Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Parallel Computing 29 (2003) 641–662

PARALLEL
COMPUTING

www.elsevier.com/locate/parco

A parallel tabu search algorithm for solving the container loading problem

A. Bortfeldt *, H. Gehring, D. Mack

FernUniversität, Fachbereich Wirtschaftswissenschaft, Postfach 940, 58084 Hagen, Germany

Received 18 July 2002; accepted 15 October 2002

Abstract

This paper presents a parallel tabu search algorithm for the container loading problem with a single container to be loaded. The emphasis is on the case of a weakly heterogeneous load. The distributed-parallel approach is based on the concept of multi-search threads according to Toulouse et al. [Issues in designing parallel and distributed search algorithms for discrete optimization problems, Publication CRT-96-36, Centre de recherche sur les transports, Universitéde Montréal, Canada, 1996] i.e., several search paths are investigated concurrently. The parallel searches are carried out by differently configured instances of a tabu search algorithm, which cooperate by the exchange of (best) solutions at the end of defined search phases. The parallel search processes are executed on a corresponding number of LAN workstations. The efficiency of the parallel tabu search algorithm is demonstrated by an extensive comparative test including well-known reference problems and loading procedures from other authors.

Keywords: Container loading problem; Tabu search; Distributed-parallel search

1. Introduction

In the area of production and distribution of goods the efficient use of transportation devices such as containers and palettes is of high economic relevance. A high utilization of the applied transportation capacities causes considerable cost savings. Further effects are the reduction of the goods traffic and the protection of natural

* Corresponding author.
E-mail address: andreas.bortfeldt@fernuni-hagen.de (A. Bortfeldt).

0167-8191/03/$ - see front matter
doi:10.1016/S0167-8191(03)00047-4

resources. Computer supported packing methods can considerably contribute to the achievement of these goals.

The modelling of practical problems concerning the optimal utilization of transportation devices leads to different kinds of packing problems. An overview of the different types of packing problems and related cutting problems is given by Dyckhoff and Finke [7]. In this contribution the problem of optimally loading a single container, also known as container loading problem, is considered. It can be characterized as follows.

Let a rectangular container and a set of rectangular packing pieces be given. The latter contain the shipped goods and are referred to as boxes. In general, the sum of the volumes of the boxes exceeds the volume of the container. The goal is to determine a feasible arrangement of a subset of boxes which maximizes the stowed box volume and meets the given loading constraints.

An arrangement of boxes in the container is called feasible if the following conditions are respected:

- each box is placed completely within the container;
- each box does not overlap with another box;
- each box is arranged parallel to the side walls of the container.

In practice the loading of containers has to consider a great number of different constraints (cf. [1]). Here, only the following two constraints are included in the problem formulation.

1.1. (C1) Orientation constraint

If it is required by the storage of the goods within a box, one or two side dimensions of the box may not be used as the height.

1.2. (C2) Stability constraint

In the interest of stability of the load, both horizontal dimensions of each box are to be supported according to a predefined percentage. In any case the centre of gravity of each box must be supported in order to avoid boxes tipping over. It is assumed that the centre of gravity and the geometric centre of each box coincide.

Box types are defined as follows. Two boxes are the same type if they coincide in all three side dimensions. On the basis of this concept of box types, the following three categories of box sets can be distinguished. A homogeneous box set is given if all boxes are of the same type. A box set is called weakly heterogeneous if there exist a few box types and many items per type. Finally, a strongly heterogeneous box set is characterized by a greater number of box types and only few items per type. Here, a weakly heterogeneous set of boxes is assumed.

In the recent years, many (sequential) solution methods for the container loading problem have been developed. It is well known that the container loading problem is NP-hard (cf. [18]). Hence, the methods developed are heuristic approaches. Problem

specific heuristics are proposed by Loh and Nee [13], Ngoi et al. [15], Bischoff et al. [2] and Bischoff and Ratcliff [1]. Intelligent graph search algorithms go back, e.g., to Morabito and Arenales [14] and Pisinger [16], while Gehring and Bortfeldt [8] and Bortfeldt and Gehring [5] present genetic algorithms (GAs). Tabu search algorithms (TSAs) are introduced by Sixt [19] and Bortfeldt and Gehring [4].

In the recent years, parallel algorithms have been successfully applied to different combinatorial optimization problems, for example to the vehicle routing problem with time windows (cf. e.g., [11,17]). Depending on the chosen parallelization approach (see Section 4), the concept of parallelization has various advantages. These are, e.g., the reduction of the computing time and the enhancement of the solution quality for a given computing time. Parallel approaches should be considered especially in situations where particularly hard combinatorial optimization problems or realistically sized problem instances are to be handled. The container loading problem considered here is an extremely hard combinatorial problem with a particularly large solution space (cf. e.g., [3]). Therefore, the application of a parallelization concept is undoubtedly obvious.

Nevertheless, there exist only a few parallel approaches for three-dimensional packing problems. A parallel TSA is proposed by Gehring and Bortfeldt [9] and a parallel GA by Gehring and Bortfeldt [10].

In this paper a parallel TSA for the container loading problem as characterized above is described. The algorithm is the result of a further development of the approach from Gehring and Bortfeldt [9]. The algorithm is hierarchically structured into three modules:

- The lowest module consists of a simple heuristic, called basic heuristic, which serves the complete loading of a container.
- The middle module contains a sequential TSA. For each solution generated by the TSA the basic heuristic is applied once. For the purpose of diversification, the search process is subdivided into several phases each carried out by the same but differently configured TSA.
- Within the uppermost module several differently configured instances of the TSA evolve independent search paths. The instances cooperate through the exchange of best solutions. The exchange always takes place at the end of defined search phases and exerts an influence on the further search of the individual instances.

The rest of the paper is structured as follows. In Section 2, the basic heuristic is described and in Section 3 the sequential TSA is presented. The subject of Section 4 is the parallelization of the TSA. The results of numerical tests are reported and evaluated in Section 5. Finally, in Section 6 the contribution is summarized and an outlook on further research is given.

2. The basic heuristic

By means of the basic heuristic a given container is loaded in several iterations. Within an iteration a so-called packing space is filled with one or more boxes.

A packing space is an empty rectangular space within the container with defined side dimensions. In the first iteration the complete interior of the container is used as the packing space. For the loading of a packing space only box arrangements with a predefined simple structure are considered. These are called local arrangements. The feasible local arrangements for a packing space are generated and evaluated by means of certain criteria. The unused part of the packing space is completely subdivided into several residual packing spaces. These are filled later. A rough description of the algorithm of the basic heuristic is given in Fig. 1.

The overall algorithm presented in Fig. 1 requires some comments.

- In order to enhance the chances of loading small packing spaces, the packing space with the smallest volume is always processed first.
- The container is embedded in a three-dimensional coordinates system. The bottom left-hand rear corner of a packing space is used as the reference corner. The coordinates of the reference corner are stored together with the dimensions of the packing space. The position of a box results from the coordinates of the reference corner of the respective packing space and its placement within the respective local arrangement (see below).
- In this section, the basic heuristic is presented as a greedy heuristic. In step (5) the best evaluated first local arrangement of *ArrList* is selected. In Section 3 the basic heuristic is extended in such a way that the best arrangement is not necessarily used for a packing space with packing space index *ipr*. Only with this extension can the basic heuristic be used for the generation of different solutions to a prob-

(1) Initialize:
the set of residual boxes *BRes* := set of all boxes;
the packing space list *PrList* := {Container};
the packing space index *ipr* := 0;
the stowing list *StList* := { }.

(2) Determine the current packing space *pcurr* as the packing space from *PrList* with minimum volume and delete *pcurr* from *PrList*.

(3) For packing space *pcurr*, initialize the arrangement list as empty list *ArrList* := { }. Generate and evaluate all local arrangements for *pcurr*. Insert the local arrangements in descending order with respect to the evaluation into the arrangement list *ArrList*.

(4) If *ArrList* is empty, go to step (8).

(5) Update the packing space index *ipr* := *ipr* + 1. Insert the pair (*pcurr*, *ArrList* (1)) as *ipr*-th element into the stowing list *StList*.

(6) Insert the residual packing spaces for the packing space *pcurr* and the local arrangement *ArrList*(1) into the packing space list *PrList*.

(7) Update the set of residual boxes *BRes*.

(8) If the packing space list *PrList* is not empty, then go to step (2).

(9) Stop.

Fig. 1. Overall algorithm for the basic heuristic.

lem instance. It should be mentioned that an index *ipr* is only assigned to fillable packing spaces in which at least one box can be placed.

- At the same time as a local arrangement is generated and evaluated (step 2), the residual packing spaces that would occur if this local arrangement was used are generated. In step (6) these residual packing spaces are possibly inserted into the packing space list *PrList*.

From the last comment it can be concluded that a more detailed description of the basic heuristic requires merely a refinement of step (3), which is subsequently described.

2.1. Generation of local arrangements

The structure of local arrangements for a packing space is defined as follows.

A local arrangement consists of one or two so-called blocks and is therefore referred to as a 1- or 2-arrangement (see Fig. 2). A block is formed from boxes of the same type. Furthermore, all boxes of a block are arranged in an identical spatial orientation variant. In each of the three dimensions (x-, y- and z-direction) a block consists of one or more boxes.

The only block of a 1-arrangement is always placed in the reference corner of the packing space. From the two blocks of a 2-arrangement, one is arranged in the reference corner. The second block can alternatively be placed as a neighbour in x-direction (arrangement type "in front of"), as a neighbour in y-direction (arrangement type "beside") or as a neighbour in z-direction (arrangement type "above"). In the case of a placement according to arrangement type "in front of", the block with the larger y-dimension is positioned in the reference corner, while for the arrangement type "beside" the block with the larger x-dimension is positioned in the reference corner. The arrangement type "above" is only used if both horizontal dimensions of a block are not smaller than the corresponding dimensions of the other block. The block with the larger horizontal dimensions is positioned in the reference corner and the other above. Fig. 2 illustrates a 1-arrangement and two 2-arrangements of the arrangement types "in front of" and "beside".

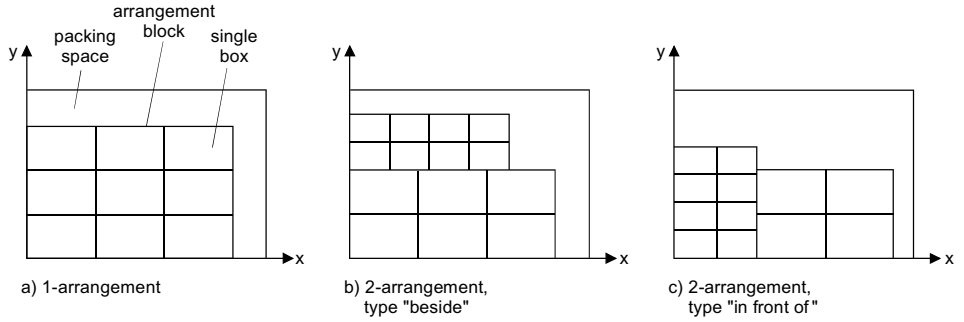

Fig. 2. 1-Arrangement and 2-arrangements within a packing space (overhead view).

For the blocks of an arrangement, the box numbers in all three dimensions are first determined in such a way that the concerned dimensions of the packing space are utilized as fully as possible. If the number of boxes of a given type required for a block exceeds the number of still available items of this type, then the numbers of boxes are reduced appropriately.

With the selection of a box type and an orientation variant for its block, an 1-arrangement is defined unambiguously. Analogously a 2-arrangement is completely defined by the selection of two box types, two orientation variants, and an arrangement type. All 1-arrangements and all 2-arrangements, which can occur if the box types, the orientation variants and—in the case of 2-arrangements—the arrangement type are varied, are generated. However, only those box types for which at least one item is still available are considered here. Furthermore, the variation of the orientation variants has to take the orientation constraint (C1) into consideration.

2.2. Generation of residual packing spaces

Immediately after the generation of a local arrangement for a packing space, the unused part of the packing space is subdivided into residual packing spaces. In order to enable an evaluation of a local arrangement (see below), different subdivisions into residual packing spaces are experimentally generated.

For the complete subdivision of the unused part of a packing space, several variants always exist. Type and number of the different subdivision variants depend on two criteria. The first concerns the type of the local arrangement within the packing space, i.e. whether it is a 1-arrangement or a 2-arrangement and, in the case of a 2-arrangement, which arrangement type is given. The second refers to the support of stowed boxes, i.e. whether the stability constraint (C2) demands a complete or only a partial support of each stowed box. In the first case, only those packing spaces that lie completely on the container bottom or on the top of blocks are considered. If only

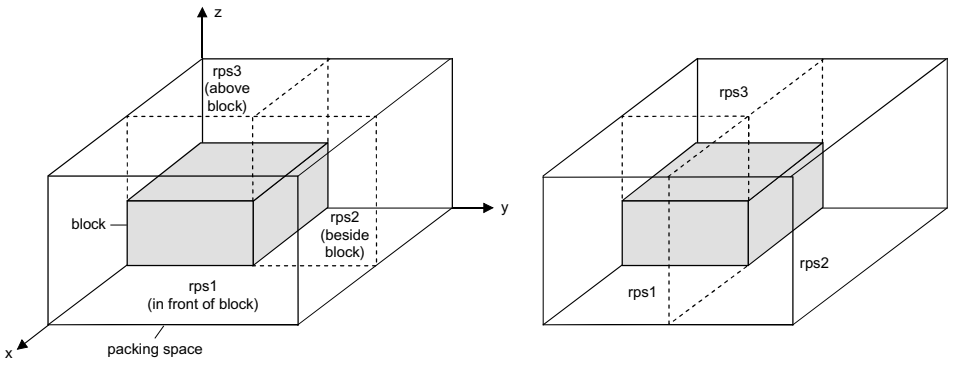

Fig. 3. Subdivision variants for a packing space and a 1-arrangement without overhanging residual packing spaces (rps), resudial packing space.

a partial support is required, then (residual) packing spaces, that project over their supporting boxes and thus cause a lateral overhang, are also to be considered.

For clarifying purposes, different 1-arrangements are illustrated in the following. In the case of complete support, there exist only two subdivision variants which are shown in Fig. 3.

In the case of a partial support, there exist four further subdivision variants, which are illustrated in Fig. 4.

As regards 2-arrangements, only the numbers of subdivision variants are stated here. For 2-arrangements of the types "in front of" and "beside", there always exist five subdivision variants without and 26 variants with overhanging packing spaces. For a 2-arrangement of the type "above", four subdivision variants without and six variants with overhanging packing spaces are to be considered.

The subdivision variants corresponding to the type of an arrangement and the given stability constraint are calculated and evaluated. In general, the evaluation is based on two criteria. The first is the loss volume; it denotes the sum of the volumes of the unfillable residual packing spaces and should be as small as possible. The second is the maximum effective volume of all residual packing spaces, which should be as large as possible. In the case of a packing space without overhang, the effective volume is simply its volume. For a packing space with overhang, however, the usable volume of the packing space is estimated on the basis of the overhanging part and

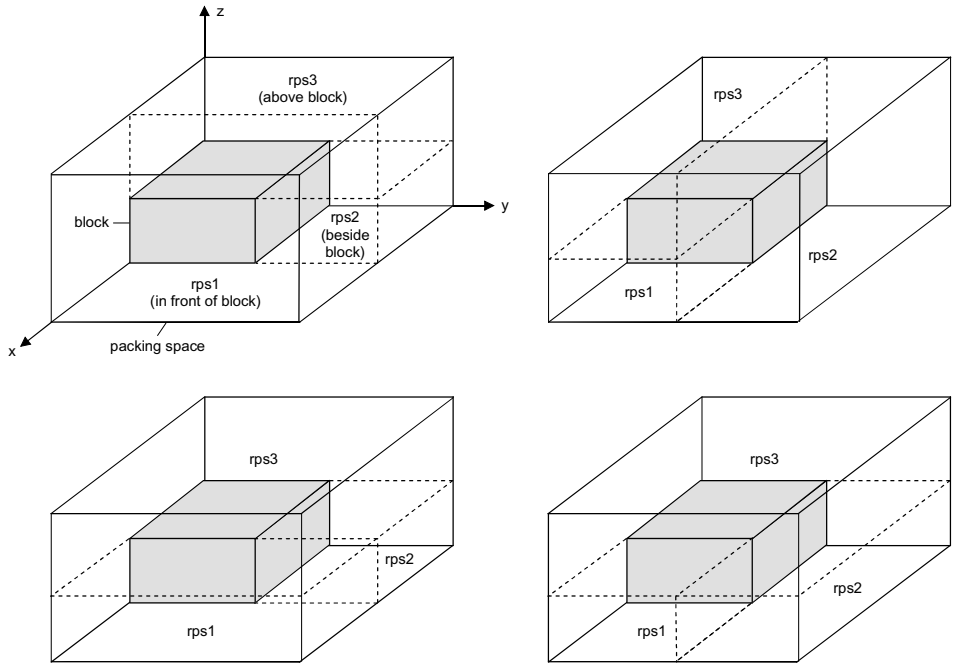

Fig. 4. Additional subdivision variants for a packing space and a 1-arrangement with overhanging residual packing spaces (rps), resudial packing space.

Table 1
Evaluation modes for subdivisions of a packing space

Evaluation mode *cutEvalMode*	Stability constraint/required support	Comments
0	Complete	1. Criterion: loss volume 2. Criterion (tie breaker): maximum effective volume
	Partial	The same criteria as for complete support
1	Complete	Not applicable
	Partial	Only subdivisions without overhanging packing spaces are considered Criterion: maximum effective volume The best subdivision is used again • Unfillable residual packing spaces on the bottom of the original packing space are horizontally cut, and • The upper parts are transferred to higher positioned residual packing spaces

used as effective volume. It should be noted that the box numbers in the horizontal directions for the blocks of an arrangement also depend on whether a packing space does or does not have overhang.

The evaluation is based on two alternative modes; the selection of the relevant mode is controlled by the parameter *cutEvalMode*. Application and effects of a mode depend on the given stability constraint. Further details are presented in Table 1.

For a packing space and an arrangement type, the residual packing spaces finally result from the determined best evaluated subdivision variant.

2.3. Evaluation of local arrangements

For the evaluation of the local arrangements generated for a packing space, two modes are available which are applied alternatively. The selection of the relevant mode is also controlled by a parameter, named *arrEvalMode*.

The first mode, encoded by the parameter value 0, applies a single evaluation criterion: the total volume of the boxes stowed in the packing space which should be as large as possible.

The second mode, encoded by the parameter value 1, additionally applies two further evaluation criteria. These are the already introduced quantities loss volume and maximum effective volume. Both criteria refer to the residual packing spaces of a local arrangement. Analogous to the evaluation of subdivisions, the loss volume should be as small as possible and the maximum effective volume as large as possible. Since the three evaluation criteria applied are weighted equally, the evaluation procedure is organized as a series of comparisons of the local arrangements for a packing space in pairs.

Finally, two additional parameters of the basic heuristic are introduced and briefly discussed. The parameter *maxArr* defines the maximum length of the arrangement list *ArrList* for a packing space. A local arrangement is only considered in the

tabu search process if it occurs in *ArrList*, i.e. belongs to the *maxArr* best arrangements. The parameter *aboveArr* determines whether 2-arrangements of type "above" are generated (parameter value 1) or not (parameter value 0). Like the different modes for subdivisions and arrangements, the parameter *aboveArr* serves the diversification of the tabu search.

3. The sequential tabu search algorithm

In Fig. 5, a generic tabu search algorithm is described on a sufficiently general level. The foundations of the search strategy tabu search are assumed to be well known (see e.g., [12]).

In the following, the generic TSA is adapted to the given container loading problem. The adaption is carried out in three steps:

(1) Introduction of the encoding of feasible solutions to the problem.
(2) Specification of the basic components of the TSA.
(3) Configuration of the TSA.

3.1. Encoding of feasible solutions

In order to define neighbourhoods for the tabu search that can be easily manipulated, an encoding of feasible solutions to the container loading problem is chosen.

```
Initialize:
    generate an initial solution s;
    set best solution s_best := s;
    set Tabulist := { };

Perform a neighbourhood search:
    while (termination criterion is not met) do
        generate a neighbourhood N(s);
        initialize the value of the objective function f(s_iter) := −∞;
        for all s' ∈ N(s) do
            if f(s') > f(s_iter) and solution s' is not tabu then
                s_iter := s';
            endif
        endfor
        if f(s_iter) > f(s_best) then
            s_best := s_iter;
        endif
        update Tabulist;
        s := s_iter;
    endwhile

Define the best solution s_best as solution to the problem.
```

Fig. 5. Generic algorithm of a tabu search for solving a maximization problem.

A feasible solution is encoded by means of a vector Ps, called packing sequence. The position ipr, $ipr = 1, 2, \ldots$, of this vector corresponds to a fillable packing space with the index ipr. The vector element at the position ipr is a data structure consisting of two components. The first component $Ps(ipr).ia$ specifies the index of a local arrangement of the arrangement list $ArrList$ for the iprth packing space. The second component $Ps(ipr).na$ records the length of this arrangement list.

The transformation of an encoded solution to a complete solution is carried out by means of the basic heuristic, which, however, has to be modified for this purpose. When the modified basic heuristic is called, a packing sequence Ps is handed over to the basic heuristic. Instead of the best local arrangement $ArrList(1)$, the arrangement $ArrList(ia)$ is now used for the iprth packing space; the index ia is determined here as $ia = Ps(ipr).ia$. Let, e.g., the packing sequence contain the arrangement index 2 at position 1 and the index 3 at position 2. Then the second arrangement of the corresponding arrangement list is selected for the first fillable packing space and the third arrangement for the second fillable packing space.

Only during the transformation can the length of the arrangement lists for the fillable packing spaces be determined and inserted into the packing sequence, Ps. All packing sequences have the same, sufficiently dimensioned length $maxpr$. The number of fillable packing spaces and therefore the number npr, $npr \leqslant maxpr$, of significant positions of Ps can only be determined during the transformation, as well. Together with the packing sequence Ps the number npr is kept in a superordinate data structure. An arrangement index is, however, always kept in all positions of a packing sequence. This and further technical measures ensure that any packing sequence can unambiguously be transformed into a feasible solution.

3.2. Specification of the basic components of the TSA

The tabu search is carried out in the space of encoded solutions. In this space two neighbourhood structures are introduced, which can be used alternatively. For any packing sequence Ps with npr significant positions, a large and a small neighbourhood are defined as follows:

(1) The large neighbourhood embraces all packing sequences Ps', for which the following applies: the recorded arrangement indices differ for exactly one position ipr', i.e. $Ps'(ipr').ia \neq Ps(ipr').ia$, and for the deviating index it is required that $Ps'(ipr').ia \in \{1, \ldots, Ps(ipr').na\}$; on the other hand, the arrangement indices must coincide for each of the remaining positions, i.e. $Ps'(ipr).ia = Ps(ipr).ia$ for all ipr with $1 \leqslant ipr \leqslant maxpr$, $ipr \neq ipr'$. The change position ipr' can be selected arbitrarily within the interval $1 \leqslant ipr' \leqslant npr$.

(2) The small neighbourhood is defined analogous by to the large neighbourhood. The only deviation concerns the change position ipr', which is pre-set on a constant value from the interval $1 \leqslant ipr' \leqslant npr$. But this fixing is always valid for one iteration only. In subsequent iterations the change position ipr' is cyclically varied starting with $ipr' = 1$. Assuming that the solution achieved in the jth iteration has been determined using the change position ipr' and includes npr

significant positions. Then, if $ipr' < npr$ holds, the change position $ipr' + 1$ is used in the $(j + 1)$th iteration; for $ipr' = npr$, on the other hand, the change position 1 is used again.

Both neighbourhood structures have in common that neighbouring packing sequences Ps and Ps' prescribe—with the exception of one position or packing space, respectively—the application of local arrangements that occupy the same place in the respective arrangement list. Hence, neighbouring packing sequences are similar with respect to the relative quality of their arrangements. In this sense both neighbourhoods can be characterized as environments in terms of quality. As a greedy heuristic, the unmodified basic heuristic fills all packing spaces in a local-optimizing way. The tabu search, on the contrary, aims at the determination of a solution near the global optimum and allows local highly favourable arrangements for a few packing spaces to be ignored. Precisely this goal is supported by the introduced neighbourhoods.

Within one iteration each neighbouring packing sequence Ps' for a given packing sequence Ps is twice transformed into a feasible solution by means of the modified basic heuristic. In the case of the first transformation, only 1-arrangements are assigned to all packing spaces, while at the second transformation 2-arrangements are also considered. Both (intermediate) solutions are evaluated and the solution with the higher value of the objective function is selected as the final result of both transformations. It should be noted that, due to the successive processing of several packing spaces, the first transformation variant can also lead to the better solution.

The tabu list and its application are designed as follows. After each iteration the tabu list is extended by the best solution of the iteration. The best solutions of the iterations are stored as complete packing sequences in the tabu list. In this way cycling is definitely avoided. Since the transformation effort is rather high, only a relatively small number of iterations can be calculated. The effort caused by the management of the tabu list is therefore small absolutely, as well as in comparison with the transformation effort. The maximum length of the tabu list is given by the number of iterations. With the approach chosen for the tabu list, aspiration criteria are not necessary.

As initial solution for the tabu search, a packing sequence is used which contains the arrangement index 1 at all positions. This means that the initial solution is de facto determined by means of the unmodified basic heuristic.

3.3. Configuration of the TSA

The configuration of the TSA includes its parameterization, the introduction of a diversification concept, and the definition of termination criteria.

In addition to the four parameters of the basic heuristic (cf. Section 2), three further parameters are introduced. The parameter *nIter* defines the number of iterations of the tabu search. The parameter *nbhType* states whether the large neighbourhood (parameter value 0) or the small neighbourhood (parameter value 1) is applied.

The size of neighbourhoods is already affected by the parameter *maxArr* of the basic heuristic; *maxArr* restricts the number of the local arrangements that are to

be considered per packing space. In addition, the parameter *nbhDecrease* is used for the reduction of both neighbourhoods; *nbhDecrease* can take integer values from 1 onwards. For a given change position ipr' it is assumed that from a total of $Ps(ipr').na$ possible arrangements within the packing sequence Ps, the arrangement $Ps(ipr').ia$ is the designated item. Then, only those alternative local arrangements are allowed for the change position ipr', the indices ia' of which meet the condition

$$|ia' - Ps(ipr').ia| \leqslant Ps(ipr').na/nbhDecrease.$$

For $nbhDecrease = 1$, the neighbourhoods will not be reduced. For $nbhDecrease > 1$, the alternative arrangements per change position are limited to those items that lie near the original arrangement with respect to the evaluation.

The method includes the following diversification concept: The overall search process is structured into *nphases* different search phases. In each phase the search starts from the same initial solution and with an empty tabu list. In order to achieve diversification, an independent set of the seven parameters mentioned above is applied in each phase. An additional diversification of the search can be caused by the stability constraint (C2), if only partial support of the stowed boxes is demanded. For this purpose, the required percentage of support with respect to both horizontal dimensions of a box is, to a certain extent, arbitrarily increased (cf. Section 5).

The termination of the overall method is caused by two criteria. The search is terminated, if all *nphases* search phases have been carried out or if the calculation time exceeds a given time limit *maxTime*. The best solution over all search phases is finally determined as the solution to the problem.

4. The parallel tabu search algorithm

According to Toulouse et al. [20], three types of parallelization strategies seem to be appropriate for the methods most often used in combinatorial optimization: (1) parallelization of operations within an iteration of the solution method, (2) decomposition of problem domain or search space, and (3) multi-search threads with various degrees of synchronization and cooperation. Which of these types is suited for the parallelization of an optimization method depends mainly on the goal pursued by the parallelization. Since the enhancement of the solution quality is in the foreground here, an approach of type 3 is chosen for the parallelization.

An instance of a container loading problem is treated by several processes. Each process is an instance of the sequential TSA and solves the complete problem. However, the individual instances are configurated differently. Furthermore, the processes cooperate through the exchange of calculated solutions. A transmitted solution is possibly used by the receiving process as a starting point for further search. While the varying configuration of the processes causes a diversification of the search, the exchange of solutions serves the intensification of the search within the regions of best solutions.

Each of the autonomous processes is assigned to a workstation of a local network (LAN). Hence, more precisely expressed, the parallel TSA is a distributed-parallel method. It is described more closely in the following.

4.1. Structuring of the search and communication frequency

According to the diversification concept of the sequential TSA, the search in each process is structured into several phases. In order to determine favourable parameter settings with respect to the definition of the phases of all processes, a series of experiments was carried out by means of the sequential TSA (cf. Section 5). The best parameter settings are distributed approximately evenly over the processes and per process the most promising parameter settings are, as far as possible, applied in early phases. In this way the intended intensified exploration of regions that contain solutions of high quality, is supported to a greater extent.

As to the communication frequency or the number of communications, the type of the underlying sequential method (here a TSA) is to be considered. The consequence of a very high communication frequency is that the individual processes are prevented from intensively exploring limited regions of the search space. Therefore, a lower communication frequency is to be chosen in advance. Here, an exchange of best solutions is only carried out at the transition from one phase to the next phase of each of the processes.

4.2. Communication model

The exchange of solutions is performed by means of a communication object. It records the solutions generated by the processes and makes the solutions available to them again. The organization of the communication object depends on the communication model applied. The latter determines the available communication paths. Two alternative communication models are provided here, a ring and a blackboard. The communication object is realized by means of a data base.

In the case of the ring, the data base is subdivided into separate areas. Each area is firmly assigned to a process. A process writes its solutions into its own data base area only. Each area is organized as a queue according to the FIFO principle. A new solution is always inserted at the end of the queue. The processes are connected in a

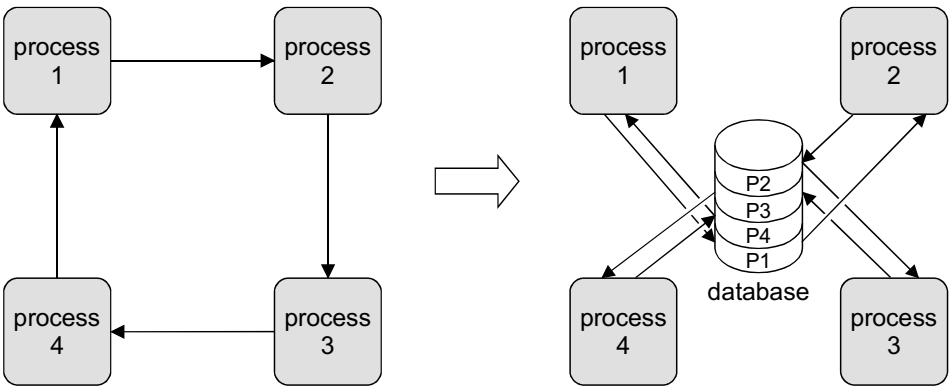

Fig. 6. The communication model ring.

way that forms a ring, i.e. each process has exactly one predecessor. A process always reads the solution at the end of the queue of its predecessor. Then the solution is removed. In Fig. 6 the communication model ring is shown.

In the case of the blackboard, the data base is organized as a common area, in which the processes write their solutions and from which they read out solutions. The area is managed as a stack according to the LIFO principle. A solution provided by a process is inserted as the top element of the stack and each process always reads the uppermost solution of the stack. In contrast to a ring, an accessed solution is not removed. Rather, it remains available for access by other processes until the next solution is inserted into the stack. Since, in general, the solution quality will be enhanced during the search, it is definitely intended that only the last inserted solution is available for reading processes. Fig. 7 illustrates the communication model blackboard.

4.3. Exchange of solutions

At the end of a phase, a process provides its best solution, i.e. the best solution found during the previous search, for the other processes. At the beginning of the subsequent phase, the process reads a solution that was provided by another process. The read solution possibly forms the new starting point for the search of the reading process. The next neighbourhood examined by the process is therefore the neighbourhood of the foreign solution.

Solutions are exchanged as packing sequences or encoded solutions, respectively. Furthermore, the parameters of the basic heuristic, which are valid in the phase in which the solution was found, are provided and taken over by the receiving process. Only in this way is it guaranteed, that the transfer of the packing sequence also leads to a solution of high quality on the side of the receiving process. If, therefore, a received foreign solution serves as a starting point for the next phase of the receiving process, then the parameters of the basic heuristic belonging to the foreign solution are applied in this phase.

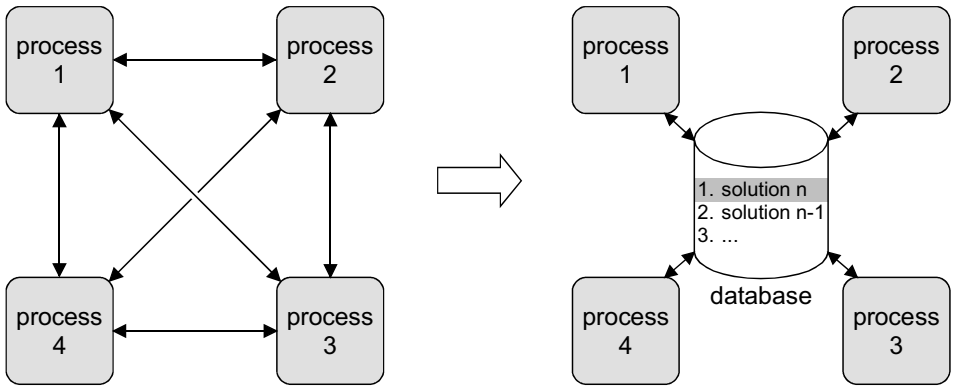

Fig. 7. The communication model blackboard.

Foreign solutions are processed in two alternative ways; one of them is always used for all processes. In the case of an unconditioned adoption of foreign solutions, a read foreign solution is always the starting point for the further search. In the case of a conditional adoption, the foreign solution is only used as a new starting solution if its value of the objective function is higher than the respective value of the present best solution of the process. Otherwise, the process continues the search as if the communication had not taken place. In particular, all predefined parameters are used for the next phase.

The adoption of a foreign solution leads to an enlargement of the subsequently explored neighbourhoods. The reason is that the parameter *nbhDecrease* of the receiving process is decremented by 1 (cf. Section 3). This measure ensures that the region around the adopted high-quality solution is subsequently subjected to an intensified search. In order to keep the increase of the computing time within limits, the total number of phases carried out by a process should be smaller than in the case of the sequential TSA.

4.4. Further details

One of the processes performing the concurrent search is excluded from the communication. Operating as sequential TSA this process carries out an isolated search. Its best generated solution is, however, finally included in the determination of the solution of the parallel method. Hence, the solution quality of the sequential method will be achieved in any case.

The termination of the parallel TSA is controlled by an additional process—the so-called master. The parallel method is terminated by the master, if either all processes have carried out all their search phases, or if the computing time consumed by the distributed-parallel system has exceeded a predefined time limit *maxTime*. After the end of the concurrent search, the solution of the parallel method is determined as the best solution found by the whole process group.

5. Numerical results

The test of the distributed-parallel approach includes two steps. In the first step the configurations of the sequential and the parallel TSA are determined. In the second step the two configured methods are subjected to a comparative test comprising container loading methods from other authors. All calculations were carried out on Pentium workstations with a cycle frequency of 2 GHz. In the following, the test problems are specified first and then the two test steps are evaluated.

5.1. Applied test problems

The test is based on well-known reference problems from the literature. These are the 15 test problems from Loh and Nee [13] and the 700 test problems from Bischoff and Ratcliff [1]. The problems from Loh and Nee may be classified as

weakly heterogeneous throughout. The problems from Bischoff and Ratcliff are subdivided into seven test cases each with 100 problems. In the seven test cases the character of the box sets changes from weakly heterogeneous to strongly heterogeneous. All problems of a test case are characterized by a constant number of box types.

The problem instances of both groups contain an orientation constraint (C1). To enable the consideration of a stability constraint (C2), an additional attribute *minSupport* is introduced for each instance. *minSupport* specifies the minimum extent to which both horizontal dimensions of a box are to be supported. In the test two extreme values are chosen for the extent of the support of boxes. The value $minSupport = 55\%$ ensures that, at minimum, boxes do not tip. For the value $minSupport = 100\%$, the bottom of each box must be supported completely.

5.2. Configuration pretests

First, the configuration of the sequential TSA is described and then the configuration of the parallel TSA is specified.

In a limited series of experiments it turned out that in the case of the sequential TSA, a subdivision of the search into six phases ($nPhases = 6$) is appropriate. Furthermore, the parameter settings for six search phases were determined for each of the two given values of *minSupport*. The parameter settings are shown in Table 2. Meaning and values of the parameters have been explained earlier (see Sections 2 and 3). As has already been mentioned, for a given value of $minSupport < 100$ this value may be increased arbitrarily for individual search phases in order to achieve additional diversification of the search. For the given value of $minSupport = 55$ and the third search phase, the respective phase-specific value of *phMinSupport* was therefore enhanced and set to the value 100. For the parameter *nbhDecrease*,

Table 2
Parameter settings for the sequential TSA

Phase-ID	*phMin-Support*	*cutEval-Mode*	*arrEval-Mode*	*maxArr*	*aboveArr*	*nbhType*	*nIter*	*nbh Decrease*
(a) Parameter settings for minSupport = 55								
1	55	0	1	100	1	1	100	3
2	55	0	1	50	1	0	20	3
3	100	0	1	100	0	1	100	3
4	55	0	0	100	0	1	100	3
5	55	1	1	50	1	0	20	3
6	55	1	1	100	1	1	100	3
(b) Parameter settings for minSupport = 100								
1	100	0	1	100	1	1	100	3
2	100	0	1	50	1	0	20	3
3	100	0	1	100	0	1	100	3
4	100	0	0	100	0	1	100	3
5	100	0	0	100	1	1	100	3
6	100	0	0	50	0	0	20	3

Table 3
Distribution of the parameter settings over the communicating processes of the parallel TSA

Phase of the respective process	Configuration of the communicating processes of the parallel TSA		
	First process (sequential phase-IDs)	Second process (sequential phase-IDs)	Third process (sequential phase-IDs)
First	1	2	3
Second	4	5	6
Third	2	3	4
Fourth	5	6	1

the value 3 is used throughout; it guarantees a high solution quality within acceptable computing times.

As to the parallel TSA, the number of processes was set to four. One of these four processes is excluded from the communication and configured in the same way as the sequential TSA with six phases. For each of the remaining three processes only four phases are used, while, starting with the value 3, the parameter *nbhDecrease* is now decremented per phase. The phases are parameterized in the same way as for the sequential TSA. Table 3 shows how the parameter settings for the phases of the sequential TSA are distributed over the communicating processes.

Remaining conceptual decisions concern the communication model and the handling of foreign solutions. As communication model either a ring or a blackboard can be chosen. For foreign solutions either an unconditional adoption or a conditional adoption can be applied. In order to evaluate the resulting four configuration variants, some computational experiments were carried out. The highest calculated volume utilization differed only slightly between these four variants; the maximum difference amounted to 0.2% of the container volume. Only the best variant, defined by the communication model ring and the unconditioned adoption of foreign solutions, is used for the comparative test described in the following. For the sequential and the parallel TSA, the limit of the computing time is equally set to $maxTime = 600$ s.

5.3. Comparative test

It should be mentioned beforehand, that results calculated by other authors are presented here as reported in the respective literature sources. For the 15 problems from Loh and Nee, computational results are reported for the heuristics from Loh and Nee [13], Ngoi et al. [15], Bischoff et al. [2], Bischoff and Ratcliff [1] and for the GAs from Gehring and Bortfeldt [8] and Bortfeldt and Gehring [5]. Table 4 shows the test results obtained for the 15 problems from Loh and Nee. Note that Loh and Nee [13] use a capacity criterion referred to as "packing density" which overestimates the volume utilization (cf. [1]). As to the quantities in the first column of Table 4, a best value is counted for a method if it achieved the best known volume utilization for a problem instance. Analogously, a global optimum is counted if a

method has stowed all boxes of a problem instance in the container. For the sequential and the parallel TSA, the support of the boxes is set to $minSupport = 100\%$.

In summary, it can be stated that for the problems from Loh and Nee, the TSA obtains better results than the other methods. However, the parallelization does not lead to an improvement here.

For the 700 test problems from Bischoff and Ratcliff, results are available for the heuristics from Bischoff et al. [2], Bischoff and Ratcliff [1] and for the GAs from Gehring and Bortfeldt [8] and Bortfeldt and Gehring [5]. These results are summarized in Table 5 together with those obtained by the sequential and parallel TSA. For the latter two methods, the support of the boxes is always set to $minSupport = 55\%$. For each test case, the number of box types and the mean number of boxes per type for the problems of the test case are given in the first column.

For the sequential and the parallel TSA, Table 6 shows the mean computing times per problem instance calculated over each of the test cases from BR1 to BR7 and over all 700 problems.

The results of this comparative test can be summarized as follows.

- The sequential and the parallel TSA clearly dominate over the other methods with respect to the mean volume utilization. The parallelization of the TSA leads to a mean enhancement of the volume utilization of 0.66% of the container volume. For the standard deviations per test case, the TSA also achieves favourable results. For the seven test cases, the highest standard deviation of the volume utilization calculated over the instances of a test case amounts to 2.25% of the container volume.
- The parallel TSA requires significantly higher computing times per problem instance than the sequential TSA. This is the result of the configuration of the parallel search and especially of the fact, that the transition of a search process to a foreign solution is coupled with an enlargement of the neighbourhoods subjected to the further search. Nevertheless, the mean computing times of the parallel TSA observed for the calculated test cases lie obviously in an acceptable range.
- The results reported for the other methods (cf. the second to the fifth column in Table 5) are based on full support of the stowed boxes. In the interest of fair comparison, additional calculations for the value $minSupport = 100$ were carried out with the sequential and the parallel TSA. The achieved mean volume utilizations amount to 91.6 % for the sequential TSA and 92.2% for the parallel TSA. This means, that the ranking of the methods with respect to the volume utilization is not affected by the required support of the boxes. On the other hand, these experiments demonstrate the extent to which the volume utilization can be improved through the relaxation of the stability constraint (C2). It should be mentioned, however, that the method from Bischoff et al. [2] achieves better results than the TSA with respect to other stability criteria such as, e.g., the number of supporting boxes per stowed box.
- In the case of the TSA the mean volume utilizations calculated per test case are, in general, monotonically decreasing with increasing number of box types. Hence, the TSA achieves the highest volume utilizations for weakly heterogeneous box

Table 4
Numerical results obtained for the 15 problems from Loh and Nee [13]

Quantities	Loh and Nee [13] Packing density	Ngoi et al. [15] Volume utilization	Bischoff et al. [2] Volume utilization	Bischoff and Ratcliff [1] Volume utilization	Gehring and Bortfeldt [8] Volume utilization	Bortfeldt and Gehring [5] Volume utilization	Authors' Sequential TSA Volume utilization	Authors' parallel TSA Volume utilization
Mean value (%)	74.2	69.0	69.5	68.6	70.0	70.1	70.9	70.9
Best values	11	11	10	11	12	13	15	15
Global optima	11	11	10	11	12	13	13	13

Table 5
Numerical results for the 700 problems from Bischoff and Ratcliff [1]

Test case (no. of box types, mean number of boxes per type)	Bischoff et al. [2] Volume utilization [%]	Bischoff and Ratcliff [1] Volume utilization [%]	Gehring and Bortfeldt [8] Volume utilization [%]	Bortfeldt and Gehring [5] Volume utilization [%]	Authors' sequential TSA Volume utilization [%]	Authors' parallel TSA Volume utilization [%]
BR1 (3, 50.1)	81.76	83.79	85.80	87.81	93.23	93.52
BR2 (5, 27.3)	81.70	84.44	87.26	89.40	93.27	93.77
BR3 (8, 16.8)	82.98	83.94	88.10	90.48	92.86	93.58
BR4 (10, 13.3)	82.60	83.71	88.04	90.63	92.40	93.05
BR5 (12, 11.1)	82.76	83.80	87.86	90.73	91.61	92.34
BR6 (15, 8.8)	81.50	82.44	87.85	90.72	90.86	91.72
BR7 (20, 6.5)	80.51	82.01	87.68	90.65	89.65	90.55
All test cases	82.0	83.5	87.5	90.1	92.0	92.7

Table 6
Computation times for the 700 problems from Bischoff and Ratcliff [1]

Test case	Sequential TSA mean computation time [s]	Parallel TSA mean computation time [s]
BR1	3	36
BR2	10	48
BR3	31	97
BR4	48	138
BR5	65	179
BR6	46	150
BR7	60	198
All test cases	38	121

sets. This solution behaviour is caused by the design of the basic heuristic. The attainable utilization depends strongly on the question of whether sufficient numbers of boxes of the same types are available for the forming of solid space-saving blocks. Note that the GA from Bortfeldt and Gehring [5], which is tailored to strongly heterogeneous box sets, shows a complementary behaviour.

Finally, the question of interest is, whether and to what extent the process communication causes a synergetic effect. Therefore, the 700 problems from Bischoff and Ratcliff were solved again using the parallel TSA, but without communication between the concurrently executed processes. For a required support of $minSupport = 55\%$ the mean volume utilization over all 700 problems amounted to 92.5% for the parallel TSA without and to 92.7% with communication, i.e., the synergetic effect caused by the process communication is rather small.

6. Conclusions

In this paper a parallel tabu search algorithm for solving the container loading problem with a single container to be loaded is presented. The parallelization approach follows the concept of multi-search threads with cooperating processes according to Toulouse et al. [20]. According to an extensive comparative test also including heuristics from other authors, high utilizations of the container volume are already obtained with the sequential TSA. A slight improvement of these results could be achieved by the parallelization. The communication between the TSA instances, however, had only a small share in this effect. Similar results are found in the literature. Crainic et al. [6], for example, report on the parallelization of a TSA for solving a warehouse location problem, where the best results were obtained without communication between the concurrently executed processes.

Previous experience with the parallelization of container loading methods gives rise to consider the parallelization of a method as a relevant methodical extension, particularly if other concepts for the improvement of a sequential method are al-

ready exhausted. However, only a limited enhancement of the solution quality can be expected. On the other hand, it is not impossible that a hybrid parallelization approach, including different types of metaheuristics, leads to better results with respect to the enhancement of the solution quality. The well-known fact that the balance between exploration and intensive examination of limited regions of the search space differs significantly with the type of the metaheuristic speaks in favour of this option. A subject of further research on this topic is therefore a hybrid approach for the container loading problem combining instances of a tabu search and a simulated annealing algorithm.

References

[1] E.E. Bischoff, B.S.W. Ratcliff, Issues in the development of approaches to container loading, Omega 23 (1995) 377–390.
[2] E.E. Bischoff, F. Janetz, M.S.W. Ratcliff, Loading pallets with non-identical items, European Journal of Operational Research 84 (1995) 681–692.
[3] A. Bortfeldt, Informierte Graphensuchverfahren und genetische Algorithmen zur Lösung von Containerbeladeproblemen, Dissertation, Freie Universität, Berlin, Verlag Dr. Köster, Berlin, 1995.
[4] A. Bortfeldt, H. Gehring, Ein Tabu Search-Verfahren für Containerbeladeprobleme mit schwach heterogenem Kistenvorrat, OR Spektrum 20 (1998) 237–250.
[5] A. Bortfeldt, H. Gehring, A hybrid genetic algorithm for the container loading problem, European Journal of Operational Research 131 (2001) 143–161.
[6] T.G. Crainic, M. Toulouse, M. Gendreau, Synchronous tabu search parallelization strategies for multicommodity location-allocation with balancing requirements, OR Spektrum 17 (1995) 113–123.
[7] H. Dyckhoff, U. Finke, Cutting and Packing in Production and Distribution, Physica, Heidelberg, 1992.
[8] H. Gehring, A. Bortfeldt, A genetic algorithm for solving the container loading problem, International Transactions in Operational Research 4 (1997) 401–418.
[9] H. Gehring, A. Bortfeldt, Ein verteilt-paralleles Tabu Search-Verfahren für Containerbeladeprobleme mit schwach heterogenem Kistenvorrat, in: P. Kall, H.-J. Lüthi (Eds.), Operations Research Proceedings 1998, Springer, Berlin, 1999, pp. 220–227.
[10] H. Gehring, A. Bortfeldt, A parallel genetic algorithm for solving the container loading problem, International Transactions in Operational Research 9 (2002) 497–511.
[11] H. Gehring, J. Homberger, A parallel two-phase metaheuristic for routing problems with time windows, Asia-Pacific Journal of Operational Research 18 (2001) 35–47.
[12] F. Glover, M. Laguna, Tabu search, in: C.R. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, Oxford, 1993, pp. 70–150.
[13] T.H. Loh, A.Y.C. Nee, A packing algorithm for hexahedral boxes, in: Proceedings of the Conference of Industrial Automation, Singapore, 1992, pp. 115–126.
[14] R. Morabito, M. Arenales, An AND/OR-graph approach to the container loading problem, International Transactions in Operational Research 1 (1994) 59–73.
[15] B.K.A. Ngoi, M.L. Tay, E.S. Chua, Applying spatial representation techniques to the container packing problem, International Journal of Production Research 32 (1994) 111–123.
[16] D. Pisinger, Heuristics for the container loading problem, European Journal of Operational Research 141 (2002) 143–153.
[17] Y. Rochat, E. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.
[18] G. Scheithauer, Algorithms for the container loading problem, in: Operations Research Proceedings, 1991, Springer, Berlin, 1992, pp. 445–452.

[19] M. Sixt, Dreidimensionale Packprobleme. Lösungsverfahren basierend auf den Meta-Heuristiken Simulated Annealing und Tabu-Suche, Europäischer Verlag der Wissenschaften, Frankfurt am Main, 1996.
[20] M. Toulouse, T.G. Crainic, M. Gendreau, Issues in designing parallel and distributed search algorithms for discrete optimization problems, Publication CRT-96-36, Centre de recherche sur les transports, Université de Montréal, Canada, 1996.