Les Cahiers du GERAD ISSN: 0711–2440

A 3-space dynamic programming heuristic for the cubic knapsack problem

I. Dan Dije, F. Djeumou Fomeni, L. C. Coelho

G–2024–50

Août 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A 3-space dynamic programming heuristic for the cubic knapsack problem

Ibrahim Dan Dije [a]

Franklin Djeumou Fomeni [a]

Leandro C. Coelho [b]

[a] *GERAD, CIRRELT, and Department of Analytics, Operations and Information Technology, Montréal (Qc), Canada*

[b] *GERAD, CIRRELT, and Canada Research Chair in Integrated Logisitics, Université Laval, Montréal (Qc), Canada*

dan_dije.ibrahim@courrier.uqam.ca
djeumou_fomeni.franklin@uqam.ca
leandro.coelho@fsa.ulaval.ca

Août 2024
Les Cahiers du GERAD
G–2024–50

Abstract : The cubic knapsack problem (CKP) is a combinatorial optimization problem, which can be seen both as a generalization of the quadratic knapsack problem (QKP) and of the linear Knapsack Problem (KP). This problem consists of maximizing a cubic function of binary decision variables subject to one linear knapsack constraint. It has many applications in biology, project selection, capital budgeting problem, and in logistics. The QKP is known to be strongly NP-hard, which implies that the CKP is also NP-hard in the strong sense. Unlike its linear and quadratic counterparts, the CKP has not received much of attention in the literature. Thus the few exact solution methods known for this problem can only handle problems with up to 60 decision variables. In this paper, we propose a deterministic dynamic programming-based heuristic algorithm for finding a good quality solution for the CKP. The novelty of this algorithm is that it operates in three different space variables and can produce up to three different solutions with different levels of computational efforts. The computational results show that our algorithm can find optimal solutions for nearly 98% of the test instances that could be solved to optimality. It also has the merit of finding, in less than five minutes, feasible solutions that cannot be outperformed by commercial solvers in 5 hours. The algorithm also empirically dominates the other existing heuristic algorithm for CKP.

Keywords : Knapsack Problem, integer programming, dynamic programming, Cubic Knapsack Problem

Acknowledgements: This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grants 2021–03307 and 2019–00094. This support is greatly appreciated. We also thank the Digital Research Alliance of Canada for providing high-performance computing facilities.

1 Introduction

The 0-1 cubic knapsack problem (CKP) is a nonlinear binary optimization problem which is a general form of the binary quadratic knapsack problem (QKP) as well as of the linear knapsack problem (KP). The problem consists of maximizing a cubic objective function of binary decision variables subject to a single linear knapsack constraint. In the formulation of the CKP, we have three classes of profits in the objective function, namely, the individual profit of selecting a single item, the pairwise profit for pairs of items, and the cubic profit for a triplet of items selected simultaneously. Considering the set $N = \{1, \dots, n\}$ of n items, each item i has a weight a_i, an individual profit p_i for being selected alone, as well as two extra profits P_{ij} if selected with another item j, and D_{ijl} if selected with another pair of items j and l. Given a knapsack with a limited integer capacity c, the objective of the CKP is to find a subset of the items in N such that the total weight does not exceed the capacity c of the knapsack, while maximizing the overall profit. A mathematical formulation of CKP is therefore given by:

$$\begin{aligned} \max \ & \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} P_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{l=1 \\ l \neq i,\, l \neq j}}^{n} D_{ijl} x_i x_j x_l \\ \text{s.t.} \ & \sum_{i=1}^{n} a_i x_i \leq c, \\ & x \in \{0,1\}^n. \end{aligned} \tag{1}$$

The CKP model presented in (1) has a standard linearization due to Glover and Woolsey (1974), which consists of replacing each product $x_i x_j$ with a binary variable y_{ij}, and each product $x_i x_j x_l$ with another binary variable z_{ijl}. Then, we have the following model:

$$\max \ \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} P_{ij} y_{ij} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{l=1 \\ l \neq i,\, l \neq j}}^{n} D_{ijl} z_{ijl}$$

$$\begin{aligned} \text{s.t.} \ & \sum_{i=1}^{n} a_i x_i \leq c, \\ & y_{ij} \leq x_i, && \forall (i,j) \ \ i < j, \\ & y_{ij} \leq x_j, && \forall (i,j) \ \ i < j, \\ & y_{ij} \geq x_i + x_j - 1, && \forall (i,j) \ \ i < j, \\ & y_{ij} \geq 0, && \forall (i,j) \ \ i < j, \\ & z_{ijl} \leq x_i, && \forall (i,j,l) \ \ i < j < l, \\ & z_{ijl} \leq x_j, && \forall (i,j,l) \ \ i < j < l, \\ & z_{ijl} \leq x_l, && \forall (i,j,l) \ \ i < j < l, \\ & z_{ijl} \geq x_i + x_j + x_l - 2, && \forall (i,j,l) \ \ i < j < l, \\ & z_{ijl} \geq 0, && \forall (i,j,l) \ \ i < j < l, \\ & x \in \{0,1\}^n. \end{aligned} \tag{2}$$

Unlike the linear KP (Kellerer et al., 2004; Martello and Toth, 1990) and the QKP (Pisinger et al., 2007; Cacchiani et al., 2022; Fennich et al., 2024) that have been studied quite significantly, the CKP

has not received much attention in the literature. Studies related to this problem can be traced to the work of Forrester (2016), wherein they presented a tight relaxation of the linearization of the general form of 0-1 cubic program. Forrester and Waddell (2022) explicitly solved an exact model for the CKP given by (2) without the auxiliary constraints given by (3), handling problems with up to 60 items.

$$\begin{aligned} &y_{ij} \geq x_i + x_j - 1, && \forall (i,j) \;\; i < j, \\ &y_{ij} \geq 0, && \forall (i,j) \;\; i < j, \\ &z_{ijl} \geq x_i + x_j + x_l - 2, && \forall (i,j,l) \;\; i < j < l, \\ &z_{ijl} \geq 0, && \forall (i,j,l) \;\; i < j < l. \end{aligned} \tag{3}$$

It should be noted that applications of the family of knapsack problems have so far been limited to the linear KP (Salkin and De Kluyver, 1975) and to the QKP (Gallo et al., 1980). This is probably because considering a practical problem from a CKP point of view brings a much bigger challenge in terms of solution methodology. However, the addition of the cubic terms could add value to the problem itself as well as to the quality of the solution. For example, in the mean-variance portfolio selection (Faaland, 1974; Peterson and Laughhunn, 1971; Weingartner, 1966), while the quadratic terms represent the covariance between projects, cubic terms could be used to provide information about the skewness and the asymmetry between the projects. Some other applications can also be found in biology (Mohammadi et al., 2016). Elsewhere, the area of combinatorial auction (Cramton et al., 2010) can also provide an interesting application of CKP. Indeed, in combinatorial auctions participants can bid on combinations of items or packages. In preparing their bids, each participant aims to maximize their utility by selecting a subset of items to bid for. A CKP formulation can be suitable for representing the additional profit of simultaneously having pairs and triplets of items in a single bid, especially in areas such as logistics (Hammami et al., 2019, 2022), where the knapsack constraints may represent the capacity of a truck or container, as well as in cloud computing (Marinescu, 2018) wherein the quadratic and cubic profits may represent the benefit of having two or three jobs running simultaneously with a restricted server capacity.

The aim of this paper is to present a dynamic programming (DP) based deterministic heuristic solution method for the CKP. While DP algorithms have been extensively used in solving the KP and the QKP, we are the first to extend this methodology to the CKP. The novelty of our DP approach is that it operates in three different space variables to produce three good feasible solutions to the CKP with different levels of computational efforts. Additionally, we introduce an adaptation of the upper planes concept defined by Caprara et al. (1999) for a cubic objective function, which helps us to define some sorting criteria in order to enhance our algorithm. We conduct a vast array of computational experiments with a total of 680 test instances randomly generated following a well-known scheme from the literature. We evaluate a large number of capabilities of our proposed algorithm as well as a comparison with a MIP solver and the only heuristic algorithm for the CKP that can be found in the literature (Forrester and Waddell, 2022). The computational results show that our proposed algorithm can find optimal solutions for nearly 98% of the instances, while for the instances for which it cannot find optimal solutions, the optimality gaps are consistently below 0.05%. Furthermore, the results also show that a basic faster version of our algorithm matches the performance of the existing heuristic algorithm from the literature, both in terms of optimality gap and computational time, while a more advanced version of our algorithm dominates this existing heuristic.

In the remainder of this paper, we briefly review the literature related to our work in Section 2, then present some background necessary to understand the development of our algorithm in Section 3. The details of the technical components of our algorithm are discussed in Section 4. We show how the algorithm can be improved in Section 5 and present and analyze the different computational experiments in Section 6. Finally, some concluding remarks are presented in Section 7.

2 Literature review

The development of solution methodologies for the CKP that can be found in the literature is limited to the work of Forrester and Waddell (2022) where they presented a strong linear reformulation of the problem, which is then solved using commercial solvers. In addition, they developed a greedy-like heuristic method for obtaining feasible CKP solutions that are used to warm-start the solver. This greedy heuristic solution is an adaptation of the constructive heuristic solution developed by Billionnet and Calmels (1996) for the QKP. The heuristic starts with a greedy weight-to-benefit ratio to find an initial feasible solution, which is later improved with the well-known "fill-up and exchange" local search (Gallo et al., 1980). They reported computational results for CKP instances with up to 60 items. In this paper, we will use the heuristic of Forrester and Waddell (2022) as a benchmark for comparing our proposed algorithm, especially for large scale instances. More generally, the CKP can be tackled like more general cubic programs using linearization approaches and commercial solvers (Adams and Forrester, 2005; Forrester, 2016). However, the study of cubic programs is still rather limited in the literature and existing exact methods can only solve very small problem instances.

On the other hand, DP is known to be an efficient exact solution method for the linear KP (Bellman, 1957). However, considering the non-linear counterparts of the KP, the Bellman's principle of optimality no longer holds as shown by Fomeni and Letchford (2014). They also showed that one can, however, use the classical DP algorithm to yield a deterministic heuristic methodology for the QKP. Since then, two DP-inspired heuristic algorithms have been proposed for the QKP. The first consists of extending the original DP of Fomeni and Letchford (2014) to consider a potential selection of future items at each stage of the algorithm (Fennich et al., 2024), while the second one implements the DP on the lifted space of the quadratic variables (Fomeni, 2023). Both algorithms have shown to be particularly effective for difficult classes of the QKP.

The core of the algorithm presented in this paper is inspired by the latter DP algorithm, with the particularly that the algorithm travels through three spaces of variables, namely the space of linear variables, the space of quadratic variables, and the space of cubic variables. Each of the three spaces produces a feasible solution, which is fundamentally an improved version of the lower space variables since the solution of a higher space builds on the solution of lower spaces.

3 Background of the 3-space lifted DP

One very well-known approach for linearizing a 0-1 cubic program consists of introducing two new families of binary decision variables that will, respectively, replace the quadratic and the cubic terms in the objective function (Glover and Woolsey, 1974; Rader, 1997; Helmberg et al., 2000; Rader and Woeginger, 2002). Thus, considering the cubic objective function given in (1) with binary decision variables $x_1, \ldots, x_n$, the standard linearization technique will require the introduction of new binary variables $y_{ij} = x_i x_j$ (with $i < j$) and $z_{ijl} = x_i x_j x_l$ (with $i < j < l$) to replace the quadratic and the cubic terms in the objective function. This could also be seen from a hyper-graph point of view. Given a complete hyper-graph $G_H = (N, (E, \mathcal{E}))$, with N being the set of nodes, while E and $\mathcal{E}$ are the set of hyper-edges consisting of respectively, two and three nodes. For each node $i \in N$, there is an associated node profit $p_i \in \mathbb{Q}^+$; for each pairwise edge $e \in E$ with $e = (i, j)$, $i < j$, and $i, j \in N$, there is an associated edge profit $p_e \in \mathbb{Q}^+$; and for each hyper-edge $g \in \mathcal{E}$ with $g = (i, j, l)$ $i < j < l$, and $i, j, l \in N$, there is also an associated hyper-edge profit $d_g \in \mathbb{Q}^+$.

The objective function of the CKP given in (1) can be written as:

$$\sum_{i=1}^{n} p_i x_i + 2 \sum_{e \in E} p_e y_e + 6 \sum_{g \in \mathcal{E}} d_g z_g, \tag{4}$$

where $y_e = x_i x_j$, $z_g = x_i x_j x_l$, and the different profits defined by $p_e = P_{ij} = P_{ji}$ for $e = (i, j)$ and $d_g = D_{ijl} = D_{ilj} = D_{jil} = D_{jli} = D_{lij} = D_{lji}$ for $g = (i, j, l)$. These newly introduced variables are

often referred to as lifted space variables (Glover and Woolsey, 1974). More precisely, y_e is known as the quadratic space variable, while z_g is known as the cubic space variable.

In this paper, we present a deterministic DP heuristic algorithm that navigates through the three spaces of variables in order to produce three feasible solutions for the CKP. To the best of our knowledge, this algorithm is the second deterministic algorithm for this problem, with the first one being the greedy-like algorithm of Forrester and Waddell (2022). The first space through which our algorithm travels is defined by the linear variables $x_1, \ldots, x_n$. The second space will be the lifted space of the quadratic variables $y_{e_1}, \ldots, y_{e_m}$, with $m = |E| = \binom{n}{2}$ and $\{e_1, \ldots, e_m\} = \{(1,2), \ldots, (n-1, n)\}$. Thirdly, it will traverse the space of cubic variables defined by $z_{g_1}, \ldots, z_{g_t}$, with $t = |\mathcal{E}| = \binom{n}{3}$ and $\{g_1, \ldots, g_t\}$=$\{(1,2,3), \ldots, (n-2, n-1, n)\}$.

The DP approach for solving the linear KP is based on the Bellman's principle of optimality (Bellman, 1957), which uses the state function $f(k, r)$ to get the maximum profit that can be obtained by packing a selection of the k first items for a total weight of r. For the cubic CKP, one could define the function $f(k, r)$ as follows:

$$f(k,r) = \max \Bigg\{ \sum_{i=1}^{k} p_i x_i + \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} P_{ij} x_i x_j + \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} \sum_{\substack{l=1 \\ l \neq i, l \neq j}}^{k} D_{ijl} x_i x_j x_l : \sum_{i=1}^{k} a_i x_i = r,\ x \in \{0,1\}^k \Bigg\}. \tag{5}$$

Unfortunately, the Bellman's principle of optimality does not hold for nonlinear objective functions as shown by Fomeni and Letchford (2014) for the quadratic case. However, exploring the idea of Fomeni and Letchford (2014), Fomeni (2023), and Fennich et al. (2024), we will use a DP to find feasible solutions for the CKP. This is achieved by defining the state function $f(k, r)$ as the profit of the best packing found by the heuristic that uses a selection of the first k elements and whose total weights is equal to r. This state function will then be explored in the three different space variables presented above in order to increase the number of stages and states explored, thus giving a higher chance to the heuristic algorithm to find a good feasible solution. The exploration through the spaces of the quadratic and the cubic variables allows to better capture the interaction between pairs and triplets of items that can be selected simultaneously.

Note that from now onwards, $f(k, r)$ represents *the profit of the best packing found by the heuristic that uses a selection of the first k elements and whose total weights is equal to r*. In this definition, we use elements instead of items, since k will traverse the space of the linear variables where it will represent items, as well as the lifted space of the quadratic and cubic variables where it will represent pairs and triplets of items respectively. In other words, the index k will go from 1 to $n + |E| + |\mathcal{E}|$. Analogously, we will define the set $S(k, r)$ to encode the set of items producing such a best packing. Throughout the run of the index k between 1 and $n + |E| + |\mathcal{E}|$, one should remark that at each stage k, the algorithm considers the inclusion of:

- the item k when $1 \leq k \leq n$;
- the edge k represented by two items i and j when $n + 1 \leq k \leq n + |E|$;
- the hyper-edge represented by three items i, j, and l when $n + |E| + 1 \leq k \leq n + |E| + |\mathcal{E}|$.

The DP in this heuristic algorithm starts in the space of the linear variables, wherein the calculation of the profit contribution of the items and the weight utilization of the knapsack capacity is straightforward. However, when the algorithm gets into the two lifted spaces, one needs to pay careful attention to the calculation of the profit contribution and the weight utilization of edges and hyper-edges, since some of the items defining the edge or hyper-edge could have already been selected when travelling in the previous spaces.

4 3-space DP algorithm

Each stage of our DP algorithm is characterized by a value function $f(k, r)$, which represents the best profit found by the heuristic for a subset of the first k elements and for a total weight of r, as well as a set $S(k, r)$, which encodes the partial solution whose profit is equal to $f(k, r)$. In this section, we show how the calculation of the profit and the weight contribution for each element k is carried out in the DP algorithm in each of the three spaces.

4.1 Profit and weight in the linear space

The space of linear variables corresponds to $1 \leq k \leq n$. Hence, in this space the consideration of an element k corresponds to making the decision of whether to add the k-th item to the existing partial solution. In order to make this decision the profit and the weight contribution of these items are calculated as follows.

$$q_k = p_k + 2 \sum_{i \in S(k-1, r-a_k)} P_{ik} + 3 \sum_{m \in S(k-1, r-a_k)} \sum_{\substack{m' \in S(k-1, r-a_k), \\ m \neq m'}} D_{mm'k} \tag{6}$$

$$w_k = a_k. \tag{7}$$

The selection of the k-th item in the linear space variables brings, in terms of profit contribution, its individual profit as well as its pairwise and triplet-wise profit contributions with respect to the items already packed, i.e., the items in the set $S(k-1, r-w_k)$. The weight contribution of this item is simply its individual weight.

4.2 Profit and weight in the quadratic space

The space of the quadratic variables corresponds to $n+1 \leq k \leq n+|E|$, and the consideration of an element k in this space corresponds to making the decision of whether to add the k-th edge, formed by two nodes (items), or not. Since each of the two items forming the edge would have already been considered in the space of linear variables, it could happen that none of the items is part of the partially built solution, only one of them is already included in the partial solution, or that both items are already part of the partial solution. Thus, the profit and weight contributions of the edge will be calculated accordingly. More precisely, if $k = (i, j)$ then we can distinguish the following four cases (in Table 1) for which we show how the profit and weight contributions are calculated.

Table 1: Calculation of the profit and weight contributions of an element $k = (i, j)$ in the space of quadratic variables.

Cases	Description	Profit q_k	Weight w_k
E_1	i and j are unpacked	$p_i + p_j + 2 \sum_{i' \in S(k-1, r-a_i-a_j)} (P_{i'i} + P_{i'j}) + 2P_{ij} + 3 \sum_{\substack{m, m' \in S(k-1, r-a_i-a_j), \\ m \neq m'}} (D_{imm'} + D_{jmm'}) + 6 \sum_{m \in S(k-1, r-a_i-a_j)} D_{ijm}$	$a_i + a_j$
E_2	i is unpacked and j is packed	$p_i + 2 \sum_{i' \in S(k-1, r-a_i)} P_{i'i} + 3 \sum_{\substack{m, m' \in S(k-1, r-a_i), \\ m \neq m'}} D_{mm'i}$	a_i
E_3	i is packed and j is unpacked	$p_j + 2 \sum_{i' \in S(k-1, r-a_j)} P_{i'j} + 3 \sum_{\substack{m \in S(k-1, r-a_j), \\ m \neq m'}} D_{mm'j}$	a_j
E_4	i and j are packed	0	0

4.3 Profit and weight in the cubic space

The space of cubic variables is characterized by $n+|E|+1 \leq k \leq n+|E|+|\mathcal{E}|$. Thus the consideration of the k-th element in the DP algorithm corresponds to deciding whether to add the k-th hyper-edge, formed by three nodes (items), or not. Similarly to the case of the quadratic space, such an hyper-edge may have some of its nodes already included in the partial solution. This results into eight different cases to be distinguished when one computes the profit and weight contribution of the hyper-edge. If the hyper-edge is given by $k = (i, j, l)$ then we can distinguish the following eight cases (in Table 2) for the profit and weight contributions.

Table 2: Calculation of the profit and weight contributions of an element $k = (i, j, l)$ in the space of cubic variables.

Cases	Description	Profit q_k	Weight w_k
H_1	i, j, and l are unpacked	$p_i + p_j + p_l + 2(P_{ij} + P_{jl} + P_{il}) + 2 \sum_{i' \in S(k-1, r-a_i-a_j-a_l)} (P_{i'i} + P_{i'j} + P_{i'l}) + 6D_{ijl} + 3 \sum_{\substack{m,m' \in S(k-1, r-a_i-a_j-a_l), \\ m \neq m'}} (D_{imm'} + D_{jmm'} + D_{lmm'}) + 6 \sum_{m \in S(k-1, r-a_i-a_j-a_l)} (D_{ijm} + D_{ilm} + D_{jlm})$	$a_i + a_j + a_l$
H_2	i is packed, j and l are unpacked	$p_j + p_l + 2P_{jl} + 2 \sum_{i' \in S(k-1, r-a_j-a_l)} (P_{i'j} + P_{i'l}) + 3 \sum_{\substack{m,m' \in S(k-1, r-a_j-a_l), \\ m \neq m'}} (D_{jmm'} + D_{lmm'}) + 6 \sum_{m \in S(k-1, r-a_j-a_l)} D_{jlm}$	$a_j + a_l$
H_3	j is packed, i and l are unpacked	$p_i + p_l + 2P_{il} + 2 \sum_{i' \in S(k-1, r-a_i-a_l)} (P_{i'i} + P_{i'l}) + 3 \sum_{\substack{m,m' \in S(k-1, r-a_i-a_l), \\ m \neq m'}} (D_{imm'} + D_{lmm'}) + 6 \sum_{m \in S(k-1, r-a_i-a_l)} D_{ilm}$	$a_i + a_l$
H_4	l is packed, i and j are unpacked	$p_i + p_j + 2P_{ij} + 2 \sum_{i' \in S(k-1, r-a_i-a_j)} (P_{i'i} + P_{i'j}) + 3 \sum_{\substack{m,m' \in S(k-1, r-a_i-a_j), \\ m \neq m'}} (D_{imm'} + D_{jmm'}) + 6 \sum_{m \in S(k-1, r-a_i-a_j)} D_{ijm}$	$a_i + a_j$
H_5	i and j are packed, l is unpacked	$p_l + 2 \sum_{i' \in S(k-1, r-a_l)} P_{i'l} + 3 \sum_{\substack{m,m' \in S(k-1, r-a_l), \\ m \neq m'}} D_{lmm'}$	a_l
H_6	j and l are packed, i is unpacked	$p_i + 2 \sum_{i' \in S(k-1, r-a_i)} P_{i'i} + 3 \sum_{\substack{m,m' \in S(k-1, r-a_i), \\ m \neq m'}} D_{imm'}$	a_i
H_7	i and l are packed, j is unpacked	$p_j + 2 \sum_{i' \in S(k-1, r-a_j)} P_{i'j} + 3 \sum_{\substack{m,m' \in S(k-1, r-a_j), \\ m \neq m'}} D_{jmm'}$	a_j
H_8	i, j, and l are all packed	0	0

4.4 The 3-space DP algorithm

We now show the pseudocode of our proposed DP heuristic for the CKP in Algorithm 1. It should be noted that although the main body of the algorithm may look similar to that of other DP algorithms, there are significant differences in the implementation as our DP algorithm travels through three different space variables with the particularities highlighted above. At the end of the exploration in each of the space variables, it is possible to extract a complete solution. This results into three different solutions for the CKP when our DP algorithm comes to an end.

Algorithm 1: The 3-Space Dynamic programming for the CKP.

Initialize: $f(0,0)$ to 0, and $f(k,r)$ to $-\infty$ for $k = 1,\dots,n+|E|+|\mathcal{E}|$ and $r = 0,\dots,c$.
Initialize: $S(k,r) = \emptyset$, for all $k = 1,\dots,n+|E|+|\mathcal{E}|$ and $r = 0,\dots,c$.
for $k = 1 \dots n+|E|+|\mathcal{E}|$ **do**
  for $r = 0,\dots,c$ **do**
    if $f(k-1,r) > f(k,r)$ **then**
      Set $f(k,r) = f(k-1,r)$ and $S(k,r) = S(k-1,r)$
    end
    if $r + w_k \leq c$ **then**
      Let β be the profit of $S(k-1,r) \cup \{k\}$
      if $\beta > f(k, r+w_k)$ **then**
        $f(k, r+w_k) = \beta$
        $S(k, r+w_k) = S(k-1,r) \cup \{k\}$
      end
    end
  end
  if $k = n$ ***(Exit from the linear space)*** **then**

$$\text{Let } (k_L^*, r_L^*) = \arg \max_{\substack{0 \leq r \leq c \\ 1 \leq k \leq n}} f(k,r)$$

  end
  if $k = n + |E|$ ***(Exit from the quadratic space)*** **then**

$$\text{Let } (k_Q^*, r_Q^*) = \arg \max_{\substack{0 \leq r \leq c \\ n+1 \leq k \leq n+|E|}} f(k,r)$$

  end
  if $k = n + |E| + |\mathcal{E}|$ ***(Exit from the cubic space)*** **then**

$$\text{Let } (k_C^*, r_C^*) = \arg \max_{\substack{0 \leq r \leq c \\ n+|E|+1 \leq k \leq n+|E|+|\mathcal{E}|}} f(k,r)$$

  end
end
Return: $S(k_L^*, r_L^*)$, $S(k_Q^*, r_Q^*)$, and $S(k_C^*, r_C^*)$

At each stage of Algorithm 1, the parameter β represents the profit of the existing partial packing to which the k-th element is added. This parameter is therefore calculated as the state value of the given stage to which we add the profit contribution of element k, which itself is calculated using the formula corresponding to one of the cases discussed above. Note that for every stage (k,r) of the DP algorithm, the state value of the existing partial packing is already stored in $f(k-1, r-w_k)$. Hence, we only need to compute the value of q_k, which is the profit contribution of the k-th element. The calculation of q_k can be done in $\mathcal{O}(n^2)$ in all three space variables. This results in a time complexity of $\mathcal{O}(n^3 c)$ for the heuristic solution $S(k_L^*, r_L^*)$ of the linear space variables, $\mathcal{O}(n^4 c)$ for the heuristic solution $S(k_Q^*, r_Q^*)$ of the quadratic space variables, as well as $\mathcal{O}(n^5 c)$ for the heuristic solution $S(k_C^*, r_C^*)$ of the cubic space variables.

5 Improving the heuristic

In this section, we present two procedures for improving the performance of our proposed DP algorithm. The first procedure, which is implemented at the end of Algorithm 1, is the well known "fill-up and exchange" local search search procedure, which has been used in the QKP (Fomeni and Letchford, 2014; Billionnet and Calmels, 1996) as well as in the CKP (Forrester and Waddell, 2022). Given a solution, this procedure works in two phases. First, it attempts to fill up the residual space in the knapsack, if possible, then it tries to improve the solution by swapping one packed item with an unpacked one. In our approach, we independently apply the "fill-up and exchange" local search method to the solution of each of the three spaces.

The second enhancement procedure for our algorithm uses the idea of upper planes with an adaptation to the CKP to define sorting criteria that will be used to order the items before entering the linear space variables for the DP. Indeed, Caprara et al. (1999) defined improved upper planes to find upper bounds for the QKP. Later on these upper planes were used by Fomeni and Letchford (2014) in order to define sorting criteria for the items in their DP heuristic algorithm for the QKP. We define an adaptation of the latter idea to the case of CKP.

In the case of CKP, given a vector $\alpha \in \mathbb{Q}_+^n$, an upper plane for the objective function of the CKP is defined by a linear function $\alpha^T x$, which satisfies the following inequality:

$$\alpha^T x \geq \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} P_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{\substack{l=1 \\ l\neq i,\, l\neq j}}^{n} D_{ijl} x_i x_j x_l, \qquad \forall x \in \{0,1\}^n. \tag{8}$$

In our computational experiments, we have used four sets of upper planes ($\alpha^1, \alpha^2, \alpha^3$, and α^4), defined as follows.

$$\alpha_i^1 = \left\{ \left(p_i + 2 \sum_{\substack{j=1 \\ j\neq i}}^{n} P_{ij} + 3 \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{\substack{l=1 \\ l\neq i,\, l\neq j}}^{n} D_{ijl} \right) \right\}, \qquad \forall i = 1, \ldots, n. \tag{9}$$

By defining the parameter

$$\pi_{ij} = 2P_{ij} + 6 \sum_{\substack{l=1 \\ l\neq i,\, l\neq j}}^{n} D_{ijl}, \tag{10}$$

we also define

$$\alpha_i^2 = p_i + \max \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{n} \pi_{ij} x_j : \sum_{\substack{j=1 \\ j\neq i}}^{n} a_j x_j \leq c - a_i, \;\; x \in \{0,1\} \right\}, \qquad \forall i = 1, \ldots, n, \tag{11}$$

$$\alpha_i^3 = p_i + \max \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{n} \pi_{ij} x_j : \sum_{\substack{j=1 \\ j\neq i}}^{n} a_j x_j \leq c - a_i, \;\; x \in [0,1] \right\}, \qquad \forall i = 1, \ldots, n, \tag{12}$$

$$\alpha_i^4 = p_i + \max \left\{ 2 \sum_{\substack{j=1 \\ j\neq i}}^{n} P_{ij} x_j + 3 \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{\substack{l=1 \\ l\neq i,\, l\neq j}}^{n} D_{ijl} x_j x_l : \sum_{\substack{j=1 \\ j\neq i}}^{n} a_j x_j \leq c - a_i, \;\; x \in \{0,1\} \right\}, \quad \forall i = 1, \ldots, n. \tag{13}$$

For any item i, the value of α_i^1 can be computed in the $\mathcal{O}(n^2)$. For α_i^2 and α_i^3, they represent a binary and fractional KP, we can obtain their values respectively in $O(n^2c)$ and $O(n^2)$ time by using Bellman's principal of optimality for α_i^2 (Bellman, 1957) and advanced median-finding techniques of Balas and Zemel (1980) for α_i^3. It should be noted also that obtaining the value of α_i^4 in the latter equation amounts to solving a QKP instance. Since an optimal solution is not necessarily required for our purpose, we use the QKP greedy algorithm of Billionnet and Calmels (1996), which offers a linear time complexity.

For each given upper plane, the items are sorted in non-decreasing order of the ratio α_i/a_i. The DP algorithm will therefore use this order when considering the items in the space of linear variables.

6 Computational experiments

In this section, we present and discuss the results of the computational experiments carried out to assess the quality of our proposed heuristic algorithm. We conducted three sets of experiments, the first of which is aimed at assessing the impact of the different components of the algorithm. The second set of experiments is aimed at evaluating the quality of the solution obtained by our algorithm, while the third set of experiments solves larger CKP instances and compares the results with the other CKP heuristic that can be found in the literature (Forrester and Waddell, 2022).

Throughout these computational experiments, the optimal solutions, where possible, have been obtained by solving the linearized CKP formulation of Forrester and Waddell (2022) using Gurobi version 10.0 for Python. Our heuristic algorithm was coded in the C programming language and run on a single node of a computer equipped with a processor running at 2.65 GHz and 200 GB of RAM.

The data used for the computational experiments have been generated randomly using the same generation scheme from Forrester and Waddell (2022). It should be noted though that this CKP instance generation scheme is an adaptation of the standard instance generation scheme used in the QKP literature (Gallo et al., 1980; Caprara et al., 1999; Fomeni et al., 2022). For each of the instances, the weight a_i of each item is an integer number drawn from a uniform distribution in the interval $[10, 50]$, while the knapsack capacity c is an integer from a uniform distribution between 70 and $\sum_{i=1}^{n} a_i$. The non-zero objective coefficients p_i for all i, P_{ij} for (i, j), and D_{ijl} for (i, j, l) are integers from a uniform distribution in the interval $[1, 100]$. We also considered varying the density of the profit matrix, which represents the percentage of non-zero elements in the 3-dimensional profit matrix. We have thus generated instances with density $\Delta \in \{25\%, 50\%, 75\%, 100\%\}$.

6.1 Preliminary experiments

In this first set of computational experiments, we aim to assess the impact of the various components of our algorithm to identify which combination offers a better trade-off between the solution quality and the computational time. We should recall that the first component of our algorithm consists of sorting the items in the non-decreasing order of the ratio of the adapted upper planes with the items weights. The second component of the algorithm is the 3-space DP run, which is able to produce a feasible CKP solution after leaving each of the space variables. The third component is the "fill-up and exchange" local search procedure. Therefore, this preliminary study consists of determining which adapted upper plane sorting as well as which space variable of the DP provides a better quality solution. For this experiment, we generated 10 instances for each combination of size $n \in \{20, 25, 30, 40, 50\}$ and profit hyper-matrix density $\Delta \in \{25\%, 50\%, 75\%, 100\%\}$, for a total of 200 instances.

The results for these experiments are presented in Tables 3 and 4. More precisely, Table 3 presents the results of our algorithm when the items are not sorted in the first phase. Then, the results of the algorithm with the different orderings are shown in Table 4 for the four adapted upper planes described in Section 5. In these tables we report, for each combination of size and density, the average (out of

Table 3: Average percentage gaps of the three spaces when items are not sorted.

		Linear		Quadratic		Cubic	
Density	Size n	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
25	20	1.365	0.00	0.000	0.09	0.000	0.87
	25	0.803	0.01	0.000	0.19	0.000	2.18
	30	0.254	0.02	0.000	0.50	0.000	7.10
	40	1.224	0.04	0.000	1.18	0.000	21.98
	50	1.312	0.12	0.005	4.18	0.000	97.70
50	20	0.000	0.00	0.000	0.06	0.000	0.58
	25	0.195	0.01	0.000	0.22	0.000	2.54
	30	0.373	0.01	0.000	0.33	0.000	4.67
	40	0.120	0.04	0.007	1.55	0.007	28.97
	50	0.184	0.09	0.000	3.40	0.000	77.63
75	20	0.000	0.00	0.000	0.06	0.000	0.60
	25	0.111	0.01	0.000	0.18	0.000	2.11
	30	1.027	0.01	0.059	0.32	0.059	4.54
	40	0.030	0.03	0.000	1.13	0.000	21.05
	50	0.000	0.09	0.000	3.64	0.000	81.03
100	20	0.172	0.00	0.000	0.10	0.000	0.92
	25	0.676	0.01	0.000	0.13	0.000	1.57
	30	0.003	0.01	0.000	0.43	0.000	6.06
	40	0.515	0.03	0.008	0.91	0.008	16.92
	50	0.009	0.10	0.000	4.33	0.000	97.28
Avg		0.419	0.03	0.004	1.15	0.004	23.82

10) optimality gap given by the heuristic solutions, as well as the average computational time needed to achieve these gaps. More precisely, the columns "Gap", refer to the optimality gap of the solution of the DP algorithm in the linear, quadratic, and cubic spaces, respectively, while the columns "Time" refer to the computational time spent before obtaining the solution for each of the three spaces. Note that the optimality gaps here are computed as

$$\left(\frac{\text{Optimal value} - \text{Lower Bound}}{\text{Optimal value}}\right) \times 100.$$

Starting with the DP solution in the space of linear variables, one can observe that when the items are not sorted (Table 3), the solutions obtained are within 1.3% of optimality, with few cases where the optimal solution is found. From the results in Table 4, one can notice that this gap drops below 0.5% for the DP solution in the space of linear variables. This suggests that ordering the items before implementing the DP algorithm may significantly improve the quality of the solution obtained in the linear space variables when sorted based on α_i^3/a_i and α_i^4/a_i. Out of the four adapted upper planes used for sorting the items, the ratio α_i^4/a_i appears to dominate the other three sorting criteria overall in terms of the optimality gap.

When the DP moves into the space of quadratic variables, one can see that the quality of the solution is much more improved, with the algorithm obtaining an optimal solution for nearly all the instances. In fact, in this space, the optimality gaps are consistently below 0.05%. However, looking at Tables 3 and 4, there does not seem to be any impact of ordering the items on the quality of the solution of the DP in the quadratic space. On the contrary, there are some instances for which ordering the items before the DP algorithm deteriorates the quality of the solution in the quadratic space. In some rare cases, the solution of the linear space is better than those of the quadratic and cubic ones, as the local search applied at the end of each of the spaces was able to perform better when applied to the linear space solution. Finally, for these instances, the cubic space does not seem to offer much of an improvement with respect to the solution obtained in the quadratic space. The only difference observed is when $n = 50$ for $\Delta = 25\%$ in both tables.

Table 4: Average percentage gaps of the three spaces of the three spaces with different sorting strategies.

Sorting		α_i^1/a_i						α_i^2/a_i					
		Linear		Quadratic		Cubic		Linear		Quadratic		Cubic	
	Size n	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
25	20	0.000	0.00	0.000	0.17	0.000	0.99	0.000	0.00	0.000	0.18	0.000	0.97
	25	0.609	0.01	0.013	0.20	0.000	2.28	1.431	0.01	0.000	0.24	0.000	2.24
	30	0.289	0.02	0.000	0.52	0.000	7.20	0.403	0.02	0.000	0.58	0.000	7.27
	40	0.223	0.04	0.000	1.25	0.000	23.07	0.742	0.04	0.000	1.21	0.000	23.00
	50	0.348	0.11	0.005	4.49	0.000	99.17	0.766	0.11	0.000	4.33	0.000	100.73
50	20	0.000	0.00	0.000	0.07	0.000	0.61	0.000	0.00	0.000	0.12	0.000	0.62
	25	0.375	0.01	0.179	0.23	0.179	2.65	0.195	0.01	0.000	0.23	0.000	2.62
	30	0.187	0.01	0.000	0.34	0.000	4.72	0.388	0.01	0.000	0.34	0.000	4.76
	40	0.007	0.05	0.007	1.67	0.007	30.31	0.188	0.04	0.008	1.60	0.007	30.53
	50	0.184	0.10	0.000	3.64	0.000	79.70	0.000	0.10	0.000	3.51	0.000	80.89
75	20	0.063	0.00	0.000	0.07	0.000	0.63	0.063	0.00	0.000	0.12	0.000	0.63
	25	0.116	0.01	0.000	0.19	0.000	2.19	1.221	0.01	0.000	0.19	0.000	2.18
	30	0.093	0.01	0.000	0.34	0.000	4.60	0.940	0.01	0.000	0.35	0.000	4.69
	40	0.214	0.04	0.000	1.22	0.000	21.61	0.466	0.03	0.000	1.17	0.000	22.07
	50	0.066	0.09	0.000	3.81	0.000	82.77	0.000	0.09	0.000	3.65	0.000	84.02
100	20	0.000	0.00	0.000	0.10	0.000	0.96	0.000	0.00	0.000	0.17	0.000	0.97
	25	0.794	0.01	0.000	0.14	0.000	1.63	0.000	0.01	0.000	0.17	0.000	1.62
	30	0.000	0.01	0.000	0.45	0.000	6.17	0.031	0.01	0.000	0.45	0.000	6.30
	40	0.000	0.03	0.008	0.97	0.008	17.33	0.222	0.03	0.000	0.94	0.000	17.65
	50	0.000	0.12	0.000	4.53	0.000	98.94	0.000	0.11	0.000	4.38	0.000	100.25
Avg		0.175	0.03	0.011	1.22	0.010	24.38	0.366	0.03	0.000	1.20	0.000	24.70
Sorting		α_i^3/a_i						α_i^4/a_i					
25	20	0.000	0.00	0.000	0.17	0.000	0.97	0.000	0.00	0.000	0.10	0.000	0.95
	25	0.205	0.01	0.000	0.18	0.000	2.25	0.263	0.01	0.000	0.20	0.000	2.39
	30	0.404	0.02	0.000	0.48	0.000	7.09	0.413	0.02	0.000	0.53	0.000	7.62
	40	0.217	0.04	0.009	1.16	0.000	21.70	0.217	0.04	0.009	1.24	0.000	23.88
	50	0.347	0.11	0.005	4.11	0.000	96.61	0.347	0.12	0.005	4.38	0.000	104.07
50	20	0.000	0.00	0.000	0.07	0.000	0.60	0.000	0.00	0.000	0.07	0.000	0.64
	25	0.179	0.01	0.179	0.21	0.179	2.60	0.179	0.01	0.179	0.23	0.179	2.78
	30	0.193	0.01	0.000	0.32	0.000	4.62	0.193	0.01	0.000	0.35	0.000	4.99
	40	0.014	0.05	0.007	1.53	0.007	28.73	0.033	0.05	0.007	1.64	0.007	31.17
	50	0.000	0.10	0.000	3.32	0.000	77.18	0.184	0.10	0.000	3.56	0.000	83.50
75	20	0.229	0.00	0.000	0.06	0.000	0.62	0.229	0.00	0.000	0.07	0.000	0.66
	25	0.132	0.01	0.000	0.18	0.000	2.15	0.104	0.01	0.000	0.19	0.000	2.31
	30	0.093	0.01	0.000	0.31	0.000	4.55	0.093	0.01	0.000	0.34	0.000	4.91
	40	0.246	0.03	0.000	1.12	0.000	20.71	0.246	0.04	0.000	1.20	0.000	22.35
	50	0.000	0.10	0.000	3.47	0.000	80.02	0.000	0.10	0.000	3.71	0.000	86.08
100	20	0.000	0.00	0.000	0.09	0.000	0.94	0.000	0.00	0.000	0.10	0.000	1.02
	25	0.670	0.01	0.000	0.13	0.000	1.61	0.670	0.01	0.000	0.15	0.000	1.71
	30	0.000	0.01	0.000	0.42	0.000	6.13	0.000	0.01	0.000	0.46	0.000	6.60
	40	0.004	0.03	0.008	0.89	0.008	16.66	0.004	0.03	0.008	0.96	0.008	17.97
	50	0.000	0.12	0.000	4.14	0.000	95.84	0.000	0.13	0.000	4.43	0.000	101.47
Average		0.147	0.03	0.010	1.12	0.010	23.58	0.116	0.03	0.001	1.19	0.001	25.35

In terms of computational times, it is expected that the algorithm will require more time as the DP algorithm progresses into higher spaces. In fact, the computational time for the DP solution in the linear space for the largest instances is obtained in about 0.1 seconds, which grows to about 4.3 seconds for the quadratic space, and around 100 seconds for the cubic space. An interesting fact about these results is that the quadratic space DP solution offers near-optimal solutions for most of the instances, while its computational times are still low. For these instances, it offers a better trade-off between the quality of the solution obtained and the computational time among the three solutions.

6.2 Efficiency of the DP solution

This second set of experiments is aimed at showcasing the efficiency of our heuristic solutions compared to the heuristic algorithm proposed by Forrester and Waddell (2022). For these experiments, we use instances of small to medium size by generating 10 instances for each combination of size $n \in \{20, 25, 30, 40, 50, 60, 70, 80\}$ and profit hyper-matrix density $\Delta \in \{25\%, 50\%, 75\%, 100\%\}$, for a total of 320 instances.

In these experiments, we solve each instance to optimality twice with a time limit of 5 hours using the MIP solver. In the first run, we provide a starting solution to the solver, while in the second attempt, we do not provide our DP heuristic solution to the solver as a warmstart. Each of the instances is also solved using both our DP heuristic and the heuristic algorithm of Forrester and

Waddell (2022). The experiment results are shown in Tables 5 and 6, wherein the first and second columns describe the characteristic (size and density) of the instances. The columns "Gap (Opt)" of the Table 5, give the optimality gap of the solutions of the DP heuristic algorithm at each space (linear, quadratic, and cubic) and of the Forrester and Waddell (2022) heuristic, for the instances that are solved to optimality within the time limit of 5 hours with warmstart. Additionally, the numbers in brackets for this column, in the format (A/B) report B as the number of instances out of 10 that have been solved to optimality within the time limit, and A the number of times (out of B instances) that the DP heuristic solution has the same value as the optimal objective function value. The columns "#Better" give the number of times that the lower bound found by our DP heuristic for each space and that of Forrester and Waddell (2022) is better than the best lower bound found by the MIP solver for the instances where an optimal solution could not be found within the 5 hours time limit. Finally, the columns "Time" of Tables 5 and 6 report the computational times of our DP heuristic algorithm for each space, of the heuristic algorithm of Forrester and Waddell (2022), and of the exact solutions with and without initial solution, respectively.

Table 5: Comparison of solution approaches for solving CKP with warmstart.

		Linear			Quadratic			Cubic			Forrester and Waddell (2022)			Exact
Density	Size n	Gap(%) (Opt)	#Better	Time(s)	Gap(%) (Opt)	#Better	Time (s)	Gap (%) (Opt)	#Better	Time(s)	Gap (%) (Opt)	#Better	Time (s)	Time (s)
25	20	1.365(5/10)	*	0.00	0.000(10/10)	*	0.09	0.000(10/10)	*	0.87	1.088(8/10)	*	0.01	0.57
	25	0.803(8/10)	*	0.01	0.000(10/10)	*	0.19	0.000(10/10)	*	2.18	0.499(7/10)	*	0.01	2.29
	30	0.254(8/10)	*	0.02	0.000(10/10)	*	0.50	0.000(10/10)	*	7.10	0.795(8/10)	*	0.02	8.93
	40	1.224(6/10)	*	0.04	0.000(10/10)	*	1.18	0.000(10/10)	*	21.98	0.149(8/10)	*	0.04	399.27
	50	1.312(3/10)	*	0.12	0.005(9/10)	*	4.18	0.000(10/10)	*	97.70	0.054(6/10)	*	0.12	2197.25
	60	0.304(2/4)	3/6	0.21	0.000(4/4)	6/6	8.42	0.000(4/4)	6/6	229.46	0.667(2/3)	3/7	0.25	12 597.50
	70	0.000(1/1)	3/9	0.43	0.000(1/1)	9/9	19.46	0.000(1/1)	9/9	620.47	0.000(1/1)	3/9	0.34	16 204.67
	80	*	4/10	0.34	*	10/10	12.37	*	10/10	432.37	*	8/10	0.40	18 009.25
50	20	0.000(10/10)	*	0.00	0.000(10/10)	*	0.06	0.000(10/10)	*	0.58	0.398(9/10)	*	0.01	1.53
	25	0.195(9/10)	*	0.01	0.000(10/10)	*	0.22	0.000(10/10)	*	2.54	1.015(9/10)	*	0.01	3.62
	30	0.373(7/10)	*	0.01	0.000(10/10)	*	0.33	0.000(10/10)	*	4.67	0.302(7/10)	*	0.02	38.69
	40	0.120(7/10)	*	0.04	0.007(9/10)	*	1.55	0.007(9/10)	*	28.97	0.569(6/10)	*	0.06	436.23
	50	0.000(4/4)	3/6	0.09	0.000(4/4)	6/6	3.40	0.000(4/4)	6/6	77.63	0.000(5/5)	3/5	0.12	11 590.56
	60	0.000(1/1)	5/9	0.19	0.000(1/1)	9/9	7.10	0.000(1/1)	9/9	192.47	0.000(1/1)	4/9	0.25	16 606.82
	70	0.308(2/3)	3/7	0.42	0.000(3/3)	7/7	19.67	0.000(3/3)	7/7	637.47	0.000(2/2)	6/8	0.32	14 601.87
	80	*	3/10	0.55	*	10/10	27.68	*	10/10	986.73	*	9/10	0.53	18 008.37
75	20	0.000(10/10)	*	0.00	0.000(10/10)	*	0.06	0.000	10	0.60	1.324(9/10)	*	0.01	1.76
	25	0.111(8/10)	*	0.01	0.000(10/10)	*	0.18	0.000(10/10)	*	2.11	0.508(8/10)	*	0.01	8.34
	30	1.027(7/10)	*	0.01	0.059(9/10)	*	0.32	0.059(9/10)	*	4.54	0.553(6/10)	*	0.02	67.93
	40	0.030(8/10)	*	0.03	0.000(10/10)	*	1.13	0.000(10/10)	*	21.05	0.272(7/10)	*	0.06	2458.40
	50	0.000(3/3)	4/7	0.09	0.000(3/3)	7/7	3.64	0.000(3/3)	7/7	81.03	0.171(1/3)	7/7	0.10	15 403.74
	60	0.000(2/2)	3/8	0.11	0.000(2/2)	8/8	4.12	0.124(0/2)	1/8	111.97	0.000(2/2)	5/8	0.19	16 886.62
	70	*	3/10	0.28	*	10/10	12.67	*	10/10	412.72	*	8/10	0.33	18 004.22
	80	*	6/10	0.56	*	10/10	28.73	*	10/10	1038.15	*	10/10	0.42	18 008.05
100	20	0.172(7/10)	*	0.00	0.000(10/10)	*	0.10	0.000(10/10)	*	0.92	0.157(7/10)	*	0.01	3.44
	25	0.676(7/10)	*	0.01	0.000(10/10)	*	0.13	0.000(10/10)	*	1.57	0.155(8/10)	*	0.01	12.61
	30	0.003(9/10)	*	0.01	0.000(10/10)	*	0.43	0.000(10/10)	*	6.06	0.017(8/10)	*	0.02	70.07
	40	0.515(6/10)	*	0.03	0.008(9/10)	*	0.91	0.008(9/10)	*	16.92	0.318(6/10)	*	0.06	1880.59
	50	0.009(3/3)	4/7	0.10	0.000(4/4)	6/6	4.33	0.000(4/4)	6/6	97.28	0.000(3/3)	4/7	0.13	11 484.60
	60	*	6/10	0.11	*	10/10	4.60	0.000(1/1)	9/10	122.38	0.000(1/1)	9/10	0.19	18 002.81
	70	*	5/10	0.20	*	10/10	8.30	*	10/10	269.33	*	7/10	0.27	18 004.12
	80	*	2/10	0.34	*	10/10	16.71	*	10/10	601.94	*	10/10	0.42	18 010.32

(Opt): (A/B) where A is the number of solutions that this method finds optimal among the B instances proved optimal by the exact solver.

#Better: number of times this algorithm is better than the exact solver.

The results in Tables 5 and 6 reveal that out of the 320 instances used for this experiment and with a time limit of 5 hours, the MIP solver could obtain optimal solutions for 192 instances when a heuristic solution was given as a warmstart and for 190 instances when no initial solution was given (see gaps columns). This suggests that, at this stage, there is no significant evidence that an initial solution makes an impact in the performance of the solver. This is also revealed in the computational times by comparing the last column of the Tables 5 and 6, where there is no significant difference in the computation time of the solver with or without the initial solution. However, out of the 192 instances that were solved to optimality by the solver, our DP heuristic can match the optimal solution value for up to 142 instances for the linear space, 188 for the quadratic space, and 189 for the cubic space. Moreover, the results in columns "#Better" show that for all the instances that could not be solved to optimality by the MIP solver, the best solutions found by the solver after 5 hours are all worse than or equal to our DP heuristic solution values in the case of the quadratic and the cubic spaces, which only requires less than 30 seconds and 20 minutes in average as shown in columns "Time".

Table 6: Comparison of solution approaches for solving CKP without warmstart.

		Linear			Quadratic			Cubic			Forrester and Waddell (2022)			Exact
Density	Size n	Gap(%) (Opt)	#Better	Time (s)	Gap(%) (Opt)	#Better	Time (s)	Gap (%) (Opt)	#Better	Time(s)	Gap (%) (Opt)	#Better	Time (s)	Time (s)
25	20	1.365 (5/10)	*	0.00	0.000 (10/10)	*	0.09	0.000 (10/10)	*	0.87	1.088 (8/10)	*	0.01	0.57
	25	0.803 (8/10)	*	0.01	0.000 (10/10)	*	0.19	0.000 (10/10)	*	2.18	0.499 (7/10)	*	0.01	2.50
	30	0.254 (8/10)	*	0.02	0.000 (10/10)	*	0.50	0.000 (10/10)	*	7.10	0.795 (8/10)	*	0.02	8.97
	40	1.224 (6/10)	*	0.04	0.000 (10/10)	*	1.18	0.000 (10/10)	*	21.98	0.149 (8/10)	*	0.04	341.65
	50	1.312 (3/10)	*	0.12	0.005 (9/10)	*	4.18	0.000 (10/10)	*	97.70	0.054 (6/10)	*	0.12	2402.07
	60	0.026 (2/3)	4/7	0.21	0.000 (3/3)	7/7	8.42	0.000 (3/3)	7/7	229.46	0.667 (2/3)	3/7	0.25	13 065.14
	70	0.000 (1/1)	4/9	0.43	0.000 (1/1)	9/9	19.46	0.000 (1/1)	9/9	620.47	0.000 (1/1)	3/9	0.34	16 205.28
	80	*	5/10	0.34	*	10/10	12.37	*	10/10	432.37	*	8/10	0.40	18 009.25
50	20	0.000 (10/10)	*	0.00	0.000 (10/10)	*	0.06	0.000 (10/10)	*	0.58	0.398 (9/10)	*	0.01	1.50
	25	0.195 (9/10)	*	0.01	0.000 (10/10)	*	0.22	0.000 (10/10)	*	2.54	1.015 (9/10)	*	0.01	3.90
	30	0.373 (7/10)	*	0.01	0.000 (10/10)	*	0.33	0.000 (10/10)	*	4.67	0.302 (7/10)	*	0.02	41.50
	40	0.120 (7/10)	*	0.04	0.007 (9/10)	*	1.55	0.007 (9/10)	*	28.97	0.569 (6/10)	*	0.06	472.78
	50	0.184 (3/5)	3/5	0.09	0.000 (4/4)	5/5	3.40	0.000 (5/5)	5/5	77.63	0.000 (5/5)	3/5	0.12	11 127.68
	60	0.000 (1/1)	5/9	0.19	0.000 (1/1)	9/9	7.10	0.000 (1/1)	9/9	192.47	0.000 (1/1)	4/9	0.25	16 560.95
	70	0.462 (1/2)	6/8	0.42	0.000 (2/2)	8/8	19.67	0.000 (2/2)	8/8	637.47	0.000 (2/2)	6/8	0.32	14 953.95
	80	*	10/10	0.55	*	10/10	27.68	*	10/10	986.73	*	6/10	0.53	18 008.37
75	20	0.000 (10/10)	*	0.00	0.000 (10/10)	*	0.06	0.000 (10/10)	*	0.60	1.324 (9/10)	*	0.01	1.95
	25	0.111 (8/10)	*	0.01	0.000 (10/10)	*	0.18	0.000 (10/10)	*	2.11	0.508 (8/10)	*	0.01	9.51
	30	1.027 (7/10)	*	0.01	0.059 (9/10)	*	0.32	0.059 (9/10)	*	4.54	0.553 (6/10)	*	0.02	77.17
	40	0.030 (8/10)	*	0.03	0.000 (10/10)	*	1.13	0.000 (10/10)	*	21.05	0.272 (7/10)	*	0.06	2667.80
	50	0.000 (3/3)	4/7	0.09	0.000 (3/3)	7/7	3.64	0.000 (3/3)	6/6	81.03	0.171 (1/3)	5/7	0.10	15 679.34
	60	0.000 (2/2)	3/8	0.11	0.000 (2/2)	8/8	4.12	0.000 (2/2)	8/8	111.02	0.000 (2/2)	2/8	0.14	19 738.57
	70	0.000 (2/2)	7/8	0.18	0.000 (2/2)	8/8	10.31	0.000 (2/2)	8/8	472.85	0.000 (2/2)	7/8	0.23	18 946.83
	80	*	9/10	0.24	*	9/10	19.12	*	10/10	546.73	*	9/10	0.37	13 646.26
100	20	0.172 (7/10)	*	0.00	0.000 (10/10)	*	0.10	0.000 (10/10)	*	0.92	0.157 (7/10)	*	0.01	3.46
	25	0.676 (7/10)	*	0.01	0.000 (10/10)	*	0.13	0.000 (10/10)	*	1.57	0.155 (8/10)	*	0.01	15.16
	30	0.003 (9/10)	*	0.01	0.000 (10/10)	*	0.43	0.000 (10/10)	*	6.06	0.017 (8/10)	*	0.02	89.07
	40	0.515 (6/10)	*	0.03	0.008 (9/10)	*	0.91	0.008 (9/10)	*	16.92	0.318 (6/10)	*	0.06	1574.51
	50	0.009 (3/3)	4/7	0.10	0.000 (3/3)	7/7	4.33	0.000 (3/3)	6/6	97.28	0.000 (3/3)	4/7	0.13	11 579.05
	60	*	10/10	0.11	*	10/10	4.60	*	10/10	122.38	*	9/10	0.19	18 002.82
	70	*	9/10	0.20	*	10/10	8.30	*	10/10	269.33	*	7/10	0.27	18 004.07
	80	*	10/10	0.34	*	10/10	16.71	*	10/10	601.94	*	10/10	0.42	18 008.10

(Opt): (A/B) where A is the number of solutions that this method finds optimal among the B instances proved optimal by the exact solver.

#Better: number of times this algorithm is better than the exact solver.

Regarding the greedy-like heuristic algorithm of Forrester and Waddell (2022), the results in column "Gap" reveal that their algorithm can only match optimal solutions for 142 instances out of 192. Moreover, their optimality gaps are most of the time larger than 0.5%. This algorithm has the merit of being fast to find its heuristic solution. Nevertheless, the gaps obtained by the heuristic solution of Forrester and Waddell (2022) can be compared with the gaps of our DP algorithm in the space of linear variables for comparable computational times. Overall, the results in Tables 5 and 6 show that for a class of strongly NP-Hard problem like the CKP, our proposed DP heuristic algorithm (in all the spaces) can be used as a fair alternative to the exact solution method specifically in the space of quadratic and cubic variables. Even though the use of such a good heuristic solution as warmstart does not seem to improve the performance of commercial solvers at this time, our experiments reveal that this is simply due to the fact that the continuous relaxation for the CKP found in the literature is still very weak.

6.3 Results for large instances

In this section, we conduct a third set of experiments with the aim of assessing the scalability of our proposed DP heuristic algorithm, as well as furthering the comparison between our algorithm and that of Forrester and Waddell (2022). For this set of experiments, we generated 10 instances for each combination of size $n \in \{90, 120, 150, 200\}$ and profit hyper-matrix density $\Delta \in \{25\%, 50\%, 75\%, 100\%\}$, for a total of 160 instances. The results are presented in Tables 7–10 for $n = 90, 120, 150, 200$. In these tables, we report, for each instance, the lower bound from our DP heuristic algorithm (solution from the linear, quadratic, and cubic spaces) in columns "LB", and the lower bound of the heuristic algorithm of Forrester and Waddell (2022), as well as the computational times of both algorithms in columns "Time" for each algorithm. All the best solutions are highlighted in bold.

The results from the Tables 7–10 show that out of the 160 instances tested, our proposed DP heuristic algorithm provides a better solution than the algorithm of Forrester and Waddell (2022) for almost instances for all the spaces, except for some cases of the linear space solution. For some

instances, the solutions provided by the two algorithms are equal. This suggests that our proposed algorithm is dominant in terms of the quality of the heuristic solution for the solution provided by all the spaces. The computational time of the algorithm of Forrester and Waddell (2022) remains very low, while the computational times of our proposed DP heuristic algorithm are also reasonable for the quality of the solutions obtained for all the spaces; instances of 200 items are solved within at most five minutes for our algorithm in the quadratic space and less than 1 minute in the linear space.

Table 7: Comparative results of large instances for $n = 90$.

	Linear		Quadratic		Cubic		Forrester and Waddell (2022)	
Instances	LB	Time (s)	LB	Time (s)	LB	Time (s)	LB	Time (s)
				$\Delta = 25$				
1	76 330	0.16	**77 282**	1.59	**77 282**	53.40	74 046	0.39
2	3 753 277	1.00	**3 754 107**	52.58	**3 754 107**	2139.25	3 753 277	1.01
3	**8 164 765**	1.49	**8 164 765**	98.98	**8 164 765**	4091.73	**8 164 765**	0.52
4	3 025 186	1.21	**3 121 213**	45.33	**3 121 213**	1821.56	3 108 689	0.50
5	**824 052**	0.47	**824 052**	14.29	**824 052**	206.61	821 723	0.90
6	359 469	0.31	**367 070**	5.73	**367 070**	206.61	361 506	0.62
7	7 041 130	1.41	**7 043 199**	96.01	**7 043 199**	4015.09	**7 043 199**	0.52
8	972 838	0.59	**979 581**	16.30	**979 581**	636.37	970 298	0.47
9	1 016 045	0.54	**1 018 817**	14.59	**1 018 817**	548.14	1 016 045	0.92
10	1 277 704	0.60	**1 281 347**	21.00	**1 281 347**	811.75	1 270 450	0.48
				$\Delta = 50$				
1	**7 720 778**	0.77	**7 720 778**	58.18	**7 720 778**	2389.13	**7 720 778**	0.51
2	**5 914 117**	0.66	**5 914 117**	45.82	**5 914 117**	1862.82	5 905 135	0.50
3	7 133 074	1.05	**7 149 040**	57.56	**7 149 040**	2350.57	7 133 074	0.77
4	**947 881**	0.11	**947 881**	7.89	**947 881**	302.07	**947 881**	0.43
5	**6 194 185**	0.72	**6 194 185**	45.02	**6 194 185**	1821.68	6 186 634	0.75
6	6 237 728	0.90	**6 243 393**	45.97	**6 243 393**	1861.30	6 198 411	0.76
7	**11 744 440**	1.22	**11 744 440**	80.18	**11 744 440**	3341.89	11 728 744	1.03
8	**17 459 040**	1.72	**17 459 040**	101.38	**17 459 040**	4227.32	**17 459 040**	0.52
9	**597 107**	0.22	**597 107**	5.67	**597 107**	211.56	593 591	0.60
10	7 524 118	1.31	**7 542 219**	53.08	**7 542 219**	2174.27	**7 542 219**	1.53
				$\Delta = 75$				
1	120 702	0.21	**135 680**	0.76	**135 680**	27.63	**135 680**	0.39
2	**22 120 049**	1.79	**22 120 049**	91.36	**22 120 049**	3812.39	22 111 362	1.04
3	16 325 675	1.63	**16 329 301**	76.26	**16 329 301**	3115.30	16 319 699	1.28
4	**11 727 974**	1.06	**11 727 974**	61.03	**11 727 974**	2488.97	11 722 401	1.01
5	**2 522 865**	0.23	**2 522 865**	11.56	**2 522 865**	439.90	**2 522 865**	0.87
6	7 175 279	0.75	**7 189 293**	40.19	**7 189 293**	1596.17	7 156 695	0.74
7	**5 498 445**	0.35	**5 498 445**	28.03	**5 498 445**	1106.90	**5 498 445**	0.47
8	**442 457**	0.11	**442 457**	3.09	**442 457**	113.09	440 932	0.59
9	**1 416 561**	0.23	**1 416 561**	8.03	**1 416 561**	299.77	**1 416 561**	0.84
10	21 911 649	1.52	**21 921 174**	91.77	**21 921 174**	3782.34	21 918 446	1.02
				$\Delta = 100$				
1	**34 875 004**	1.69	**34 875 004**	102.90	**34 875 004**	4279.40	**34 875 004**	0.50
2	**5 066 348**	0.37	**5 066 348**	21.04	**5 066 348**	807.82	**5 066 348**	0.68
3	**14 105 606**	0.84	**14 105 606**	56.50	**14 105 606**	2272.86	14 101 483	0.99
4	572 101	0.25	**573 351**	2.87	**573 351**	98.98	572 254	0.78
5	12 301 710	0.78	**12 302 734**	50.95	**12 302 734**	2044.73	**12 302 734**	0.97
6	14 165 308	1.22	**14 174 114**	58.11	**14 174 114**	2326.94	14 171 858	1.51
7	6 108 731	0.47	**6 114 324**	25.22	**6 114 324**	990.71	6 105 185	0.71
8	**5 775 282**	0.60	**5 775 282**	21.41	**5 775 282**	839.33	5 764 495	1.14
9	18 404 178	1.25	**18 407 089**	63.16	**18 407 089**	2575.17	18 361 312	0.77
10	**31 527 068**	2.26	**31 527 068**	100.72	**31 527 068**	4243.21	**31 527 068**	1.03

Table 8: Comparative results of large instances for $n = 120$.

	Linear		Quadratic		Cubic		Forrester and Waddell (2022)	
Instances	LB	Time (s)	LB	Time (s)	LB	Time (s)	LB	Time (s)
				$\Delta = 25$				
1	9 279 674	4.80	**9 456 200**	58.52	**9 456 200**	3015.63	**9 456 200**	1.19
2	**7 349 385**	2.80	**7 349 385**	48.85	**7 349 385**	2499.67	**7 349 385**	2.32
3	7 724 484	2.93	7 730 660	49.54	**7 734 043**	2510.18	7 702 921	2.95
4	2 008 274	2.59	**2 021 103**	17.43	**2 021 103**	786.67	2 004 783	2.06
5	12 945 189	3.17	**12 948 787**	73.36	**12 948 787**	3883.25	12 922 957	1.88
6	**199 764**	2.96	196 009	4.46	197 565	103.71	196 960	2.17
7	**1 863 707**	2.25	**1 863 707**	15.92	**1 863 707**	703.98	1 863 205	1.56
8	422 120	5.06	**426 893**	6.11	**426 893**	199.43	414 388	1.41
9	4 328 656	3.45	**4 442 244**	31.60	**4 442 244**	1491.79	**4 442 244**	1.10
10	642 489	4.35	**665 540**	7.07	**665 540**	267.46	664 711	3.21
				$\Delta = 50$				
1	**32 327 061**	3.42	32 325 449	86.64	32 325 449	4547.49	32 289 222	1.87
2	**31 438 837**	3.49	**31 438 837**	82.92	**31 438 837**	4327.89	31 417 140	3.10
3	22 145 293	4.93	**22 713 584**	67.27	**22 713 584**	3534.86	**22 713 584**	1.79
4	**34 019 718**	4.05	**34 019 718**	88.84	**34 019 720**	4628.94	34 013 193	2.44
5	15 308 033	2.65	**15 323 512**	49.21	**15 323 512**	2476.99	15 270 921	2.40
6	392 065	1.46	**396 594**	3.40	**396 594**	102.95	394 132	1.32
7	**39 837 844**	2.86	**39 837 844**	90.59	**39 837 844**	4874.88	**39 837 844**	2.41
8	**3 447 635**	1.42	3 443 790	13.07	**3 447 635**	610.90	3 431 079	1.50
9	11 872 578	4.08	**12 215 020**	38.42	**12 215 020**	2010.40	**12 215 020**	1.67
10	2 775 486	3.37	**2 883 063**	12.51	**2 883 063**	573.91	2 882 089	2.46
				$\Delta = 75$				
1	108 774	1.30	**112 417**	2.13	110 586	30.33	110 586	2.45
2	39 654 846	3.32	**39 658 846**	75.37	**39 658 848**	4026.15	39 657 451	1.81
3	348 836	1.93	349 356	3.32	**349 665**	67.78	348 291	2.98
4	**11 749 758**	1.15	**11 749 758**	26.89	**11 749 758**	1371.60	**11 749 758**	1.11
5	2 016 970	0.70	**2 017 881**	6.22	**2 017 881**	289.13	2 014 730	0.98
6	**5 517 389**	1.19	5 517 311	15.05	5 517 311	718.90	5 513 056	2.53
7	15 636 301	1.49	**15 639 736**	37.41	**15 639 736**	1922.98	**15 639 736**	1.70
8	8 383 633	2.16	**8 391 203**	21.42	**8 391 203**	1017.09	8 371 136	3.15
9	2 360 690	2.46	2 364 243	8.65	**2 366 496**	325.42	2 356 674	2.36
10	**52 477 323**	4.06	**52 477 323**	91.34	**52 477 323**	4864.30	**52 477 323**	2.48
				$\Delta = 100$				
1	23 571 631	3.18	**23 576 112**	41.15	**23 576 112**	2077.31	23 558 391	2.90
2	1 598 455	2.52	**1 602 873**	5.72	**1 602 873**	195.12	**1 602 873**	2.25
3	**81 711 960**	2.94	81 710 797	93.68	81 710 797	5172.31	81 710 797	1.26
4	**5 731 134**	1.51	**5 731 134**	11.92	**5 731 134**	556.00	5 720 711	1.02
5	**45 333 412**	4.14	45 331 466	68.25	45 331 466	3695.87	45 304 422	3.09
6	**1 610 855**	2.86	**1 615 386**	6.50	**1 615 386**	199.78	1 610 200	3.12
7	**71 827 652**	4.13	**71 827 652**	89.07	**71 827 652**	4926.31	71 820 271	1.23
8	**68 045 123**	3.46	**68 045 123**	84.66	**68 045 123**	4721.35	68 015 017	1.83
9	**38 545 748**	1.32	**38 545 748**	53.14	**38 545 748**	2992.18	**38 545 748**	1.76
10	**1 074 652**	1.72	**1 074 652**	4.00	**1 074 652**	130.39	1 062 777	1.35

Table 9: Comparative results of large instances for $n = 150$.

	Linear		Quadratic		Cubic		Forrester and Waddell (2022)	
Instances	LB	Time (s)	LB	Time (s)	LB	Time (s)	LB	Time (s)
				$\Delta = 25$				
1	307 563	5.83	304 099	6.88	**308 448**	281.06	**308 448**	4.12
2	26 994 650	8.91	**27 010 725**	218.52	**27 010 725**	14 610.62	26 915 655	3.66
3	58 068	7.27	**65 238**	5.40	**65 238**	78.14	60 697	4.78
4	18 144 024	11.11	**18 146 086**	175.14	**18 146 710**	11 392.92	18 142 475	5.87
5	10 366 626	3.67	**10 368 697**	107.10	**10 368 697**	7347.08	**10 368 697**	4.23
6	**488 280**	5.76	**488 280**	11.25	486 836	399.02	480 275	3.44
7	21 767 637	8.71	**21 779 488**	195.72	**21 779 488**	13 992.62	21 730 100	4.72
8	**13 702 545**	5.04	13 700 680	135.59	13 700 680	9515.59	**13 702 545**	2.25
9	12 617 318	8.79	**12 634 558**	131.67	**12 634 558**	8506.72	12 602 667	7.73
10	22 181 834	8.14	**22 182 190**	191.87	**22 182 190**	13 339.19	22 143 216	4.64
				$\Delta = 50$				
1	12 344 230	8.14	**12 355 819**	72.81	**12 355 819**	4267.36	12 329 270	5.24
2	28 107 394	7.70	28 121 147	137.30	**28 125 369**	8831.43	28 089 853	4.55
3	**13 146 261**	2.28	**13 146 261**	67.02	**13 146 261**	4291.24	**13 146 261**	2.07
4	13 281 059	9.65	**13 677 767**	69.82	**13 677 767**	4291.24	13 146 261	3.04
5	**42 160 379**	7.30	**42 160 379**	187.95	**42 160 379**	12 282.73	42 132 483	4.59
6	15 256 593	4.64	**15 258 160**	78.86	**15 258 160**	4887.55	15 235 640	4.19
7	**133 409**	3.23	131 723	4.42	**133 409**	86.67	126 860	2.41
8	61 388 796	7.63	**61 391 129**	239.52	**61 391 129**	15 804.01	61 357 865	3.60
9	**34 201 208**	7.99	**34 201 208**	166.53	**34 201 208**	10 690.14	34 127 260	3.43
10	122 923	11.34	121 923	8.92	121 984	82.12	**123 409**	3.21
				$\Delta = 75$				
1	61 591 944	7.75	**61 593 554**	185.12	**61 593 554**	12 039.04	61 587 152	7.06
2	7 264 166	6.46	7 269 268	32.02	**7 272 307**	1638.29	7 266 263	2.86
3	8 435 698	4.61	**8 447 654**	35.12	**8 447 654**	1938.58	8 412 294	2.92
4	9 300 181	5.83	**9 300 982**	38.76	**9 300 982**	2170.13	9 296 241	4.75
5	17 001 492	7.93	**17 015 404**	68.12	17 015 404	4032.46	16 963 389	5.20
6	2 205 730	4.78	**2 212 735**	14.45	**2 212 735**	610.28	2 202 779	3.54
7	**1 214 097**	5.66	1 213 559	9.85	1 213 559	351.61	1 195 051	3.43
8	**13 296 299**	4.51	**13 296 882**	51.42	**13 296 882**	3114.77	13 292 641	5.03
9	577 221	4.39	**583 210**	7.95	**583 126**	219.64	578 984	7.26
10	**78 355 623**	3.87	**78 355 623**	203.89	**78 355 623**	13 790.92	78 301 769	2.36
				$\Delta = 100$				
1	48 176 606	5.63	**48 178 048**	118.85	**48 178 048**	7631.35	48 124 309	4.44
2	**25 372 202**	3.40	**25 372 202**	67.67	**25 372 202**	4300.92	25 360 901	3.12
3	11 229 399	5.92	**11 230 009**	38.47	**11 230 009**	2105.02	**11 230 009**	8.47
4	31 565 982	5.82	**31 569 984**	86.75	**31 569 984**	5405.67	31 558 442	3.24
5	131 253 652	7.72	131 253 652	233.60	131 253 652	16 422.78	**131 256 999**	5.89
6	6 527 155	5.61	**6 537 469**	25.16	**6 537 469**	1294.17	**6 537 469**	4.77
7	2 474 233	2.90	**2 476 308**	10.37	**2 476 308**	514.64	**2 476 308**	3.35
8	**2 063 011**	3.07	**2 063 011**	9.73	**2 063 011**	464.67	**2 063 011**	5.02
9	**457 291**	4.19	456 139	6.22	456 386	144.56	456 290	4.87
10	16 939 371	6.86	16 943 967	54.32	**16 943 968**	3210.14	16 907 817	4.03

Table 10: Comparative results of large instances for $n = 200$.

	Linear		Quadratic		Cubic		Forrester and Waddell (2022)	
Instances	LB	Time (s)	LB	Time (s)	LB	Time (s)	LB	Time (s)
				$\Delta = 25$				
1	14 526 441	16.68	**14 532 003**	193.21	**14 532 003**	22 885.53	14 523 962	9.56
2	17 667 205	25.05	**17 682 838**	241.21	**17 682 838**	28 231.72	17 647 937	9.86
3	**12 725 623**	31.08	**12 725 623**	176.99	**12 725 623**	20 216.74	12 617 387	9.55
4	**14 039 991**	10.03	**14 039 991**	181.62	**14 039 991**	21 417.31	14 009 980	7.12
5	6 303 727	33.71	**6 428 801**	87.61	**6 428 801**	10 264.29	**6 428 801**	10.85
6	856 118	31.80	876 123	18.43	**881 283**	1440.98	837 196	4.06
7	66 970	15.79	70 620	7.46	**72 911**	190.63	66 245	7.50
8	**7 011 341**	19.37	**7 011 341**	93.38	7 009 593	10 567.24	6 983 943	8.93
9	12 565 363	28.21	**12 850 461**	173.60	**12 850 461**	20 456.30	12 835 633	9.21
10	11 475 591	20.94	**11 479 185**	153.28	**11 479 185**	17 892.96	11 443 823	11.84
				$\Delta = 50$				
1	117 249	17.75	**130 623**	4.55	**130 623**	195.53	129 952	12.74
2	284 341	17.45	**288 386**	9.58	**288 386**	354.62	272 635	5.67
3	16 437 953	17.53	16 437 206	108.88	**16 449 730**	12 751.87	16 427 721	9.07
4	35 216 784	15.10	**35 223 879**	233.23	**35 223 879**	27 722.76	35 178 681	7.75
5	**374 397**	11.85	**374 397**	8.41	**374 397**	187.20	363 242	7.44
6	326 902	22.23	**331 424**	11.51	327 543	1334.30	327 543	18.52
7	6 956 626	28.04	7 171 506	48.95	**7 119 485**	8745.12	**7 119 485**	6.45
8	19 465 535	21.17	**19 468 568**	138.51	**19 468 568**	13 903.72	19 440 834	14.07
9	**35 126 361**	17.63	**35 126 361**	235.45	**35 126 361**	28 859.32	35 084 317	9.82
10	9 449 363	20.11	**9 465 478**	67.24	**9 465 478**	9528.47	9 445 162	13.03
				$\Delta = 75$				
1	**5 532 892**	10.11	**5 532 426**	29.46	**5 532 426**	3041.64	5 524 090	8.20
2	25 402 818	17.97	**25 408 546**	109.63	**25 408 546**	13 494.20	25 399 252	13.28
3	**1 846 174**	4.58	**1 846 174**	10.51	**1 846 174**	1065.76	**1 846 174**	5.84
4	89 998 736	26.44	**90 032 846**	389.50	**90 032 846**	47 557.50	89 867 159	10.74
5	**52 488 416**	13.17	**52 488 416**	225.12	**52 488 416**	27 732.26	**52 488 416**	9.73
6	**12 742 285**	11.88	**12 742 285**	55.49	**12 742 285**	6392.49	**12 742 285**	6.45
7	1 107 329	15.67	1 109 177	11.99	**1 109 240**	721.21	**1 109 240**	13.57
8	1 752 988	11.85	**1 755 917**	13.52	**1 755 917**	1103.22	**1 755 917**	7.80
9	**1 469 203**	17.20	**1 469 203**	14.35	**1 469 203**	894.05	1 468 598	7.78
10	**94 118 431**	10.79	**94 118 431**	389.05	**94 118 431**	48 695.73	94 077 164	5.34
				$\Delta = 100$				
1	107 005 616	15.24	**107 006 473**	345.88	**107 006 473**	41 376.87	**107 006 473**	7.76
2	64 456 101	19.14	64 455 120	215.68	**64 458 135**	25 127.08	64 440 735	12.13
3	64 490 929	24.49	**64 495 127**	221.83	**64 495 127**	26 286.00	64 440 545	12.26
4	71 821 586	13.30	**71 833 210**	230.40	**71 833 210**	28 463.05	71 823 054	9.82
5	10 672 204	12.09	**10 681 163**	39.50	**10 681 163**	4364.95	**10 678 204**	8.16
6	**14 878 153**	18.01	**14 878 153**	53.27	**14 878 153**	5893.01	14 860 968	8.41
7	**33 819 301**	21.98	**33 819 301**	114.60	**33 819 301**	13 682.30	33 804 255	19.89
8	112 119 709	17.22	**112 124 156**	344.77	**112 124 156**	42 432.65	**112 124 156**	10.30
9	17 698 429	13.26	**17 723 212**	63.06	**17 723 212**	7158.17	17 665 120	6.57
10	18 499 129	8.18	**18 499 325**	61.09	**18 499 325**	7410.72	**18 499 325**	8.55

7 Conclusion

In this paper, we have presented a DP-based deterministic heuristic algorithm for the CKP. The overall algorithm has three phases, which start with an item ordering, then a main DP framework, and ends with the well-known "fill-up-and-exchange" local search procedure. The novelty in this proposed algorithm is twofold. Firstly, we define an adaptation of the notion of upper planes in order to define sorting criteria for the first phase of the algorithm. Secondly, the main DP framework of the algorithm is a journey through three different spaces of variables, wherein each space produces a feasible solution for the CKP.

We have conducted a vast array of computational tests with a total of 680 test instances, in which we evaluate a large number of capabilities of our proposed algorithm, as well as provide comparisons with the other existing heuristic algorithm for the CKP. The computational results show that our proposed algorithm can find optimal solutions for nearly 98% of the instances that could be solved to optimality, while for the instances for which it cannot find an optimal solution, the optimality gaps are consistently below 0.05% and significantly outperform an exact MIP solver. Furthermore, the results also show that a basic version of our algorithm can match the performance of the existing heuristic from the literature in terms of optimality gap and computational time, while a more advanced version of our algorithm dominates this existing heuristic. Finally, they also show that our algorithm scales well for larger CKP instances, while still dominating the existing heuristic for this problem.

References

Warren P Adams and Richard J Forrester. A simple recipe for concise mixed 0-1 linearizations. Operations Research Letters, 33(1):55–61, 2005.

Egon Balas and Eitan Zemel. An algorithm for large zero-one knapsack problems. operations Research, 28(5): 1130–1154, 1980.

Richard Bellman. Dynamic programming. Princeton University Press, Princeton, New Jersey, 1957.

Alain Billionnet and Frédéric Calmels. Linear programming for the 0–1 quadratic knapsack problem. European Journal of Operational Research, 92(2):310–325, 1996.

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.

Alberto Caprara, David Pisinger, and Paolo Toth. Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing, 11(2):125–137, 1999.

Peter Cramton, Yoav Shoham, and Richard Steinberg. Combinatorial Auctions. The MIT Press, 2010.

B. Faaland. An integer programming algorithm for portfolio selection. Management Science, 20:1376–1384, 1974.

M Eliass Fennich, Franklin Djeumou Fomeni, and Leandro C Coelho. A novel dynamic programming heuristic for the quadratic knapsack problem. European Journal of Operational Research, 2024.

Franklin Djeumou Fomeni. A lifted-space dynamic programming algorithm for the quadratic knapsack problem. Discrete Applied Mathematics, 335:52–68, 2023.

Franklin Djeumou Fomeni and Adam N Letchford. A dynamic programming heuristic for the quadratic knapsack problem. INFORMS Journal on Computing, 26(1):173–182, 2014.

Franklin Djeumou Fomeni, Konstantinos Kaparis, and Adam N. Letchford. A cut-and-branch algorithm for the quadratic knapsack problem. Discrete Optimization, 44:100579, 2022.

Richard J Forrester. Tightening concise linear reformulations of 0-1 cubic programs. Optimization, 65(4): 877–903, 2016.

Richard J Forrester and Lucas A Waddell. Strengthening a linear reformulation of the 0-1 cubic knapsack problem via variable reordering. Journal of Combinatorial Optimization, 44(1):498–517, 2022.

Giorgio Gallo, Peter L Hammer, and Bruno Simeone. Quadratic knapsack problems. Combinatorial Optimization, pages 132–149, 1980.

Fred Glover and Eugene Woolsey. Converting the 0-1 polynomial programming problem to a 0-1 linear program. Operations Research, 22(1):180–182, 1974.

Farouk Hammami, Monia Rekik, and Leandro C. Coelho. Exact and heuristic solution approaches for the bid construction problem in transportation procurement auctions with a heterogeneous fleet. Transportation Research Part E: Logistics and Transportation Review, 127:150–177, 2019.

Farouk Hammami, Monia Rekik, and Leandro C. Coelho. An exact method for the combinatorial bids generation problem with uncertainty on clearing prices, bids success, and contracts materialization. Computers & Operations Research, 148:105982, 2022.

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

Hans Kellerer, Ulrich Pferschy, and David Pisinger. Knapsack Problems. Springer, Berlin, 2004.

Dan C Marinescu. Cloud resource management and scheduling. Cloud Computing, pages 321–363, 2018.

S. Martello and P. Toth. Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester, 1990.

Shahin Mohammadi, David F Gleich, Tamara G Kolda, and Ananth Grama. Triangular alignment (tame): A tensor-based approach for higher-order network alignment. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14(6):1446–1458, 2016.

D. E. Peterson and D. J. Laughhunn. Capital expenditure programming and some alternative approaches to risk. Management Science, 17:320–336, 1971.

D. Pisinger, A.B. Rasmussen, and R. Sandvik. Solution of large quadratic knapsack problems through aggressive reduction. INFORMS Journal on Computing, 19:280–290, 2007.

D.J. Rader. Lifting results for the quadratic 0-1 knapsack polytope. Technical Report Cosmic Program MFS-287000, Rutgers University, 1997.

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

Harvey M Salkin and Cornelis A De Kluyver. The knapsack problem: a survey. Naval Research Logistics Quarterly, 22(1):127–144, 1975.

H Martin Weingartner. Capital budgeting of interrelated projects: survey and synthesis. Management Science, 12(7):485–516, 1966.