizes the gain and does not exceed the capacity of the knapsack. The gain G_i of an item i is given by $v_i(L - h_i)$, where h_i is the position where the top of item i was added, and it corresponds to the total size of items preceding i in the knapsack. See an example in Figure 1, and notice that the packing starts at L and not at 0.

Several approaches for KP have been considered. We refer the reader to the two-part survey recently presented by Cacchiani et al. [1, 2] on KP and its variants, where exact and approximations algorithms are discussed. Gawiejnowicz et al. [3] consider knapsack problems with variable weights or profits of items. In these problems, an item's weight or profit depends on the item's index in the sequence of items packed in the knapsack. Note that these problems differ from PKP since, in PKP, the items vary according to the position in the knapsack, not with the index in the sequence of items. Gawiejnowicz et al. [3] presented FPTASes for the proposed problems considering monotonic functions.

Both KP and PKP have the same sets of inputs and feasible solutions. Despite this similarity, the items of an optimal KP solution do not necessarily form an optimal PKP solution for the same input even with $s_i = v_i$ for every item i. Consider an instance with the items described in Table 1 and a knapsack of capacity $L = 10$. Optimal solutions for KP with these items are $\mathrm{KP}_1^* = \{I_1, I_3, I_6\}$ and $\mathrm{KP}_2^* = \{I_2, I_4, I_5\}$. When we consider the objective function of PKP, the values of these solutions are 69 and 67, respectively. However, an optimal solution for PKP with these items is $\mathrm{PKP}^* = \{I_1, I_2\}$ and has value 70. Note that PKP^* is not an optimal solution for KP since $\sum_{i \in \mathrm{PKP}^*} v_i = 9$ and $\sum_{i \in \mathrm{KP}_1^*} v_i = \sum_{i \in \mathrm{KP}_2^*} v_i = 10$. Moreover, note that KP_1^* and KP_2^* are solutions for Partition Problem with this instance, and PKP^* is not. These solutions are shown in Figure 2.

We can show PKP is NP-hard even when every item has a value equal to its size by a reduction of the Equal-Cardinality Partition Problem, a variant of the Partition Problem. The idea behind this reduction is to have very large items such that an integer in the original instance is much smaller than the size of the generated item. This makes all items very similar in size, leading to some useful properties.

We present a pseudo-polynomial algorithm and an FPTAS for this problem. Since a solution of PKP includes the position in which items are packed, the objective function depends on the sizes and values of packed items. This is distinct from KP, whose objective function depends only on the values of packed items. To handle this difficulty, our FPTAS uses a recursive rounding approach.

2. Preliminaries

We define the *fullness* of a solution $S \subseteq I$ as the sum $\sum_{i \in S} s_i$, the *efficiency* of an item i as $e_i = v_i / s_i$ and denote the largest value of an item by $V_{max} = \max_{i=1}^{n} v_i$. In Lemma 1, we note that a solution of PKP can be represented only by the set of packed items, as we show that sorting the items in non-increasing order of efficiency maximizes the total gain. Note that when the efficiency of every item is equal to 1, that is, when $v_i = s_i$ for every item i, the value of a solution is the same for any order. However, as shown in Figure 2, the problem is still different from KP.

Lemma 1. *Let S be a solution that packs items $I' \subseteq I$ in an optimal order. Then, the packing order is non-increasing with regard to efficiency.*

3. Pseudo-polynomial time algorithm

We present a pseudo-polynomial algorithm for PKP similar to the dynamic programming algorithm presented by Ibarra and Kim [4] for KP. By Lemma 1, we assume that the items are sorted in non-increasing order of efficiency. The algorithm creates a matrix A with dimensions $n \times nLV_{max}$ where each entry $A(i, j)$ of the matrix corresponds to the minimum fullness of a solution considering only the first i items and with total gain exactly j. If there is no such solution, then $A(i, j) = \infty$. The following recurrence computes each entry of the matrix:

$$A(i,j) = \begin{cases} \infty, & \text{if } i = 1 \text{ and } Lv_1 \neq j, \\ s_1, & \text{if } i = 1 \text{ and } Lv_1 = j, \\ \min(A(i-1,j),\ \min_k A(i-1,k) + s_i), & \text{otherwise,} \end{cases}$$

where the second minimum in the third case ranges over every integer k such that $0 \leq k < j$ and $k + (L - A(i-1,k))v_i = j$. We can show that this recurrence is optimal for PKP. This algorithm's running time is $O(L^2 V_{max}^2 n^2)$, which is pseudo-polynomial in the instance size.

4. FPTAS

Based on the pseudo-polynomial algorithm, we present an FPTAS by rounding down the solution value. We assume that the indices of items are sorted in non-increasing order of efficiency accordingly to Lemma 1. Consider a constant ε such that $0 < \varepsilon < 1$, and define $\delta = \varepsilon/n$. Also, for each number $x \geq 0$, define $R_\delta(x)$ as the largest integer power of $(1+\delta)$ that is not greater than x. Therefore, if $R_\delta(x) = (1+\delta)^j$ for some j, then $(1+\delta)^j \leq x < (1+\delta)^{j+1}$. We say that a value x is *rounded down* to $R_\delta(x)$. Our algorithm creates a matrix A_δ with dimensions $n \times \lceil \log_{1+\delta} nLV_{max} \rceil$ such that each entry is computed using the following recurrence:

$$A_\delta(i,j) = \begin{cases} \infty, & \text{if } i = 1 \text{ and } R_\delta(Lv_1) < (1+\delta)^j, \\ s_1, & \text{if } i = 1 \text{ and } R_\delta(Lv_1) \geq (1+\delta)^j, \\ \min(A_\delta(i-1,j),\ \min_k A_\delta(i-1,k) + s_i), & \text{otherwise,} \end{cases}$$

where the second minimum in the third case ranges over every integer k such that $0 \leq k < j$ and also $R_\delta((L - A_\delta(i-1,k))v_i + (1+\delta)^k) \geq (1+\delta)^j$. The idea behind this recurrence is to use a dynamic programming algorithm similar to the one presented in Section 3, but approximating the value of a solution rounded down to a power of $1+\delta$. Each entry $A_\delta(i, j)$ of the matrix corresponds to the fullness of a solution considering only the first i items and with a total gain of at least $(1+\delta)^j$. If the algorithm cannot find a solution, it sets $A_\delta(i, j) = \infty$.

Let S be a solution with m items. We denote by S_i the i-th item of S in non-increasing order of efficiency. The total gain of S is defined as $f(S) = \sum_{i \in S} G_i$. The *rounded total gain* of S is defined recursively by letting

$$f_\delta(S) = \begin{cases} 0, & \text{if } m = 0, \\ R_\delta(G_{S_m} + f_\delta(S \setminus \{S_m\})), & \text{if } m \geq 1. \end{cases}$$

To show this algorithm is an FPTAS, we prove that this recurrence finds a solution S that maximizes $f_\delta(S)$ and prove that $f_\delta(S) \geq \exp(-2\varepsilon) f(\text{OPT})$. For every $0 < \varepsilon' < 1/2$, we can set $\varepsilon = \frac{1}{2}\ln(1/(1-\varepsilon'))$, such that $f(S) \geq \exp(-2\varepsilon) f(\text{OPT}) = (1-\varepsilon') f(\text{OPT})$. This algorithm's running time is $O\left(\frac{n^3 \log^2(nLV_{max})}{\varepsilon^2}\right)$, which is polynomial in $1/\varepsilon$ and in $1/\varepsilon'$ since $1/\varepsilon = 1/\left(\frac{1}{2}\ln\left(\frac{1}{1-\varepsilon'}\right)\right) = -2/(\ln(1-\varepsilon')) \leq 2/\varepsilon' = O(1/\varepsilon')$.

References

[1] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello. Knapsack problems — An overview of recent advances. Part I: Single knapsack problems. *Computers & Operations Research*, 143:105692, 2022. doi: 10.1016/j.cor.2021.105692.

[2] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello. Knapsack problems — An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143:105693, 2022. doi: 10.1016/j.cor.2021.105693.

[3] S. Gawiejnowicz, N. Halman, and H. Kellerer. Knapsack problems with position-dependent item weights or profits. *Annals of Operations Research*, pages 1–20, 2023.

[4] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.