Datalogisk Institut
KØBENHAVNS UNIVERSITET

DIKU

Heuristic approaches for the two- and three-dimensional knapsack packing problems

David Pisinger and Jens Egeblad

Technical Report no. 06/13
ISSN: 0107-8283

Dept. of Computer Science
University of Copenhagen • Universitetsparken 1
DK-2100 Copenhagen • Denmark

Heuristic approaches for the two- and three-dimensional knapsack packing problems*

Jens Egeblad and David Pisinger

Department of Computer Science, University of Copenhagen,
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark,
jegeblad@diku.dk, pisinger@diku.dk

December 2006

Abstract

The maximum profit two- or three-dimensional knapsack packing problem asks to pack a maximum profit subset of some given rectangles or boxes into a larger rectangle or box of fixed dimensions. Items must be orthogonally packed, but no other restrictions are imposed to the problem. The problem could also be considered as a knapsack problem generalized to two or three dimensions. In this paper we present a new heuristic based on the sequence pair representation proposed by Murata et al. (1996) using a semi-normalized packing by Pisinger (2006) for the two-dimensional knapsack problem. A local search algorithm maintains a pair of sequences given as permutations of the item numbers. In each step a neighbor solution is generated by making a small permutation in one or both sequences. The new sequence pair is transformed to a packing and the corresponding objective function is evaluated. Solutions are accepted based on a Simulated Annealing. The heuristic is also able to handle problem instances where rotation is allowed. A similar approach with a novel abstract representation of box placements, called sequence tripple, has been developed for the three-dimensional knapsack problem. Comprehensive computational experiments comparing the developed heuristics with previous approaches indicate that the results are very promising for both two- and three-dimensional problems.

Keywords: Cutting and Packing, knapsack, 2D knapsack, 3D knapsack, sequence pair, abstract representation, heuristic, simulated annealing

1 Introduction

Given a set of n rectangles $j = 1,\dots,n$, each having a width w_j, height h_j and profit p_j and a rectangular plate having width W and height H. The *maximum profit two-dimensional knapsack*

*DIKU Technical-report no. 2006-13, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark.

packing problem (2DKP) asks to assign a subset of the rectangles onto the plate such that the associated profit sum is maximized. All coefficients are assumed to be nonnegative integers, and the rectangles may not be rotated. A packing of rectangles on the plate is feasible if no two rectangles overlap, and if no part of any rectangle exceeds the plate.

The *maximum profit three-dimensional knapsack packing* problem (3DKP) asks to assign a subset of boxes each with dimensions w_j, h_j, d_j into a larger box with dimensions W, H and D but is otherwise similar.

The problem has direct applications in various packing and cutting problems where the task is to use the space or material in an optimal way. The 2DKP problem also appears as pricing problem when solving the two-dimensional bin-packing problem [4, 15, 16]. 2DKP and 3DKP are NP-hard in the strong sense, which can be shown by reduction from the one-dimensional bin packing problem.

Integer Programming formulations of the 2DKP have been presented by Beasley [1], Hadjiconstantinou and Christofides [7], and Boschetti, Hadjiconstantinou, Mingozzi [2] among others.

Fekete and Schepers [4, 5, 6] solved the 2- and 3DKP through a branch-and-bound algorithm which assigns items to the knapsack without specifying the position of the rectangles. For each assignment of items a two-dimensional packing problem is solved, deciding whether a feasible assignment of coordinates to the items is possible such that they all fit into the knapsack without overlaps. An advanced graph representation was used for solving the latter problem. Pisinger and Sigurd [16] solved the 2DKP through a branch-and-cut approach in which an ordinary one-dimensional knapsack problem is used to select the most profitable items whose overall area does not exceed the area of the plate. Having selected the most profitable items, a two-dimensional packing problem in decision form is solved, through constraint programming. If all items can be placed in the knapsack the algorithm terminates, otherwise an inequality is added to the one-dimensional knapsack stating that not all the current items can be selected simultaneously, and the process is repeated. Finally, Caprara and Monaci [3] developed a branch-and-bound algorithm for the 2DKP. The algorithm is based on a branch-and-bound scheme which assigns items to the knapsack without specifying the position of each item, followed by a feasibility check. The latter is done using an enumeration scheme from Martello, Monaci, Vigo [10].

In the present paper we first present an IP formulation of the 2- and 3DKP. In Section 3 we introduce the sequence pair representation, which we use in Section 4 combined with a simple local search neighborhood and Simulated Annealing to solve 2DKP. In Section 5 we introduce a novel abstract representation of box placements in three dimensions and use the same methods as for two dimensions to solve 3DKP. Finally in Section 6 we present our result on existing and new benchmarks instances for 2- and 3DKP.

2 Integer Programming Formulation of the problem

In the following we show an integer programming formulation of the 3DKP. A formulation of 2DKP easily follows by removing variables and constraints for the third dimension.

We will introduce the decision variable s_i to indicate whether box i is packed within the knapsack box. The coordinates of box i are (x_i, y_i, z_i), meaning that the lower left back corner of the box is located at this position. If a rectangle is not packed within the knapsack we may assume that $(x_i, y_i, z_i) = (0,0,0)$. As no part of a packed box may exceed the knapsack, we have the obvious constraints

$$0 \le x_i \le W - w_i, \qquad 0 \le y_i \le H - h_i, \qquad 0 \le z_i \le D - d_i. \tag{1}$$

We introduce the binary decision variables ℓ_{ij} (left), r_{ij} (right), u_{ij} (under), o_{ij} (over), b_{ij} (behind) and f_{ij} (in-front), to indicate the relative position of boxes i, j where $i < j$. To ensure that no two packed boxes i, j overlap we will demand that

$$\ell_{ij} + r_{ij} + u_{ij} + o_{ij} + b_{ij} + f_{ij} \ge 1, \tag{2}$$

whenever $s_i = s_j = 1$. Depending on the relative position of two rectangles the coordinates must satisfy the following inequalities

$$\begin{array}{llll} \ell_{ij} = 1 \;\Rightarrow\; x_i + w_i \le x_j, & & r_{ij} = 1 \;\Rightarrow\; x_j + w_j \le x_i, \\ u_{ij} = 1 \;\Rightarrow\; y_i + h_i \le y_j, & & o_{ij} = 1 \;\Rightarrow\; y_j + h_j \le y_i, \\ b_{ij} = 1 \;\Rightarrow\; z_i + d_i \le z_j, & & f_{ij} = 1 \;\Rightarrow\; z_j + d_j \le z_i. \end{array} \tag{3}$$

The problem may now be formulated as

$$\begin{array}{lll} \max & \sum_{i=1}^{n} p_i s_i & \\ \text{s.t.} & \ell_{ij} + r_{ij} + u_{ij} + o_{ij} + b_{ij} + f_{ij} \ge s_i + s_j - 1 & i,j = 1,\dots,n \\ & x_i - x_j + W\ell_{ij} \le W - w_i & i,j = 1,\dots,n \\ & x_j - x_i + W r_{ij} \le W - w_j & i,j = 1,\dots,n \\ & y_i - y_j + H u_{ij} \le H - h_i & i,j = 1,\dots,n \\ & y_j - y_i + H o_{ij} \le H - h_j & i,j = 1,\dots,n \\ & z_i - z_j + D b_{ij} \le D - d_i & i,j = 1,\dots,n \\ & z_j - z_i + D f_{ij} \le D - d_j & i,j = 1,\dots,n \\ & 0 \le x_i \le W - w_i & i = 1,\dots,n \\ & 0 \le y_i \le H - h_i & i = 1,\dots,n \\ & 0 \le z_i \le D - d_i & i = 1,\dots,n \\ & \ell_{ij}, r_{ij}, u_{ij}, o_{ij}, b_{ij}, f_{ij} \in \{0,1\} & i,j = 1,\dots,n \\ & s_i \in \{0,1\} & i = 1,\dots,n \\ & x_i, y_i, z_i \ge 0 & i = 1,\dots,n \end{array} \tag{4}$$

The first constraint ensures that if boxes i and j are packed, then they must be located left, right, under, over, behind or in-front of each other as stated in (2). The next six constraints are just linear versions of the constraints (3). The last three inequalities correspond to the constraints (1).

The IP-model has $6n^2 + n$ binary decision variables and $3n$ continuous variables. Although the size of $O(n^2)$ binary variables is not alarming, the problem is difficult to solve. This is mainly due to the use of conditional constraints (3), as these will loose their effect when solving the LP-relaxation, and thus bounds from LP-relaxation are in general far from the IP-optimal solution.

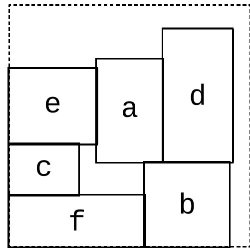


Figure 1: A packing represented by sequence $A =$ `<e,c,a,d,f,b>` and sequence $B =$ `<f,c,b,e,a,d>`.

3 Sequence Pairs

Murata et al. [9] presented an abstract representation of two-dimensional rectangle packings based on sequence pairs. The problem they consider is the minimum area enclosing rectangle packing problem. In the abstract representation every compact packing can be represented by two permutations of the numbers $\{1, 2, \ldots, n\}$ where each number represents a rectangle in the problem instance. The pair of permutations is called a *sequence pair* (A, B).

For a given packing, the two permutations A and B are found as follows: We use the terminology $\mathcal{A}_{ij}$ to denote that item i precedes j in sequence A. Then we have

$$(x_i + w_i \le x_j \quad \vee \quad y_j + h_j \le y_i) \quad \Leftrightarrow \quad \mathcal{A}_{ij} \tag{5}$$

In a similar way we use the terminology $\mathcal{B}_{ij}$ to denote that item i precedes j in sequence B, getting

$$(x_i + w_i \le x_j \quad \vee \quad y_i + h_i \le y_j) \quad \Leftrightarrow \quad \mathcal{B}_{ij} \tag{6}$$

Each of the two criteria (5) and (6) define a semi-ordering, and hence for a given packing the two permutations A and B can easily be found by repeatedly choosing one (of possibly more) minimum elements. Figure 1 illustrates a packing and a corresponding sequence pair (A, B).

From the definitions (5) and (6) we immediately see that if item i precedes item j in both sequences, then i must be placed left of j. If i succeeds j in sequence A but i precedes j in sequence B then i must be placed under j. Formally we have

$$\mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \quad \Rightarrow \quad i \text{ is left of } j \tag{7}$$

$$\neg\mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \quad \Rightarrow \quad i \text{ is under } j \tag{8}$$

where we use the terminology $\neg\mathcal{A}_{ij}$ to denote that $\mathcal{A}_{ji}$.

The relations (7) and (8) can be used to derive a pair of constraint graphs as illustrated in Figure 2. In both graphs the nodes correspond to the items. In the first graph we have an edge from i to j if and only if item i should be placed left of j $(\mathcal{A}_{ij} \wedge \mathcal{B}_{ij})$. In the second graph we have an edge from i to j if and only if item i should be placed under j $(\neg\mathcal{A}_{ij} \wedge \mathcal{B}_{ij})$. Traversing the nodes in topological order while assigning coordinates to the items, a packing (i.e. the coordinates of the items) can be obtained in $O(n^2)$ time. Tang et al. [18, 17] showed how the same packing can be derived without explicitly defining the constraint graph, but by finding weighted longest common subsequences in the sequence pair.

Pisinger [14] further improved the algorithm, by presenting an algorithm which transforms a sequence pair to a packing in time $O(n \log \log n)$ ensuring that the packing is *semi-normalized*. A *normalized* packing is a packing where the items are packed according to the sequence B and where each new item is placed such that it touches an already placed item on its left side, and an already placed item on its lower side. A *semi-normalized* packing is a packing where the items are packed according to the sequence B and where each new item is placed such that it touches the *contour* of the already placed items both from left and from below. The difference between an ordinary packing and the semi-normalized packing is illustrated in Figure 3.

The sequence pair representation makes it easy to construct a local search heuristic for packing problems. In each step a neighbor solution is generated by making a permutation of two items in one or both sequences A and B. The new sequence pair is transformed to a packing and the corresponding objective function is evaluated. Based on the local search framework chosen, the new solution can be accepted, or the algorithm tries a new neighbor solution to the previous solution.

4 Sequence pairs for two-dimensional knapsack packing

Let any sequence pair represent a legal solution to the 2DKP. To evaluate the solution we transform the sequence pair into a packing and add profit values for items located completely within the knapsack $W \times H$. This is illustrated in Figure 4.

To further speed up the algorithm, we stop the transformation from sequence pair to a packing as soon as the contour of already placed items is completely outside the knapsack. For large problems where only a limited amount of items fit in the knapsack, this saves a significant part of the computational time.

4.1 Simulated Annealing

To solve the 2DKP we use the metaheuristic Simulated Annealing which works well in cooperation with the sequence pair representation [9, 18, 17, 14].

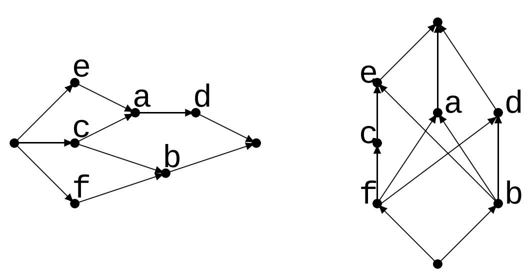


Figure 2: Constraint graphs corresponding to the sequence pair $(A,B) =$ (<e,c,a,d,f,b>,<f,c,b,e,a,d>). Redundant edges are removed for clarity. Edges indicate which rectangles should be placed left of each other (respectively under each other).

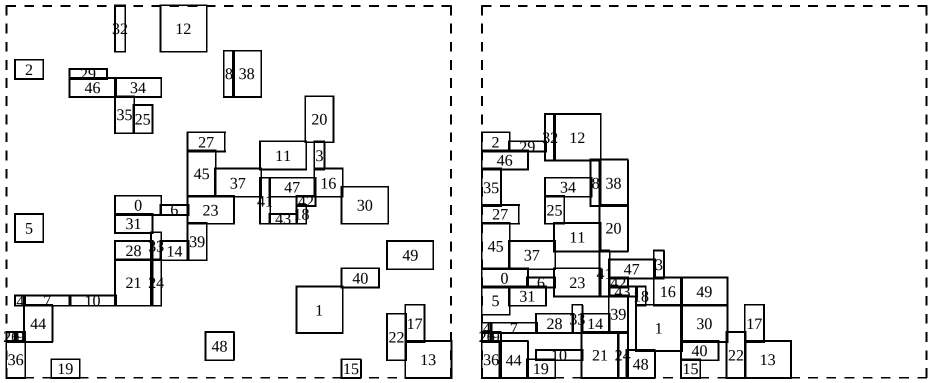


Figure 3: Transformation of a sequence pair to a packing using the ordinary transformation (left) and using the semi-normalized transformation (right)

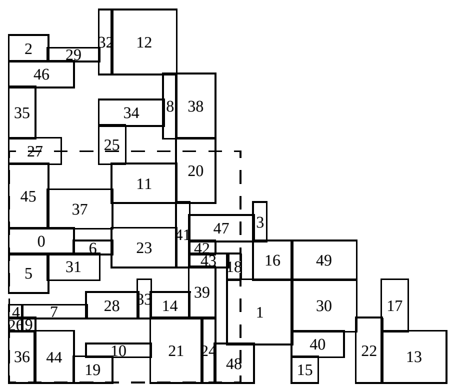


Figure 4: A sequence pair (A,B) has been transformed to a packing using the semi-normalized transformation. Only rectangles completely within the knapsack $W \times H$ (dashed line) contribute to the profit sum

In this setting we repeatedly make a small modification to the sequence pair, evaluate the profit of the corresponding packing, and accept the solution depending on the outcome. Simulated Annealing is used to determine whether a solution should be accepted. An outline of the algorithm is found in Figure 5.

Our variant of Simulated Annealing is as follows; At any given time the temperature is evaluated as $1/(t_0 + t_s \cdot a)$ where t_0 is a start time-value, t_s is a time-step value and a is the number of accepted solutions. The temperature depends on the time, so the higher t_0 + t_s is, the lower is the current temperature. The temperature is only decreased if a new solution is accepted, since this is the only situation where the time is incremented.

The neighborhood $N(s)$ of a solution $s = (A,B)$ is defined as one of the following three permutations: Either exchange two items in sequence A; exchange two items in sequence B; or exchange two items in both sequence A and B. The items are selected randomly.

```
choose initial solution s ∈ S
choose initial time t_0
choose time step t_s
a := 0
repeat
    choose s' ∈ N(s)
    if f(s') ≤ f(s) then accept := true else
        p := rand(0,1)
        T := 1/(t_0 + t_s·a)
        Δ := (f(s') − f(s))/f(s)
        if p < e^(−Δ/T) then accept := true
    end
    if accept then
        s := s'
        a := a+1
    end
until stop-criteria
return s
```

Figure 5: Simulated Annealing Heuristic

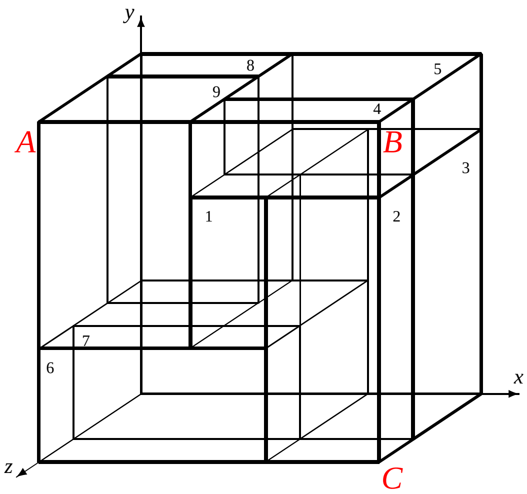
4.2 Rotations

Few papers consider exact algorithms for packing problems where rotation is allowed. A possible explanation could be the increased size of the solution space and the lack of high-quality upper bounds. In our heuristic, rotations are easy to handle as we may represent each packing by the triple (A,B,R). Here (A,B) is the sequence pair, and R is a binary vector, representing the rotations of 0 or 90 degrees. If rotation is allowed the neighborhood $N(s)$ of our heuristic is extended with a fourth permutation; Change the rotation flag of an item in R.

5 Three Dimensions

For the three-dimensional problem we will consider a new representation which like the sequence pair for two dimensions will contain relative box placement for three dimensions. We call the representation sequence tripple since it consists of three sequences. Not all three-dimensional packings are obtainable with this representation but we will prove that a large subset of all normalized packings may be represented. The same Simulated Annealing strategy we use for the sequence pair is applied to the sequence tripple to form a heuristic for 3DKP.

A *robot packing* is a packing which can be achieved by successively placing boxes starting from the bottom-left-behind corner, and such that each box is in-front of, right of, or over each of the previously placed boxes [11]. A *fully robot packable packing* is a packing which satisfies the robot packing criteria from any of the corners of the large bin.



i	w_i	h_i	d_i	x_i	y_i	z_i
1	2	4	9	4	3	0
2	3	7	3	6	0	6
3	3	7	6	6	0	0
4	5	2	3	4	7	6
5	5	2	6	4	7	0
6	6	3	3	0	0	6
7	6	3	6	0	0	0
8	4	6	3	0	3	0
9	4	6	6	0	3	3

Figure 6: A packing and the corresponding sequence tripple $(A,B,C) = \{<9,4,8,5,1,6,2,7,3>, <4,2,5,3,1,9,6,8,7>, <2,3,6,7,1,4,5,9,8>\}$. The letters A,B,C on the figure indicate the directions which are used for defining the corresponding sequence of boxes

Robot packings are motivated by several industrial applications, where boxes have to be packed by robots equipped with a rectangular "hand" parallel to the base of the large box. To avoid collisions between the hand and the boxes, it is demanded that no already packed box block for the movement of the "hand". In [11] it is shown that the quality of a packing is seldom affected by restricting the solution space to the set of robot packings.

5.1 Sequence Tripple

A given fully robot packable packing is represented by three sequences A, B and C where each sequence is a permutation of the n boxes. For any sequence X we say $\mathcal{X}_{ij}$ if and only if i is before j in sequence X and define $\neg\mathcal{X}_{ij} \Leftrightarrow \mathcal{X}_{ji}$.

In a similar way as in Section 3 we define sequence A by the criteria

$$(x_i + w_i \leq x_j \quad \vee \quad y_i \geq y_j + h_j \quad \vee \quad z_i \geq z_j + d_j) \quad \Leftrightarrow \quad \mathcal{A}_{ij} \tag{9}$$

In other words $\mathcal{A}_{ij}$ iff i is located left, over or in-front of j. Using the formulation (2) we have $\mathcal{A}_{ij} \Leftrightarrow \ell_{ij} + o_{ij} + f_{ij} \geq 1$.

Sequence B is defined by

$$(x_i \geq x_j + w_j \quad \vee \quad y_i \geq y_j + h_j \quad \vee \quad z_i \geq z_j + d_j) \quad \Leftrightarrow \quad \mathcal{B}_{ij} \tag{10}$$

This means $\mathcal{B}_{ij}$ iff i is located right, over or in-front of j. The relation can be expressed as $\mathcal{B}_{ij} \Leftrightarrow r_{ij} + o_{ij} + f_{ij} \geq 1$.

Finally, sequence C is defined by

$$(x_i \geq x_j + w_j \quad \vee \quad y_i + h_i \leq y_j \quad \vee \quad z_i \geq z_j + d_j) \quad \Leftrightarrow \quad \mathcal{C}_{ij} \tag{11}$$

In words, $\mathcal{C}_{ij}$ iff i is located right, under or in-front of j, which can be expressed as $\mathcal{C}_{ij} \Leftrightarrow r_{ij} + u_{ij} + f_{ij} \geq 1$.

Due to the definition of fully robot packable packings, there will always be an item which is located furthest left-over-behind. By removing this item and repeating the operation, we get the ordering of sequence A. In a similar way the orderings of B and C can be determined, as illustrated in Figure 6. This shows that every fully robot packable packing can be represented by a sequence tripple.

Using the relations

$$\begin{array}{ll} \mathcal{A}_{ij} \Leftrightarrow \ell_{ij} + o_{ij} + f_{ij} \geq 1, & \ell_{ij} + r_{ij} \leq 1, \\ \mathcal{B}_{ij} \Leftrightarrow r_{ij} + o_{ij} + f_{ij} \geq 1, & o_{ij} + u_{ij} \leq 1, \\ \mathcal{C}_{ij} \Leftrightarrow r_{ij} + u_{ij} + f_{ij} \geq 1, & f_{ij} + b_{ij} \leq 1, \end{array} \tag{12}$$

we find that

$$\begin{array}{rcl} \mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \wedge \mathcal{C}_{ij} & \Leftrightarrow & f_{ij} = 1 \\ \mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij} & \Leftrightarrow & \ell_{ij} + r_{ij} \geq 1 \vee o_{ij} + u_{ij} \geq 1 \vee f_{ij} + b_{ij} \geq 1 \\ \neg\mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \wedge \mathcal{C}_{ij} & \Leftrightarrow & r_{ij} = 1 \\ \neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij} & \Leftrightarrow & u_{ij} = 1 \\ \mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij} & \Leftrightarrow & o_{ij} = 1 \\ \mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij} & \Leftrightarrow & \ell_{ij} = 1 \\ \neg\mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij} & \Leftrightarrow & \ell_{ij} + r_{ij} \geq 1 \vee o_{ij} + u_{ij} \geq 1 \vee f_{ij} + b_{ij} \geq 1 \\ \neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{i} \wedge \neg\mathcal{C}_{ij} & \Leftrightarrow & b_{ij} = 1 \end{array} \tag{13}$$

Notice that $\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij}$ and $\neg\mathcal{A}_{ij} \wedge \mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij}$ cannot occur for any packing. We have, however, chosen to assign these cases a meaning, such that every sequence tripple has a corresponding packing. This leads to the following four criteria, similar to (7) and (8), which are used to determine the relative box positions:

$$\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij} \Rightarrow i \text{ is left of } j \tag{14}$$

$$\neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij} \Rightarrow i \text{ is under } j \tag{15}$$

$$\neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij} \Rightarrow i \text{ is behind } j \tag{16}$$

$$\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij} \Rightarrow i \text{ is behind } j \tag{17}$$

Notice that both (16) and (17) impose that i must be behind j in the packing. The unfortunate consequence of this is that the representation is biased towards orderings in that direction which could have a negative impact on the solution process, but as we wish to let every sequence tripple represent a packing, an arbitrary choice had to be done.

5.2 A Placement Algorithm

To find a placement (i.e. the coordinates of the boxes) corresponding to a sequence tripple, we can construct three constraint graphs similar to Figure 2: In the first graph we have an edge from item i to item j if i is located left of j (i.e. $\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij}$). In the second graph we have an edge from item i to item j if i is located under j (i.e. $\neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij}$). In the last graph we have an edge from item i to item j if i is located behind j (i.e. $\neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij}$ or $\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij}$). Traversing the nodes in topological order for each graph while assigning coordinates to the items, we find the location of all boxes in time $O(n^2)$.

By observing that $\neg\mathcal{B}_{ij}$ is a necessary criteria for node i to precede node j in each of the three constraint graphs, we may actually omit the topological ordering as it is in each case given by the reverse order of sequence B.

The last box in B is placed at $(x,y,z) = (0,0,0)$ and succeeding boxes are placed one by one according to the reverse order of sequence B. At any time let P consist of all previously placed boxes. Now assume we wish to place box i. To determine the position of i we compare i with every box $j \in P$. Let $P_x \subseteq P$ be the subset of boxes which satisfy (14), i.e. $\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij}$, let $P_y \subseteq P$ be the subset which satisfy (15), i.e. $\neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij}$, and let $P_z \subseteq P$ be the subset which satisfy (16) or (17), i.e. $\neg\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \neg\mathcal{C}_{ij}$ or $\mathcal{A}_{ij} \wedge \neg\mathcal{B}_{ij} \wedge \mathcal{C}_{ij}$). Now assign i coordinates (x_i, y_i, z_i) determined by

$$x_i = \max(0, \max_{j \in P_x}(x_j + w_j)) \tag{18}$$

$$y_i = \max(0, \max_{j \in P_y}(y_j + h_j)) \tag{19}$$

$$z_i = \max(0, \max_{j \in P_z}(z_j + d_j)) \tag{20}$$

Once a box has been placed it is inserted into P.

If we maintain a table in which the position of each box i in the three sequences A, B, C is saved, we can test whether $\mathcal{A}_{ij}$, $\mathcal{B}_{ij}$ or $\mathcal{C}_{ij}$ holds in constant time for two given boxes i, j. Since placing a box only requires comparison with every previously placed box, calculating (18) to (20) for a given box i can be done in $O(|P|) = O(n)$ time. Placing all n boxes then requires $O(n^2)$ time.

To speed up the placement procedure slightly we remove a box from P if it is completely "shaded" by a newly inserted box. A box j is shaded by a box i if $x_j + w_j \leq x_i + w_i$, $y_j + h_j \leq y_i + h_i$ and $z_j \leq d_j < z_i + d_i$.

5.3 Simulated Annealing

To solve 3DKP we use Simulated Annealing similarly as for two dimensions but with the three-dimensional sequence representation. The neighborhood is increased to accommodate the extra sequence and consists of the following permutations: 1) exchange two boxes from one of the sequences, 2) exchange two boxes in sequence A and B, 3) exchange two boxes in sequence A and C, 4) exchange two boxes in sequence B and C, 5) exchange two boxes in all sequences.

6 Computational Experiments

The heuristic described in the previous sections was implemented in C++ using the sequence pair algorithm by Pisinger [14] for two dimensions and an implementation of the placement algorithm for sequence tripple described in section 5.2 for three dimensions. The implementation was tested on a computer with an AMD Athlon 64 3800+ (2.4 GHz) processor with 2 GB ram using the GNU-C++ compiler (gcc 4.0). This section is divided into two parts; one about 2DKP and one about 3DKP.

6.1 2D Computational Experiments

To test the 2DKP heuristic we used both the classical instances and a new set of instances. The instances were used for parameter tuning of the heuristic. Results are reported for instances both without and with rotation allowed.

6.1.1 Classical Instances for 2DKP

We use the benchmarks instances considered by Fekete et al. [6] and Caprara and Monaci. The instances are listed in Table I. The instances `beasley1-7` originates from [1]. The `cgcut` and `gcut` instances are guillotine-cut instances from the OR library. The instance `wang20` is from [19] and also a guillotine-cut instance. The instances `3` to `CHL5` are also guillotine-cut instances by Hifi [8]. Finally, the instance `okp1-5` are by Fekete and Schepers [5].

For each instance we determine two values n_0 and n_1. The first value, n_0, is defined as $n_0 = n[\text{Knapsack Area}]/[\text{Total area}]$. This value should give a hint as to how many items the knapsack will contain on average in solutions. The value n_1 is the number of items chosen in the optimal solution for the one dimensional relaxation, as described in the sequel.

We also use the value n_0 to determine the running time of our experiments: For an instance with n rectangles and n_0 defined as above let $F(n,n_0) = n_0 \lg n$. The idea of this function is that if we expect there to be n_0 items in the knapsack then there are roughly n_0^n different possible solutions to search, therefore $F(n,n_0)$ should give us a rough indication of the size of solution space.

The running time of each instance is determined from the value $F(n,n_0)$ of the instance, by considering the interval that $F(n,n_0)$ belongs to. Different intervals and their running times are shown in Table II. Thus the minimum and maximum running times are 30 and 600 seconds respectively.

Let the *one-dimensional relaxation* of the two-dimensional packing problem be a one-dimensional knapsack problem where the knapsack and the items have size equal to their area, and items have profit equal to the profit of the rectangles. For all instances we consider here this problem is solved to optimum within 5 seconds using the exact method by Pisinger [12]. The value n_1 is the number of items chosen in the optimal solution for the one dimensional relaxation and we use this value to set the Simulated Annealing parameters for the instance (see section 6.2.1), and n_1 should indicate the number of rectangles to be expected in an optimal solution.

6.2 New Instances for 2DKP

For 2DKP we have created 80 new instances. The rectangle dimensions in each instance belongs to one of five different classes which are listed in Table III. The five classes are `tall` (T),

Instance	n	n_0	n_1	Instance	n	n_0	n_1
beasley1	10	5.3	5	gcut10	20	3.7	5
beasley2	17	6.3	8	gcut11	30	4.6	6
beasley3	21	7.6	6	gcut12*	50	4.0	4
beasley4	7	6.5	6	gcut13*	32	20.1	18
beasley5	14	6.0	7	wang20*	42	5.0	4
beasley6	15	7.8	8	3	62	3.9	11
beasley7	8	18.3	8	3s	62	3.9	6
beasley8	13	8.2	9	a1	62	4.2	11
beasley9	18	7.4	9	a1s	62	4.2	7
beasley10	13	6.8	7	a2	53	5.5	11
beasley11	15	9.1	8	a2s	53	5.5	7
beasley12	22	8.6	11	chl2	19	9.1	10
cgcut1*	16	10.7	8	chl2s	19	9.1	9
cgcut2*	23	14.8	11	chl3	35	89.8	35
cgcut3*	62	3.90	11	chl3s	35	89.8	35
gcut1*	10	3.82	4	chl4	27	92.7	27
gcut2	20	4.6	5	chl4s	27	92.7	27
gcut3*	30	4.6	6	chl5	18	7.4	5
gcut4	50	4.3	6	okp1*	50	14.3	9
gcut5	10	4.6	4	okp2	30	9.6	11
gcut6	20	4.1	5	okp3*	30	8.3	11
gcut7	30	3.7	5	okp4	61	10.1	8
gcut8	50	4.5	5	okp5*	97	12.6	15
gcut9	10	4.9	5				

Table I: Literature instances for 2DKP. Instances marked with '*' are used for fine-tuning of the heuristic.

For $F(n,n_0) \in$	$[0;25)$	$[25;65)$	$[65;100)$	$[100;250)$	$[250;\infty)$
Set $T(n,n_0)$	30	60	120	240	600

Table II: The running $T(n,n_0)$ in seconds determined from $F(n,n_0)$.

Class	Description	Width	Height
T	**Tall.** Rectangles are tall	$[1, \frac{1}{3} \cdot 100]$	$[\frac{2}{3} \cdot 100, 100]$
W	**Wide.** Rectangles are wide	$[\frac{2}{3} \cdot 100, 100]$	$[1, \frac{1}{3} \cdot 100]$
S	**Square.** Rectangles are square	$[1, 100]$	Equal to width
U	**Uniform.** Largest dimension is no more than 150% of the smallest	$[\frac{2}{3} \cdot 100, 100]$	$[\frac{2}{3} \cdot 100, 100]$
D	**Diverse.** Largest dimension can be up-to 100 times the smallest	$[1, 100]$	$[1, 100]$

Table III: The 5 different classes of new EP instances. The width and height of the rectangles in each class are selected randomly from the intervals in the 'Width' and 'Height' column.

`wide` (W), `square` (S), `uniform` (U) and `diverse` (D). The number of rectangles, *n*, in each instance is selected from the set $\{30, 50, 100, 200\}$. The rectangles may be `clustered` (C) and `random` (R). `Clustered` instances consists of only 20 rectangles which are duplicated appropriately, while in the `random` instances all rectangles are independently generated. Finally the area of the bin is either 25 % or 75 % of the total area of the rectangles and the height of the bin is always two times the width. The naming convention is EP-n-c-t-p, where $n \in \{30, 50, 100, 200\}$ is the number of rectangles, $c \in (\mathrm{T}, \mathrm{W}, \mathrm{S}, \mathrm{U}, \mathrm{D})$ describes the class, $t \in (\mathrm{C}, \mathrm{R})$ describes if it is clustered or random, $p \in 25, 75$ describes the size of the bin in percentage of the total rectangle area. The profit of the rectangles is always the area of the rectangle + 20 units. The instances are presented in Table IV and are available along with the source code to generate them at this web-address: `http://www.diku.dk/~pisinger/codes.html`.

6.2.1 Parameter tuning

As seen in Figure 5, two values are crucial for the results of Simulated Annealing; The start time t_0 and the time step t_s. To determine appropriate values of t_0 and t_s we experimented with the 22 instances marked with '*' in Table I and IV. These contain between 16 and 200 rectangles. We performed the experiments with $t_0 \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}\}$ and $t_s \in \{10^{2}, 10^{1}, 10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}, 10^{-11}, 10^{-13}\}$ For the 22 instances each of the 81 combinations were tested using the running times from Table II. Results from four selected instances are presented in Figure 7.

Based on the results of the parameter tuning for the 22 instances we were able to establish that good values of t_0 and t_s are:

$$t_0 = n_1^2, \qquad t_s = \frac{n_1^2}{10^7}.$$

The values can be interpreted in the following way: The higher t_0 and t_s the less likely is acceptance of a non-improving permutation. The larger the number of rectangles is in an optimal solution the more improving steps must be done before the heuristic should escape local minima by accepting a non-improving change.

Instance	n	n_0	n_1	Instance	n	n_0	n_1
ep-30-D-C-25	30	8.3	18	ep-100-D-C-25	100	24.9	58
ep-30-D-C-75	30	22.5	26	ep-100-D-C-75	100	74.5	92
ep-30-D-R-25	30	7.7	17	ep-100-D-R-25	100	24.6	60
ep-30-D-R-75	30	22.5	26	ep-100-D-R-75	100	74.1	91
ep-30-S-C-25	30	7.4	16	ep-100-S-C-25	100	24.9	58
ep-30-S-C-75	30	22.5	26	ep-100-S-C-75	100	75.7	89
ep-30-S-R-25	30	7.4	17	ep-100-S-R-25	100	24.9	60
ep-30-S-R-75	30	22.4	26	ep-100-S-R-75	100	74.9	89
ep-30-T-C-25	30	7.4	13	ep-100-T-C-25	100	25.0	44
ep-30-T-C-75	30	22.3	25	ep-100-T-C-75	100	74.9	84
ep-30-T-R-25	30	7.4	13	ep-100-T-R-25	100	24.9	47
ep-30-T-R-75	30	22.4	25	ep-100-T-R-75	100	74.8	85
ep-30-U-C-25	30	7.5	9	ep-100-U-C-25	100	25.0	30
ep-30-U-C-75	30	22.5	23	ep-100-U-C-75	100	75.0	79
ep-30-U-R-25	30	7.5	9	ep-100-U-R-25	100	24.9	31
ep-30-U-R-75	30	22.4	23	ep-100-U-R-75	100	74.8	80
ep-30-W-C-25	30	10.7	17	ep-100-W-C-25	100	24.7	45
ep-30-W-C-75	30	22.3	25	ep-100-W-C-75	100	74.5	86
ep-30-W-R-25	30	11.3	17	ep-100-W-R-25	100	24.8	50
ep-30-W-R-75	30	22.5	26	ep-100-W-R-75	100	74.8	87
ep-50-D-C-25	50	12.2	28	ep-200-D-C-25	200	50.0	117
ep-50-D-C-75	50	37.2	45	ep-200-D-C-75	200	149.8	183
ep-50-D-R-25	50	12.2	27	ep-200-D-R-25	200	49.5	119
ep-50-D-R-75	50	37.2	45	ep-200-D-R-75	200	149.8	182
ep-50-S-C-25	50	12.4	28	ep-200-S-C-25	200	50.0	118
ep-50-S-C-75	50	37.5	44	ep-200-S-C-75	200	149.7	179
ep-50-S-R-25	50	12.5	29	ep-200-S-R-25	200	49.9	116
ep-50-S-R-75	50	37.4	44	ep-200-S-R-75	200	149.9	177
ep-50-T-C-25	50	12.5	22	ep-200-T-C-25	200	49.9	89
ep-50-T-C-75	50	37.3	42	ep-200-T-C-75	200	149.7	170
ep-50-T-R-25	50	12.5	22	ep-200-T-R-25	200	49.9	97
ep-50-T-R-75	50	37.3	42	ep-200-T-R-75	200	149.8	172
ep-50-U-C-25	50	12.4	15	ep-200-U-C-25	200	49.9	60
ep-50-U-C-75	50	37.5	39	ep-200-U-C-75	200	149.7	159
ep-50-U-R-25	50	12.5	15	ep-200-U-R-25	200	49.9	63
ep-50-U-R-75	50	37.5	40	ep-200-U-R-75	200	149.8	160
ep-50-W-C-25	50	14.1	25	ep-200-W-C-25	200	49.9	91
ep-50-W-C-75	50	37.3	43	ep-200-W-C-75	200	149.7	174
ep-50-W-R-25	50	13.9	25	ep-200-W-R-25	200	49.9	102
ep-50-W-R-75	50	37.4	43	ep-200-W-R-75	200	149.5	175

Table IV: New instances for 2DKP.

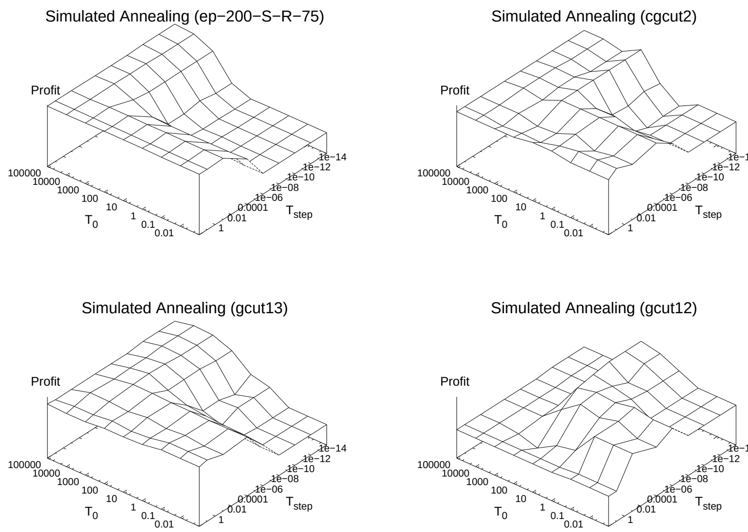


Figure 7: Results of the Simulated Annealing heuristic for different values of t_0 and t_s on four different instances.

6.2.2 Results

Based on the parameter tuning from the previous section our heuristic was applied to the benchmark instances described in Section 6.1.1 and 6.2. To determine the robustness of the heuristic we ran each benchmark instance with 10 different random seeds. We have reported the best, worst and average solution value in Table V along with the running time of each instance for each seed. The results on the 80 newly proposed benchmarks are listed in Table VI and VII. The heuristic finds the optimal value in all but 4 of the classical instances and on the new instances the results are generally higher than 95% of the value of the one-dimensional relaxation, which demonstrates its ability to find good solutions for both small and large instances.

6.2.3 Rotations

We repeated all tests allowing rotation, however we doubled the running time to accommodate for the larger solution space. A maximum time limit of 600 seconds was still assigned to all instances. Parameter-tuning revealed that the same settings as reported in section 6.2.1 also give good results when rotation is allowed. The results on the two sets of instances are reported in Table VIII, Table IX and Table X.

For the classical benchmark instances the results with rotation are always better or as good as the results without rotation. Interestingly enough the results with rotation are generally larger than 95% of the optimal solution for the one-dimensional relaxed problem, and often they are close to 98%.

For the new test instances we got slightly worse results when rotation is allowed in 32 out of the 80 cases. This is mainly due to the increased solution space. In only two of the instances are

			Egeblad and Pisinger					Exact Methods Time	
Instance	1D	Optimal	Best	Avg	Worst	Best Time	Seed Time	Fek-Sch	Cap-Mon
beasley1	201	164	164	164	164	$\leq$ 0.02	30	$\leq$ 0.02	-
beasley2	253	230	230	230	230	$\leq$ 0.02	60	$\leq$ 0.02	-
beasley3	266	247	247	247	247	0.02	60	$\leq$ 0.02	-
beasley4	275	268	268	268	268	$\leq$ 0.02	30	$\leq$ 0.02	-
beasley5	373	358	358	358	358	$\leq$ 0.02	30	$\leq$ 0.02	-
beasley6	317	289	289	289	289	0.07	60	$\leq$ 0.02	-
beasley7	430	430	430	430	430	$\leq$ 0.02	30	$\leq$ 0.02	-
beasley8	938	834	834	834	834	$\leq$ 0.02	60	$\leq$ 0.02	-
beasley9	962	924	924	924	924	0.41	60	$\leq$ 0.02	-
beasley10	1517	1452	1452	1452	1452	$\leq$ 0.02	60	$\leq$ 0.02	-
beasley11	1864	1688	1688	1688	1688	0.04	60	$\leq$ 0.02	-
beasley12	2012	1865	1865	1865	1865	0.5	60	$\leq$ 0.02	-
cgcut1	260	244	244	244	244	$\leq$ 0.02	60	1.46	0.3
cgcut2	2919	2892	2892	2892	2892	1.8	120	531.93	531.93
cgcut3	2020	1860	1860	1842	1840	28.24	30	4.58	4.58
gcut1	62488	48368	48368	48368	48368	$\leq$ 0.02	30	0.01	0
gcut2	62500	59798	59798	59680.5	59563	23.27	30	0.22	0.19
gcut3	62500	61275	61275	61152.6	60663	3.32	30	3.24	2.16
gcut4	62500	61380	61380	61380	61380	1.68	30	376.52	346.99
gcut5	249854	195582	195582	195582	195582	0.03	30	0.5	0
gcut6	249992	236305	236305	236305	236305	0.02	30	0.12	0.06
gcut7	249998	240143	240143	240143	240143	0.32	30	1.07	0.22
gcut8	250000	245758	245758	245758	245758	0.07	60	168.5	136.71
gcut9	997256	939600	939600	939600	939600	0.01	30	0.08	0
gcut10	999918	937349	937349	937349	937349	0.89	30	0.14	0
gcut11	1000000	969709	969709	968582.3	958442	0.19	30	16.3	14.76
gcut12	1000000	979521	979521	977670.2	976877	7.04	30	25.39	16.85
gcut13	9000000	$\geq$8408316 $\geq$8622498 $\leq$9000000	8669457	8629142.7	8613889	85.44	240	 1800	1800
wang20	2800	2726	2716	2712.5	2711	48.41	60	2.72	2.72
3	2020	1860	1860	1842	1840	28.22	30	$\leq$ 0.02	-
3s	2800	2726	2726	2722	2721	17.49	30	$\leq$ 0.02	-
a1	2140	2020	1980	1968	1960	2.31	30	$\leq$ 0.02	-
a1s	3000	2956	2950	2950	2950	0.16	30	$\leq$ 0.02	-
a2	2705	2615	2615	2566	2545	7.53	30	$\leq$ 0.02	-
a2s	3600	3535	3535	3517.9	3516	14.16	30	$\leq$ 0.02	-
chl2	2502	2326	2326	2326	2326	6.78	60	$\leq$ 0.02	-
chl2s	3410	3336	3336	3334.7	3323	0.17	60	$\leq$ 0.02	-
chl3	5283	5283	5283	5283	5283	$\leq$ 0.02	240	$\leq$ 0.02	-
chl3s	7402	7402	7402	7402	7402	$\leq$ 0.02	240	$\leq$ 0.02	-
chl4	8998	8998	8998	8998	8998	$\leq$ 0.02	240	$\leq$ 0.02	-
chl4s	13932	13932	13932	13932	13932	$\leq$ 0.02	240	$\leq$ 0.02	-
chl5	600	589	589	586.5	584	2.4	60	$\leq$ 0.02	-
okp1	27718	27718	27718	27542.7	27486	11.83	120	35.84	11.6
okp2	22502	22502	22214	22098.6	21947	36.41	60	1559	1535.95
okp3	24019	24019	24019	23804.6	23531	17.4	60	10.63	1.91
okp4	32893	32893	32893	32893	32893	8.89	60	4.05	2.13
okp5	27923	27923	27923	26753	25456	12.22	120	488.27	488.27

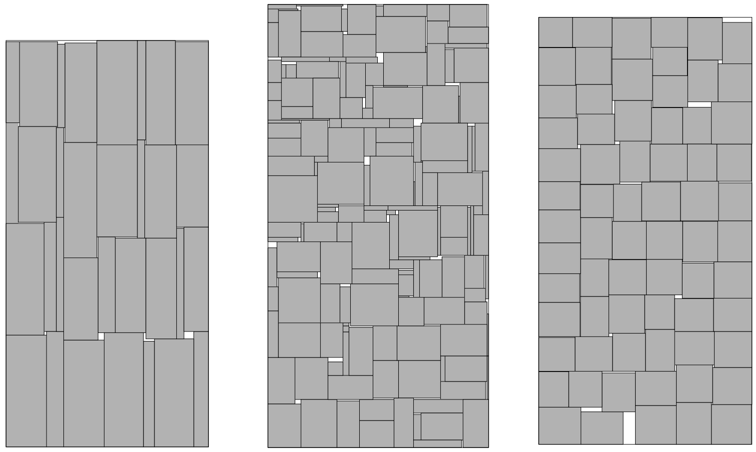
Table V: Results for the classical benchmark instances. '1D' is the result of the one-dimensional relaxed problem. 'Optimal' is the value of the optimal solution. The columns under 'Egeblad and Pisinger' are results of this heuristic. 'Avg', 'Best' and 'Worst' columns are the average, best and worst results on each instance for the 10 seeds. 'Best Time' is the time before the heuristic discovered the best solution. 'Seed Time' is the given to each of the 10 seeds. Thus the total running time on each instance is 10 times 'Seed Time'. The 'Exact Methods Time' represent the running time of the other methods. All running times are in seconds. For gcut13 no optimal value is currently known, but we have reported the results of respectively Caprara and Monaci and Fekete and Schepers along with their upper-bound.

Instance	1D	Best	Avg	Worst	Best Time	Seed Time	1D Percentage
ep-30-D-C-25	2155	2062	2025.8	1953	21.38	60	95.7
ep-30-D-C-75	5135	5061	5008.2	4953	66.14	240	98.6
ep-30-D-R-25	2364	2244	2224	2207	45.74	60	94.9
ep-30-D-R-75	6191	6080	6055.9	5995	33.54	240	98.2
ep-30-S-C-25	22494	21858	21222.8	20692	25.99	60	97.2
ep-30-S-C-75	66935	65675	65313	64765	15.27	240	98.1
ep-30-S-R-25	21674	20752	20287.4	20067	3.53	60	95.7
ep-30-S-R-75	65319	64496	64106.5	62743	129.56	240	98.7
ep-30-T-C-25	10938	9262	9262	9262	0.29	60	84.7
ep-30-T-C-75	32377	32097	32034.8	31827	124.19	240	99.1
ep-30-T-R-25	10647	9919	9919	9919	0.45	60	93.2
ep-30-T-R-75	31750	31461	31426.3	31358	172.63	240	99.1
ep-30-U-C-25	52002	49395	49395	49395	13.8	60	95.0
ep-30-U-C-75	155306	145613	145613	145613	1.95	240	93.8
ep-30-U-R-25	50246	50029	50029	50029	17.85	60	99.6
ep-30-U-R-75	151710	147159	144782.1	144518	26.14	240	97.0
ep-30-W-C-25	14060	13130	13121.3	13104	34.88	60	93.4
ep-30-W-C-75	29780	21598	21593.2	21582	29.19	240	72.5
ep-30-W-R-25	14840	14235	14235	14235	2.96	60	95.9
ep-30-W-R-75	33396	23860	23860	23860	15.56	240	71.4
ep-50-D-C-25	3232	3107	2971.3	2874	119.45	120	96.1
ep-50-D-C-75	8849	8673	8605.4	8533	12.05	240	98.0
ep-50-D-R-25	3546	3361	3308.4	3227	91.19	120	94.8
ep-50-D-R-75	9678	9463	9426	9401	105.08	240	97.8
ep-50-S-C-25	37185	36107	35887.7	35468	69.99	120	97.1
ep-50-S-C-75	111329	109511	108574	107633	78.51	240	98.4
ep-50-S-R-25	35599	34864	34527.5	33841	12.72	120	97.9
ep-50-S-R-75	106699	104872	104191.5	103482	210.65	240	98.3
ep-50-T-C-25	17757	17324	17270	17231	111.73	120	97.6
ep-50-T-C-75	52863	50907	50065.9	49430	93.18	240	96.3
ep-50-T-R-25	18643	18264	18213.7	18152	40.51	120	98.0
ep-50-T-R-75	55475	54922	54712.1	54261	106.91	240	99.0
ep-50-U-C-25	85978	80416	79727.7	77030	5.8	120	93.5
ep-50-U-C-75	257446	248564	247737.5	242582	239.47	240	96.5
ep-50-U-R-25	83736	78474	78225	77346	67.79	120	93.7
ep-50-U-R-75	251412	245628	241946.3	240375	117.35	240	97.7
ep-50-W-C-25	18082	17149	17129	17103	81.64	120	94.8
ep-50-W-C-75	48909	46170	45617.8	45024	35.59	240	94.4
ep-50-W-R-25	19215	18449	18449	18449	8.33	120	96.0
ep-50-W-R-75	55475	54708	54601.8	54431	175.06	240	98.6

Table VI: Results for the small ep instances. ’1D’ is the optimal solution of the one-dimensional relaxed problem. ’Best’, ’Avg’ and ’Worst’ columns are the best, average and worst results for each instance on the 10 seeds. ’Best Time’ is the time it took before the best solution was encountered. ’Seed Time’ is the time spend on each seed and the total time for each instance is 10 times ’Seed Time’. ’1D Percentage’ is the percentage deviation between the heuristic solution and the one-dimensional relaxed problem upper-bound.

Instance	1D	Best	Avg	Worst	Best Time	Seed Time	1D Percentage
ep-100-D-C-25	6740	6316	6150.1	5856	108.62	240	93.7
ep-100-D-C-75	18402	18003	17910	17769	454.68	600	97.8
ep-100-D-R-25	8201	7804	7713.3	7642	59.34	240	95.2
ep-100-D-R-75	23121	22635	22536.8	22465	515.47	600	98.0
ep-100-S-C-25	73640	72154	71389.8	69990	41.67	240	98.0
ep-100-S-C-75	219930	215294	214085	213022	554.55	600	98.0
ep-100-S-R-25	89670	88334	86831.6	85952	209.43	240	98.5
ep-100-S-R-75	267156	263237	261317.3	259433	574.8	600	98.5
ep-100-T-C-25	34780	34123	33978.3	33884	160.24	240	98.1
ep-100-T-C-75	102983	100645	100343.1	100078	174.21	600	97.7
ep-100-T-R-25	35553	34948	34818.1	34617	193.51	240	98.3
ep-100-T-R-75	105728	103888	102903.1	102539	471.5	600	98.3
ep-100-U-C-25	170233	165313	163982.6	160482	158.73	240	97.1
ep-100-U-C-75	511610	495431	495002.2	492289	281.83	600	96.8
ep-100-U-R-25	171128	168779	167433	166087	146.84	240	98.6
ep-100-U-R-75	512135	503471	498286	491477	596.37	600	98.3
ep-100-W-C-25	31972	24325	24325	24325	5.33	240	76.1
ep-100-W-C-75	95052	79803	78353.1	76448	430.71	600	84.0
ep-100-W-R-25	38595	29593	29507.1	29145	188.5	240	76.7
ep-100-W-R-75	115235	109102	107764.8	107113	584.25	600	94.7
ep-200-D-C-25	13541	12415	12150.5	11783	493.41	600	91.7
ep-200-D-C-75	36607	35489	35273.7	35029	514.43	600	96.9
ep-200-D-R-25	16770	15919	15780.6	15682	461.28	600	94.9
ep-200-D-R-75	46985	45867	45671.8	45537	482.68	600	97.6
ep-200-S-C-25	147526	143875	142676.6	141416	242.86	600	97.5
ep-200-S-C-75	440265	429097	426756.9	424657	511.84	600	97.5
ep-200-S-R-25	172233	169314	168298.6	167148	283.44	600	98.3
ep-200-S-R-75	514175	504328	501084.3	497746	586.69	600	98.1
ep-200-T-C-25	69694	68208	67963	67686	555.5	600	97.9
ep-200-T-C-75	206059	199431	197752.1	196407	590.09	600	96.8
ep-200-T-R-25	65706	64335	64195	63854	576.58	600	97.9
ep-200-T-R-75	194811	192034	190747.8	189436	585.89	600	98.6
ep-200-U-C-25	340668	337958	335537.5	328940	365.07	600	99.2
ep-200-U-C-75	1022712	991048	982796.8	971152	574.16	600	96.9
ep-200-U-R-25	337830	334623	329949.7	325238	248.8	600	99.1
ep-200-U-R-75	1014222	973279	970129.5	967034	412.68	600	96.0
ep-200-W-C-25	64342	62657	62456.4	62048	583.09	600	97.4
ep-200-W-C-75	189835	175161	173025.6	170785	538.25	600	92.3
ep-200-W-R-25	75840	74331	74202	74042	528.19	600	98.0
ep-200-W-R-75	225571	217963	215865.9	211880	521.45	600	96.6

Table VII: Results for the large ep instances. See table VI for a description of the columns.



ep-50-T-R-75.eps ep-100-D-R-75.eps ep-200-U-R-75.eps

Figure 8: Best results for three two-dimensional instances without rotation.

the results with rotation more than 1% better than without. It is also important to note that just as for the smaller benchmark instances the results are generally better than 95% of the optimal solution to the one-dimensional relaxed problem, which means that the heuristic perform well even for large instances.

6.2.4 The Instance gcut13

Since gcut13 is the only instance of the classical benchmark instances from the literature where the optimal solution is unknown, we decided to investigate this instance further without considering rotations. We used 144 seeds with 10 minutes running time on each seed, given a total of 24 hours for this instance. The parameters for the Simulated Annealing was based on the results we gathered during parameter-tuning for gcut13, and were set $t_0 = 0.1$ and $t_s = 10$. The best result was reached after 367 seconds for one of the seeds and was 8728123 which is somewhat better than the 8622498 we were able to reach with 10 seeds and 4 minutes running time. This demonstrates that the heuristic is able to find better solution given more running time. The resulting placement can be seen in Figure 9.

6.3 3D Computational Experiments

A set of experiments where conducted for the three-dimensional variant similar to the two-dimensional problems. However, since we were unable to find any common benchmark instances in the literature these experiments were conducted only on new instances.

Instance	1D	No rotation	Best	Avg	Worst	Best Time	Seed Time	Deviation
beasley1	201	164	193	193	193	0.01	60	96.02
beasley2	253	230	250	250	250	0.25	120	98.81
beasley3	266	247	259	259	259	0.35	120	97.37
beasley4	275	268	268	268	268	0	60	97.45
beasley5	373	358	370	370	370	0	60	99.20
beasley6	317	289	300	298	298	23.92	120	94.6
beasley7	430	430	430	430	430	0	60	100.0
beasley8	938	834	886	886	886	0.19	120	94.46
beasley9	962	924	924	921.4	918	4.38	120	96.05
beasley10	1517	1452	1452	1452	1452	0	120	95.72
beasley11	1864	1688	1786	1786	1786	0.01	120	95.82
beasley12	2012	1865	1932	1921	1875	1.82	120	96.02
cgcut1	260	244	260	260	260	0.7	120	100.0
cgcut2	2919	2892	2909	2909	2909	115.29	240	99.66
cgcut3	2020	1860	1940	1922	1900	6.3	60	96.04
gcut1	62488	48368	58136	58136	58136	0	60	93.04
gcut2	62500	59798	60656	60489.1	60431	33.8	60	97.05
gcut3	62500	61274	61275	61009.4	60663	47.18	60	98.04
gcut4	62500	61380	61710	61684.2	61581	55.2	60	98.74
gcut5	249854	195582	233969	233969	233969	1.35	60	93.64
gcut6	249992	236305	239467	239467	239467	0.06	60	95.79
gcut7	249998	240143	245306	243401.2	242925	54.16	60	98.12
gcut8	250000	245758	247462	247188.6	246857	38.25	120	98.95
gcut9	997256	939600	953293	953293	953293	0.01	60	95.59
gcut10	999918	937349	938036	938036	938036	0.07	60	93.81
gcut11	1000000	969709	979580	975419.1	970433	30.68	60	97.96
gcut12	1000000	979521	987674	987674	987674	16.3	60	98.77
gcut13	9000000	>=8622498	8873551	8847773.9	8784146	302.34	480	98.60
okp1	29133	27718	28423	27968.9	27538	74.75	240	97.56
okp2	24800	22502	24263	23489	22804	3.09	120	97.83
okp3	26714	24019	25216	24727	23777	42.48	120	94.39
okp4	33631	32893	32893	32882.1	32784	35.52	120	97.81
okp5	29045	27923	27971	26980.7	25712	96.41	240	96.31
wang20	2800	2726	2758	2758	2758	59.25	120	98.50
3	2020	1860	1940	1922	1900	5.02	60	96.04
3s	2800	2726	2758	2756.4	2756	25.22	60	98.50
a1	2140	2020	2080	2038	2000	7.79	60	97.20
a1s	3000	2956	2985	2985	2985	3.53	60	99.50
a2	2705	2615	2690	2647.5	2595	27.55	120	99.45
a2s	3600	3335	3579	3579	3579	4.46	120	99.42
chl2	2502	2326	2429	2422	2394	13.12	120	97.08
chl2s	3410	3336	3390	3390	3390	11.24	120	99.41
chl3	5283	5283	5283	5283	5283	0	480	100.0
chl3s	7402	7402	7402	7402	7402	0	480	100.0
chl4	8998	8998	8998	8998	8998	0	480	100
chl4s	13932	13932	13932	13932	13932	0	480	100.0
chl5	600	589	600	600	600	7.58	120	100.0

Table VIII: Results with rotation for the benchmark instances. ’1D’ is the result of the one-dimensional relaxed problem. ’No rotation’ is the value of the optimal solution without rotation. The columns ’Avg’, ’Best’ and ’Worst’ columns are the average, best and worst results on each instance for the 10 seeds. ’Best Time’ is the time before the heuristic discovered the best solution. ’Seed Time’ is the given to each of the 10 seeds. Thus the total running time on each instance is 10 times ’Seed Time’. ’Deviation’ is the deviation between the best solution with rotation and the optimal solution of the one-dimensional relaxed problem. For gcut13 no optimal values without rotation is know and we have reported the best solution from Table V

Instance	1D	No rotation	Best	Avg	Worst	Best Time	Seed Time	Deviation
ep-30-D-C-25	2155	2062	2068	2014.6	1953	49.02	120	95.96
ep-30-D-C-75	5135	5061	5059	5007.3	4947	231.53	480	98.52
ep-30-D-R-25	2364	2244	2308	2251.8	2208	60.26	120	97.63
ep-30-D-R-75	6191	6080	6121	6083.4	6040	108.39	480	98.87
ep-30-S-C-25	22494	21858	21858	21355.8	20692	40.17	120	97.17
ep-30-S-C-75	66935	65675	65254	64938.2	64257	174.34	480	97.49
ep-30-S-R-25	21674	20752	20679	20217.8	20067	33.62	120	95.41
ep-30-S-R-75	65319	64496	64356	63613.6	62765	68.13	480	98.53
ep-30-T-C-25	10938	9262	10412	10412	10412	0.81	120	95.19
ep-30-T-C-75	32377	32097	32129	31897.3	31675	92.48	480	99.23
ep-30-T-R-25	10647	9919	10287	10194.6	9919	0.42	120	96.62
ep-30-T-R-75	31750	31461	31417	31301.1	31221	454.41	480	98.95
ep-30-U-C-25	52002	49395	51839	51832.7	51776	24.52	120	99.69
ep-30-U-C-75	155306	145613	154176	153921.8	153509	408.7	480	99.27
ep-30-U-R-25	50246	50029	50136	50099.4	50053	117.12	120	99.78
ep-30-U-R-75	151710	147159	150795	150586.1	150381	428.89	480	99.40
ep-30-W-C-25	14060	13130	13801	13756.8	13673	63.04	120	98.16
ep-30-W-C-75	29780	21598	29403	29250.4	28916	412.64	480	98.73
ep-30-W-R-25	14840	14235	14648	14571.8	14560	14.09	120	98.71
ep-30-W-R-75	33396	23860	32878	32786.6	32670	104.04	480	98.45
ep-50-D-C-25	3232	3107	3033	2962.6	2912	54.55	240	93.84
ep-50-D-C-75	8849	8673	8672	8624.4	8596	440.8	480	98.00
ep-50-D-R-25	3546	3361	3399	3346.9	3293	153.3	240	95.85
ep-50-D-R-75	9678	9463	9526	9468.4	9405	259.74	480	98.43
ep-50-S-C-25	37185	36107	36541	35892.1	35021	86.99	240	98.27
ep-50-S-C-75	111329	109511	109380	108066	107270	125.62	480	98.25
ep-50-S-R-25	35599	34864	34924	34528.9	34223	204.86	240	98.10
ep-50-S-R-75	106699	104872	104398	103409.2	102063	377.16	480	97.84
ep-50-T-C-25	17757	17324	17508	17463.9	17441	217.14	240	98.60
ep-50-T-C-75	52863	50907	52089	51475.3	49593	462.5	480	98.54
ep-50-T-R-25	18643	18264	18336	18266.6	18217	33.88	240	98.35
ep-50-T-R-75	55475	54922	54809	54485.7	54084	474.01	480	98.80
ep-50-U-C-25	85978	80416	83575	82999	82935	127.31	240	97.21
ep-50-U-C-75	257446	248564	249462	249462	249462	39.36	480	96.90
ep-50-U-R-25	83736	78474	81379	80470.6	79851	68.58	240	97.19
ep-50-U-R-75	251412	245628	246572	241614.4	240375	200.1	480	98.07
ep-50-W-C-25	18082	17149	17715	17461.1	17297	226.62	240	97.97
ep-50-W-C-75	48909	46170	48476	47738.2	46637	217.76	480	99.11
ep-50-W-R-25	19215	18449	18857	18795.5	18756	189.73	240	98.14
ep-50-W-R-75	55475	54708	54492	54272.1	53764	407.09	480	98.23

Table IX: Results with rotation for the new instances. ’1D’ is the result of the one-dimensional relaxed problem. ’No rotation’ is the value of the best solution without rotation. The columns ’Avg’, ’Best’ and ’Worst’ columns are the average, best and worst results on each instance for the 10 seeds. ’Best Time’ is the time before the heuristic discovered the best solution. ’Seed Time’ is the given to each of the 10 seeds. Thus the total running time on each instance is 10 times ’Seed Time’. ’Deviation’ is the deviation between the best solution with rotation and the optimal solution of the one-dimensional relaxed problem.

Instance	1D	No rotation	Best	Avg	Worst	Best Time	Seed Time	Deviation
ep-100-D-C-25	6740	6316	6265	6171.2	6094	445.01	480	92.95
ep-100-D-C-75	18402	18003	17969	17852.7	17653	454.86	600	97.65
ep-100-D-R-25	8201	7804	7838	7802.5	7728	416.35	480	95.57
ep-100-D-R-75	23121	22635	22601	22509.9	22371	428.67	600	97.75
ep-100-S-C-25	73640	72154	71722	71200.9	69841	61.49	480	97.40
ep-100-S-C-75	219930	215294	215107	213695.9	212298	538.94	600	97.81
ep-100-S-R-25	89670	88334	88065	86913.9	85633	95.67	480	98.21
ep-100-S-R-75	267156	263237	261440	260008.3	256377	591.12	600	97.86
ep-100-T-C-25	34780	34123	34147	33947.9	33586	332.36	480	98.18
ep-100-T-C-75	102983	100645	101011	100386.1	99328	506.91	600	98.09
ep-100-T-R-25	35553	34948	34846	34672.6	34390	259.96	480	98.01
ep-100-T-R-75	105728	103888	103986	103513.3	102612	585.37	600	98.35
ep-100-U-C-25	170233	165313	169317	166949	164235	225.45	480	99.46
ep-100-U-C-75	511610	495431	502311	495617	488923	262.23	600	98.18
ep-100-U-R-25	171128	168779	170707	169411.6	168041	370.04	480	99.75
ep-100-U-R-75	512135	503471	505508	503315.1	499263	437.34	600	98.71
ep-100-W-C-25	31972	24325	31194	30870.9	30453	193.83	480	97.57
ep-100-W-C-75	95052	79803	92502	91626.3	90481	536.29	600	97.32
ep-100-W-R-25	38595	29593	37945	37703.2	37340	325	480	98.32
ep-100-W-R-75	115235	109102	112656	111547.1	110506	423.84	600	97.76
ep-200-D-C-25	13541	12415	12391	12146.7	11932	571.35	600	91.51
ep-200-D-C-75	36607	35489	35293	34911.6	34479	588.85	600	96.41
ep-200-D-R-25	16770	15919	15983	15785.6	15680	471.18	600	95.30
ep-200-D-R-75	46985	45867	45541	45317.4	45120	587.72	600	96.93
ep-200-S-C-25	147526	143875	143315	142748.6	141133	180.06	600	97.15
ep-200-S-C-75	440265	429097	427467	424157.2	421547	556.73	600	97.09
ep-200-S-R-25	172233	169314	169257	167738.6	165887	592.84	600	98.27
ep-200-S-R-75	514175	504328	500715	498968.5	497016	586.85	600	97.38
ep-200-T-C-25	69694	68208	68201	67591.2	66378	404.75	600	97.86
ep-200-T-C-75	206059	199431	199998	198990.2	197051	577.65	600	97.06
ep-200-T-R-25	65706	64335	64103	63813.8	62966	579.86	600	97.56
ep-200-T-R-75	194811	192034	189799	188440.2	186921	566.36	600	97.43
ep-200-U-C-25	340668	337958	338184	334282.4	328454	583.73	600	99.27
ep-200-U-C-75	1022712	991048	991055	981830.3	977775	519.57	600	96.90
ep-200-U-R-25	337830	334623	334407	331243.2	324314	437.31	600	98.99
ep-200-U-R-75	1014222	973279	985733	975732.4	967043	552.4	600	97.19
ep-200-W-C-25	64342	62657	62570	62240.1	61794	463.94	600	97.25
ep-200-W-C-75	189835	175161	182498	181175.9	178827	595.65	600	96.14
ep-200-W-R-25	75840	74331	74320	74103.6	73914	500.73	600	98.00
ep-200-W-R-75	225571	217963	219454	216625.1	213193	593.27	600	97.29

Table X: (See description of table IX)

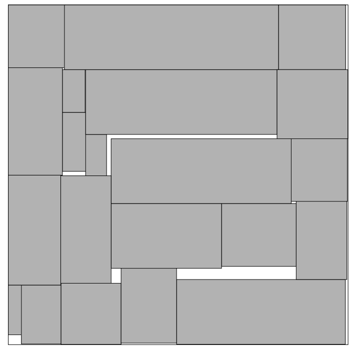


Figure 9: The best result achieved with gcut13 after 144 runs of 10 minutes each. The resulting profit is 8728123.

6.3.1 New Instances

As we were unable to locate any benchmark instances for the three-dimensional knapsack problem from the literature we have generated 60 random instances. The instances contain 20, 40 or 60 boxes. The dimensions of the boxes were chosen from 5 different classes described in Table XI. As for the two-dimensional case, boxes are clustered and random, and the container has volume equal to 50% or 90% of the total volume of the boxes. The naming convention is `EP`-n-c-t-p, where $n \in \{20, 40, 60\}$ is the number of boxes, $c \in (\mathrm{F}, \mathrm{L}, \mathrm{C}, \mathrm{U}, \mathrm{D})$ describes the class, $t \in (\mathrm{C}, \mathrm{R})$ describes if it is clustered or random, $p \in 50, 90$ describes the size of the bin in percentage of the total box volume. The profit of a box is always the volume of the box + 200 units. The instances are presented in Table XII and are available along with the source code to generate them at this web-address: `http://www.diku.dk/~pisinger/codes.html`. As for the two-dimensional instances we have determined two values for each instance; n_0 and n_1. $n_0 = n[\text{Knapsack volume}]/[\text{Total box volume}]$ and n_1 is the number of boxes selected in the one-dimensional relaxation where each item and the knapsack have a size equal to their three-dimensional counterpart and the profit of each item is the same as its three-dimensional counterpart.

For each instance we set the running time based on the function $F(n, n_0) = n_0 \lg n$, so that for $F(n, n_0) \leq 110$ the running time is set to 120 seconds, for $110 < F(n, n_0) \leq 200$ the running time is set to 300 seconds, and for $F(n, n_0) > 200$ the running time is set to 600 seconds.

6.3.2 Parameter Tuning

9 three-dimensional instances were selected for parameter tuning tests and we tried with $t_0 \in \{10^0, 10^1, 10^2, 10^3, 10^4, 10^5, 10^7, 10^8\}$ and $t_s \in \{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-1}, 10^0, 10^2, 10^4, 10^6\}$. Based on the 81 parameter combinations we found results similar to the two-dimensional

Class	Description	Width	Height	Depth
F	**Flat.** Boxes are flat	$[50,100]$	$[50,100]$	$[25,60]$
W	**Long.** Boxes are long	$[1,\frac{2}{3}\cdot 100]$	$[1,\frac{2}{3}\cdot 100]$	$[50,100]$
S	**Cubes.** Boxes are cubes	$[1,100]$	Equal to width	Equal to width
U	**Uniform.** Largest dimension is no more than 200% of the smallest	$[50,100]$	$[50,100]$	$[50,100]$
D	**Diverse.** Largest dimension can be up-to 50 times the smallest	$[1,50]$	$[1,50]$	$[1,50]$

Table XI: The 5 different classes of new EP3D instances. The width, height and depth of the boxes in each class are selected randomly from the intervals in the 'Width', 'Height' and 'Depth' column.

Instance	n	n_0	n_1	Instance	n	n_0	n_1
ep3d-20-C-C-50	20	9.9	15	ep3d-40-F-R-50	40	19.7	25
ep3d-20-C-C-90	20	17.8	12	ep3d-40-F-R-90	40	35.7	37
ep3d-20-C-R-50	20	9.9	14	ep3d-40-L-C-50	40	19.9	26
ep3d-20-C-R-90	20	17.8	16	ep3d-40-L-C-90	40	35.7	31
ep3d-20-D-C-50	20	9.8	12	ep3d-40-L-R-50	40	19.7	30
ep3d-20-D-C-90	20	17.8	18	ep3d-40-L-R-90	40	35.6	37
ep3d-20-D-R-50	20	10.3	15	ep3d-40-U-C-50	40	19.9	22
ep3d-20-D-R-90	20	17.8	18	ep3d-40-U-C-90	40	36.0	36
ep3d-20-F-C-50	20	9.9	11	ep3d-40-U-R-50	40	20.0	24
ep3d-20-F-C-90	20	17.8	17	ep3d-40-U-R-90	40	35.8	36
ep3d-20-F-R-50	20	9.9	12	ep3d-60-C-C-50	60	29.6	44
ep3d-20-F-R-90	20	18.0	17	ep3d-60-C-C-90	60	53.3	51
ep3d-20-L-C-50	20	9.9	9	ep3d-60-C-R-50	60	29.8	50
ep3d-20-L-C-90	20	17.7	15	ep3d-60-C-R-90	60	53.7	56
ep3d-20-L-R-50	20	9.9	14	ep3d-60-D-C-50	60	29.6	40
ep3d-20-L-R-90	20	17.6	18	ep3d-60-D-C-90	60	53.0	55
ep3d-20-U-C-50	20	10.0	10	ep3d-60-D-R-50	60	29.0	49
ep3d-20-U-C-90	20	17.9	18	ep3d-60-D-R-90	60	52.5	57
ep3d-20-U-R-50	20	9.9	11	ep3d-60-F-C-50	60	29.7	36
ep3d-20-U-R-90	20	18.0	17	ep3d-60-F-C-90	60	53.3	52
ep3d-40-C-C-50	40	19.9	27	ep3d-60-F-R-50	60	29.6	38
ep3d-40-C-C-90	40	35.3	30	ep3d-60-F-R-90	60	53.6	56
ep3d-40-C-R-50	40	19.9	32	ep3d-60-L-C-50	60	30.0	43
ep3d-40-C-R-90	40	35.6	37	ep3d-60-L-C-90	60	53.6	53
ep3d-40-D-C-50	40	19.5	24	ep3d-60-L-R-50	60	29.8	48
ep3d-40-D-C-90	40	35.2	31	ep3d-60-L-R-90	60	53.7	57
ep3d-40-D-R-50	40	19.3	32	ep3d-60-U-C-50	60	29.9	33
ep3d-40-D-R-90	40	35.3	37	ep3d-60-U-C-90	60	53.9	54
ep3d-40-F-C-50	40	20.0	26	ep3d-60-U-R-50	60	29.7	37
ep3d-40-F-C-90	40	35.8	34	ep3d-60-U-R-90	60	53.6	55

Table XII: New instances for 3DKP.

case and based on these we determined good values to be

$$t_0 = \frac{n_1^2}{5}, \qquad t_s = n_1^2.$$

6.3.3 Results

The results from our 3D tests are presented in Table XIII. For the instances with 20 items the gap between the found solution and the optimal value of the one-dimensional relaxation is relatively large. It may be due to the fact that it is impossible to fully utilize the available space within the three-dimensional knapsack – The dimensions of the boxes does not allow for this.

For the instances with 40 items we reach solutions which are better than 80% of the one-dimensional relaxation in almost half of the instances and in one instance we have as solution which is better than 90%.

For the instances with 60 items, we achieve results which are better than 80% of the one-dimensional relaxation for most of the instances, and we are even able to reach 90% in 3 of the instances.

The explanation could be that the larger the knapsack becomes the easier it is to get close to the one-dimensional relaxation bound, because the dimensions of the boxes are small compared to the knapsack and this allows for greater flexibility.

The best results of 8 of the instances are presented in Figure 10. To the best of our knowledge no published papers have reported utilization results for three-dimensional knapsack packing problems of the sizes we consider here, so it is difficult to compare our results to other approaches.

Methods for container loading (e.g. [13]) generally capable of achieving filling rates of around 90%. However, these problem instances consider far more boxes than the method we have presented here. Since our problems are smaller, it may be harder to achieve high filling rates, so the obtained filling rates around 80-90% for large-sized instances are very promising.

7 Conclusion

In this paper we have presented heuristic approaches for the two- and three-dimensional knapsack problem. The heuristics are based on Simulated Annealing. For the two-dimensional knapsack problem we utilize an abstract representation for rectangle packings called sequence pair and for the three-dimensional problem we utilize a novel abstract representation for box packings called sequence tripple. We have proved that the sequence tripple is able to represent any fully robot packable packing.

The heuristic for two dimensions is generally able to reproduce the results of exact algorithms with similar running times. The heuristic also gives the best known results for the only unsolved classical-instance; gcut13. To demonstrate the high quality of the results of the heuristic for larger instances we have created a new set of instances with up-to 200 rectangles and also here the heuristic performs extremely well by generating results higher 95% of our upper-bound.

Instance	1D	Best	Avg	Worst	Best Time	Seed Time	1D Percentage
ep3d-20-C-C-50	1026348	633672	612688	591704	26.51	120	61.7
ep3d-20-C-C-90	1834340	916241	916241	916241	0.01	120	50.0
ep3d-20-C-R-50	2188245	1492413	1492413	1492413	0.01	120	68.2
ep3d-20-C-R-90	3925057	2497691	2475532.8	2386900	86.58	120	63.6
ep3d-20-D-C-50	395916	239532	238906	233272	64.04	120	60.5
ep3d-20-D-C-90	718692	468112	459724.8	456836	1.12	120	65.1
ep3d-20-D-R-50	240621	195937	193631.8	178164	0.09	120	81.4
ep3d-20-D-R-90	414188	318848	314384.6	290780	0.53	120	77.0
ep3d-20-F-C-50	2395087	1900250	1900250	1900250	0.02	120	79.3
ep3d-20-F-C-90	4304020	2989393	2857081.7	2651170	0.09	120	69.5
ep3d-20-F-R-50	2252037	1563997	1547242.3	1466902	0.05	120	69.5
ep3d-20-F-R-90	4099982	2918002	2881536.5	2793585	0.57	120	71.2
ep3d-20-L-C-50	1064487	834335	821443.8	801375	20.78	120	78.4
ep3d-20-L-C-90	1894489	1589303	1563319.1	1560432	99.63	120	83.9
ep3d-20-L-R-50	718561	569900	554808.5	418985	2.41	120	79.3
ep3d-20-L-R-90	1282710	1051084	1029080.9	962400	31.6	120	81.9
ep3d-20-U-C-50	4495440	3088676	3088676	3088676	0	120	68.7
ep3d-20-U-C-90	8067424	5360280	5326935.6	5113988	5.82	120	66.4
ep3d-20-U-R-50	4413077	3509748	3486244.6	3433478	0.77	120	79.5
ep3d-20-U-R-90	8041072	6921250	6712730.1	6359659	1.49	120	86.1
ep3d-40-C-C-50	2065540	1265664	1265664	1265664	0.04	120	61.3
ep3d-40-C-C-90	3652448	2828160	2828160	2828160	0.21	300	77.4
ep3d-40-C-R-50	4102972	3002269	2760843.9	2643102	88.17	120	73.2
ep3d-40-C-R-90	7335602	5972946	5937447.5	5704665	47.04	300	81.4
ep3d-40-D-C-50	788124	539040	525116.4	523276	19.47	120	68.4
ep3d-40-D-C-90	1423896	1126300	1124263.6	1119512	17.75	300	79.1
ep3d-40-D-R-50	399894	349470	338144.5	328861	5.44	120	87.4
ep3d-40-D-R-90	728248	639819	612172.9	593147	202.97	300	87.9
ep3d-40-F-C-50	4816926	3590244	3538845.4	3427480	0.88	120	74.5
ep3d-40-F-C-90	8664122	6435962	6158772.9	5829899	96.43	300	74.3
ep3d-40-F-R-50	4518343	3477469	3407281.7	3280100	33.15	120	77.0
ep3d-40-F-R-90	8199224	7336067	7233223.4	7107398	50.31	300	89.5
ep3d-40-L-C-50	2127316	1675122	1659816.2	1649077	0.42	120	78.7
ep3d-40-L-C-90	3819412	2943657	2815563.7	2700358	99.75	300	77.1
ep3d-40-L-R-50	1784686	1609648	1579902.6	1538537	14.92	120	90.2
ep3d-40-L-R-90	3224295	2699629	2618748	2484532	84.49	300	83.7
ep3d-40-U-C-50	8988536	7008136	7008136	7008136	0.2	120	78.0
ep3d-40-U-C-90	16241380	14065676	13761564.4	13449344	79.29	300	86.6
ep3d-40-U-R-50	8666294	7766238	7653893.4	7553251	6.96	120	89.6
ep3d-40-U-R-90	15531980	13077284	12759327	12502175	200.88	300	84.2
ep3d-60-C-C-50	3063219	1504980	1504980	1504980	0.18	300	49.1
ep3d-60-C-C-90	5517671	4475024	4461790.4	4443374	22.92	600	81.1
ep3d-60-C-R-50	6493464	5695120	5250054	4621686	182.83	300	87.7
ep3d-60-C-R-90	11675188	10209801	9970784.5	9724806	374.56	600	87.5
ep3d-60-D-C-50	1200408	1057032	1014487.6	983668	68.89	300	88.1
ep3d-60-D-C-90	2143544	1843584	1786826.8	1736392	6.06	600	86.0
ep3d-60-D-R-50	538113	484363	469189.2	449308	158.52	300	90.0
ep3d-60-D-R-90	966582	861655	847241.2	831469	521.42	600	89.1
ep3d-60-F-C-50	7193700	6257697	6255808.2	6250424	149.27	300	87.0
ep3d-60-F-C-90	12913715	10412682	10196815.2	9972028	199.47	600	80.6
ep3d-60-F-R-50	6780100	6146420	5987831.3	5824694	277.61	300	90.7
ep3d-60-F-R-90	12301636	10866326	10597347	10283829	435.17	600	88.3
ep3d-60-L-C-50	3211612	2327139	2256880.4	2199391	164.39	300	72.5
ep3d-60-L-C-90	5736894	4832080	4742354.4	4665578	184.46	600	84.2
ep3d-60-L-R-50	2391507	2042317	2014109.8	1977414	135.01	300	85.4
ep3d-60-L-R-90	4304649	3872594	3803699.9	3710530	286.07	600	90.0
ep3d-60-U-C-50	13508800	12033592	11506459.2	10609988	117.85	300	89.1
ep3d-60-U-C-90	24342664	19787768	19474422.8	18970932	529.11	600	81.3
ep3d-60-U-R-50	12097660	10857656	10608234.4	10343738	108.3	300	89.8
ep3d-60-U-R-90	21893096	19304585	19047133.3	18549711	382.61	600	88.2

Table XIII: Results for the new ep3d instances. ’1D’ is the optimal solution of the one-dimensional relaxed problem. ’Best’, ’Avg’ and ’Worst’ columns are the best, average and worst results for each instance on the 10 seeds. ’Best Time’ is the time it took before the best solution was encountered. ’Seed Time’ is the time spent on each seed and the total time for each instance is 10 times ’Seed Time’. ’1D Percentage’ is the percentage deviation between the heuristic solution and the one-dimensional relaxed problem upper-bound.

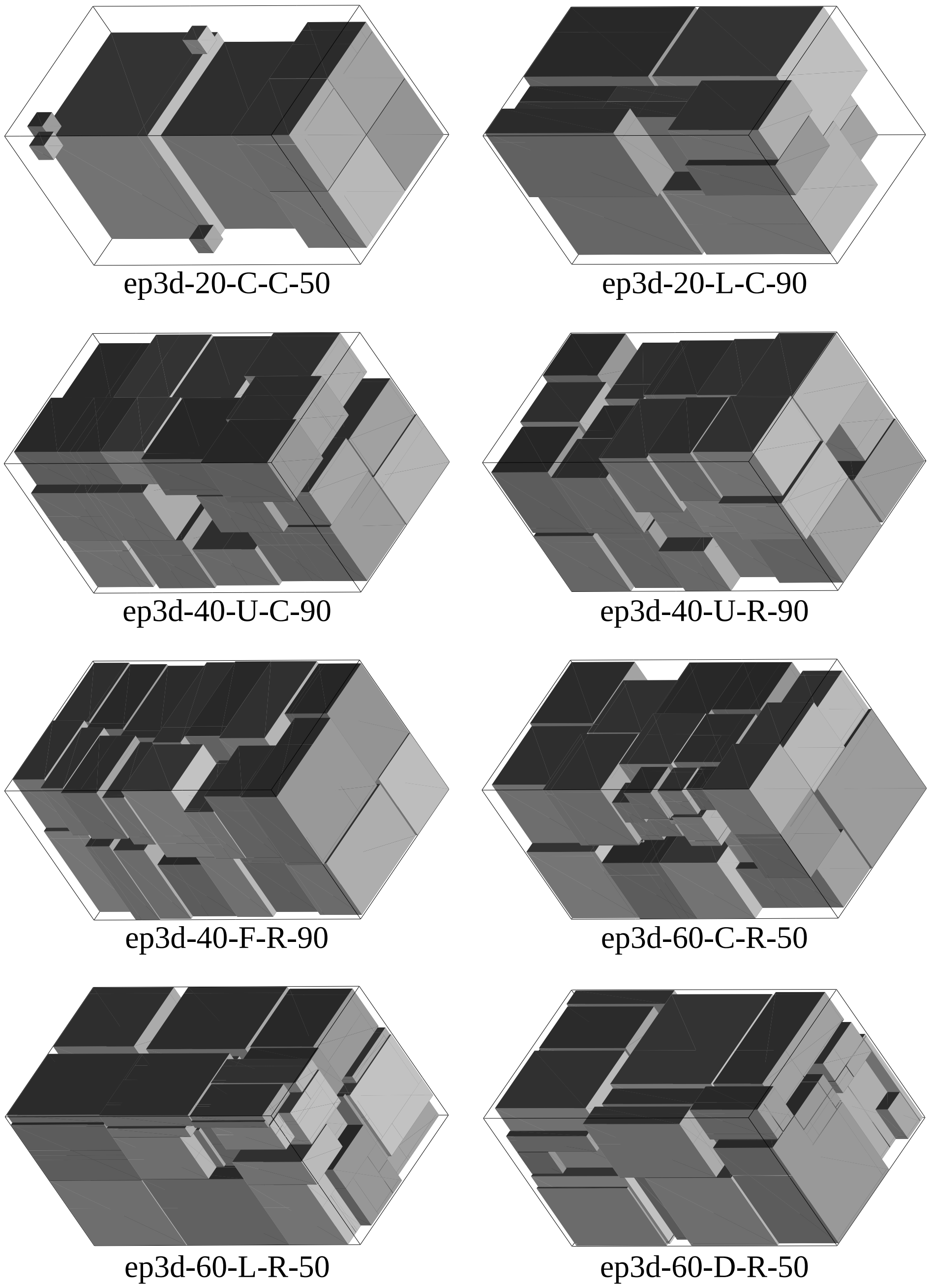


Figure 10: Best results for 8 three-dimensional instances.

The heuristic for three dimensions demonstrates the potential for the sequence tripple representation. We cannot compare the results for three-dimensional problems with results from other authors because of the lack of a benchmark set. To remedy this, we have created a new benchmark set for three-dimensional knapsack problems. Our heuristic performs well for these problems often returning results above 85% of the value of the upper-bound. Since the upper-bound is based on the one-dimensional relaxed problem and we expect this value to be a quite poor upper-bound, the results are promising.

The heuristics are generally able to return very good results for both two- and three-dimensional problems within few minutes, and often within few seconds for the classical two-dimensional benchmark instances.

References

[1] J.E. Beasley. Algorithms for two-dimensional unconstrained guillotine cutting. *Journal of the Operational Research Society*, 36:297–306, 1985.

[2] M.A. Boschetti, E. Hadjiconstantinou, and A. Mingozzi. New upper bounds for the two-dimensional orthogonal cutting stock problem. *IMA Journal of Management Mathematics*, 13:95–119, 2002.

[3] A. Caprara and M. Monaci. On the 2-dimensional knapsack problem. *Operations Research Letters*, 1(32):5–14, 2004.

[4] S. P. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In *Algorithms ESA '97, Springer Lecture Notes in Computer Science*, volume 1284, pages 144–156, 1997.

[5] S. P. Fekete and J. Schepers. On more-dimensional packing III: Exact algorithms. *submitted to Discrete Applied Mathematics*, 1997.

[6] S. P. Fekete, J. Schepers, and J. C van der Veen. An exact algorithm for higher-dimensional orthogonal packing. Technical Report cs/0604045, ArXiv Computer Science e-prints, 2006.

[7] E. Hadjiconstantinou and N. Christophides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of Operational Research*, 83:39–56, 1995.

[8] Mhand Hifi. Two-dimesional (un)constrained cutting stock problems. *http://www.laria.u-picardie.fr/hifi/OR-Benchmark/2Dcutting/*, 2006.

[9] H.Murata, K.Fujiyoshi, S.Nakatake, and Y.Kajitani. Vlsi module packing based on rectangle-packing by the sequence pair. *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, 15:1518–1524, 1996.

[10] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip packing problem. *INFORMS Journal on Computing*, 3(15):310–319, 2003.

[11] S. Martello, D. Pisinger, D. Vigo, Edgar den Boef, and Jan Korst. Algorithms for general and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software*, page to appear, 2006.

[12] D. Pisinger. A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 45:758–767, 1997.

[13] D. Pisinger. Heuristics for the container loading problem. *European Journal of Operations Research*, 3(141):382–392, 2002.

[14] D. Pisinger. Denser packings obtained in O(n log log n) time. *INFORMS Journal on Computing*, to appear, 2006.

[15] D. Pisinger and M. Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, to appear, 2005.

[16] D. Pisinger and M. M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem. *INFORMS Journal on Computing*, to appear, 2006.

[17] X.Tang and D.F.Wong. Fast-sp: a fast algorithm for block packing based on sequence pair. In *Asia and South Pacific Design Automation Conference*, 2001.

[18] X.Tang, R.Tian, and D.F.Wong. Fast evaluation of sequence pair in block placement by longest common subsequence computation. In *Proceedings of DATE 2000 (ACM), Paris, France*, pages 106–110, 2000.

[19] P. Y.Wang. Two algorithms for constrained two dimensional cutting stock problems. *Operations Research*, 31:573–586, 1983.