Universidad
de La Laguna

A parallel tabu search algorithm for solving the container loading problem

Robabeh Salehi

Universidad de La Laguna

11 de enero de 2016

Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Indices

Intruducion

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

- Orientation constraint
- Stability constraint

Intruducion

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.

The basic heuristic

The lowest module consists of a simple heuristic, called basic heuristic, which serves the complete loading of a container.

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 .

• 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 problem instance.

Heuristic algorithm

The basic heuristic

- The algorithm

(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*.

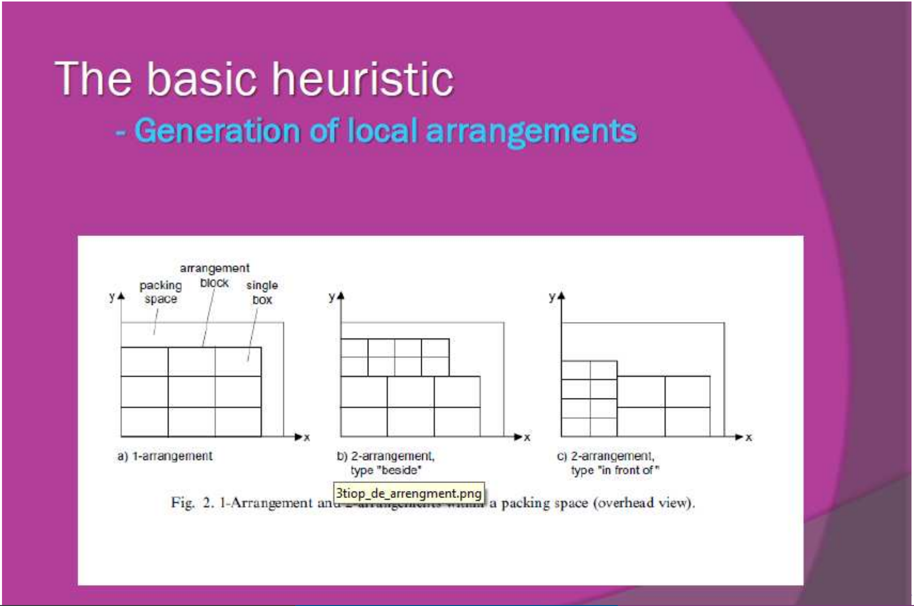
(7) Update the set of residual boxes *BRes*.

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

las formulas.png

(9) Stop.

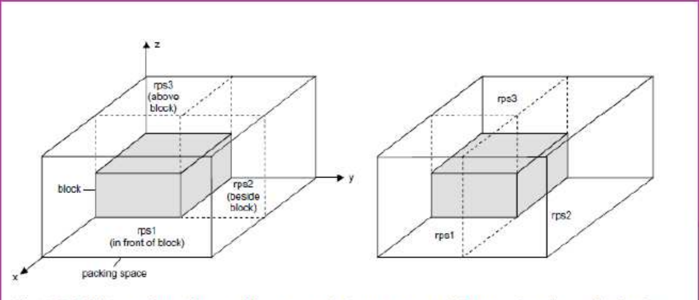
Modelization



The basic heuristic

- 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.



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

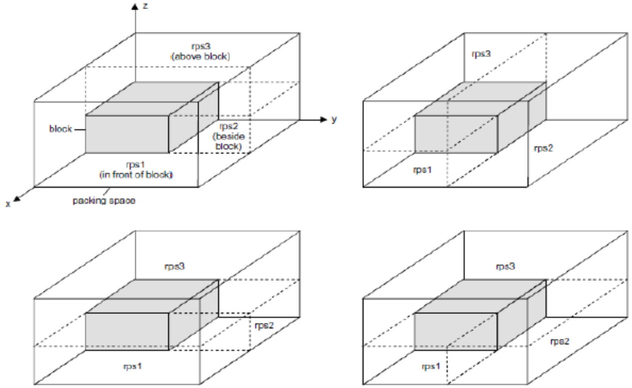
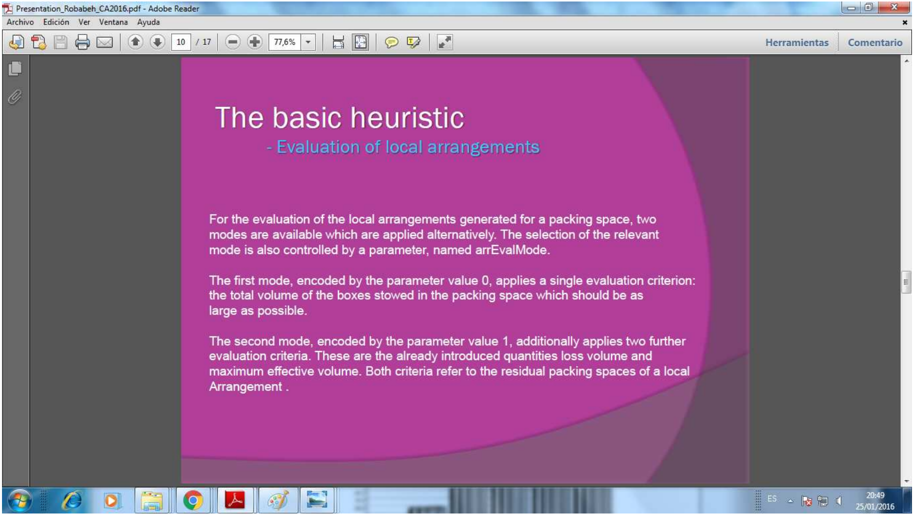


Fig. 4. Additional subdivi a.png ariants for a packing space and a 1-arrangement with overhanging residual packing spaces (rps), resudial packing space.

Local arrangements



The sequential tabu search algorithm

- Encoding of feasible solutions

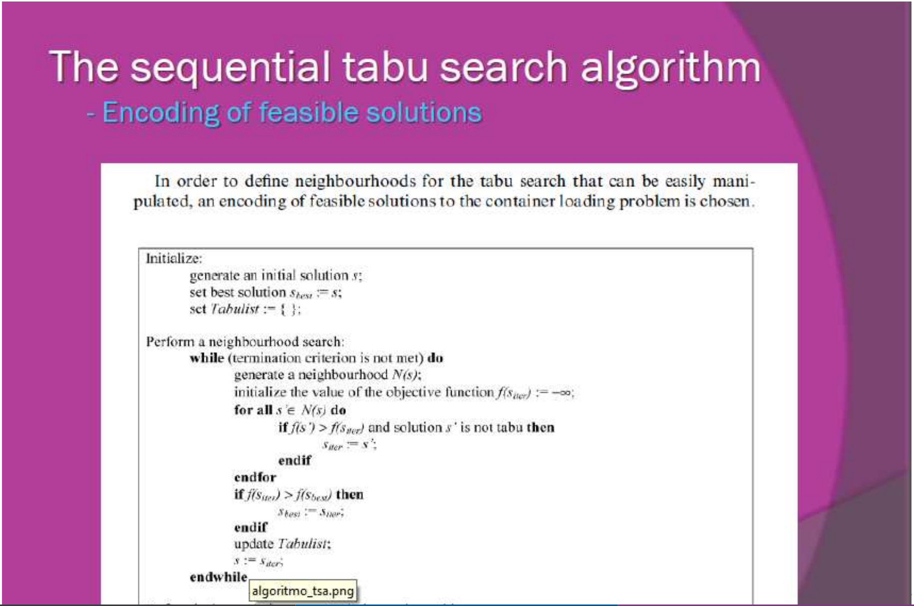
In order to define neighbourhoods for the tabu search that can be easily manipulated, an encoding of feasible solutions to the container loading problem is chosen.

```
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
```

algoritmo_tsa.png

Tabu search algorithm



The sequential tabu search algorithm

- Encoding of feasible solutions

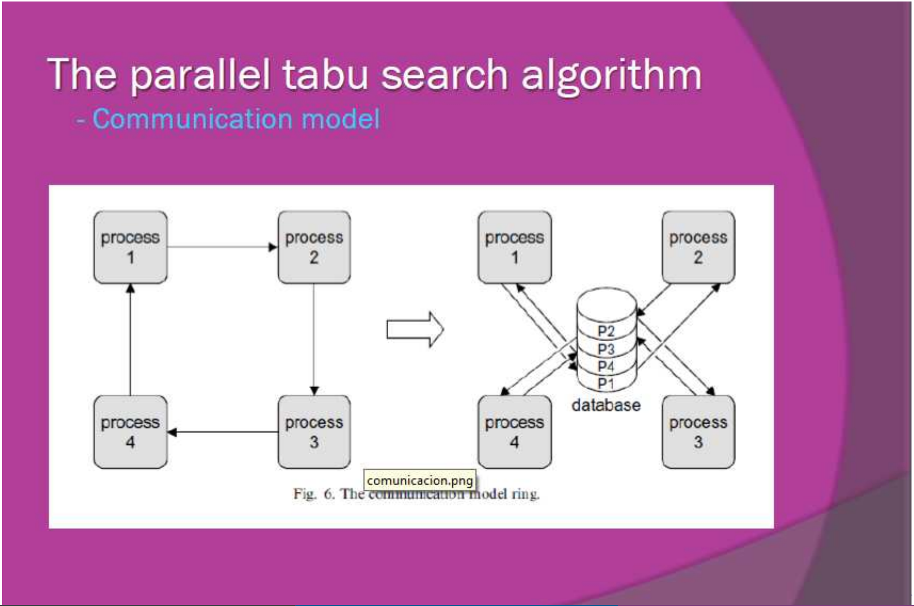
In order to define neighbourhoods for the tabu search that can be easily manipulated, an encoding of feasible solutions to the container loading problem is chosen.

```
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
```

algoritmo_tsa.png

Ring model


The parallel tabu search algorithm
- Communication model
process 1
process 2
process 3
process 4
process 1
process 2
process 4
process 3
P2
P3
P4
P1
database
comunicacion.png
Fig. 6. The model ring.

Exchanging solutions

The parallel tabu search algorithm

- 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.

Blackboard model

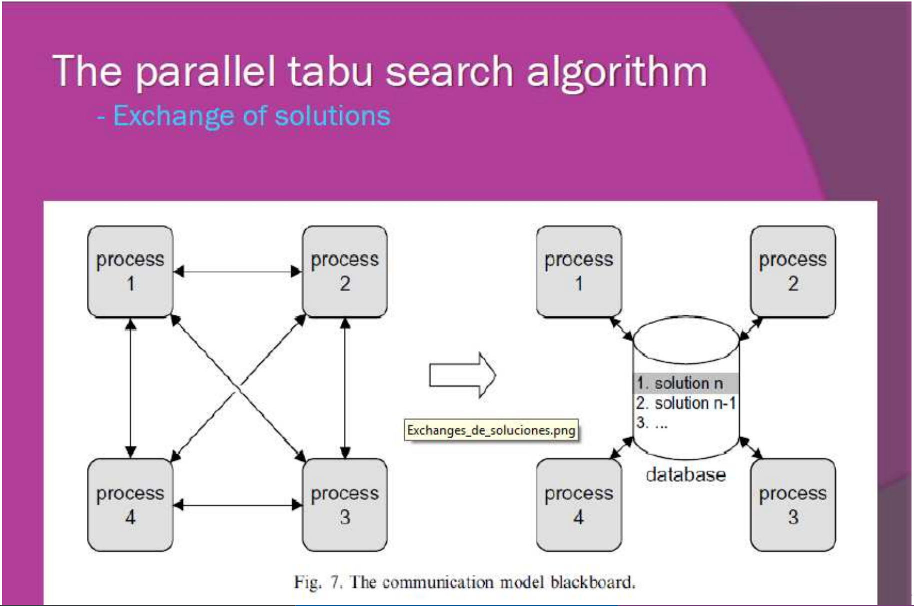


Fig. 7. The communication model blackboard.

Further detailes

The parallel tabu search algorithm

- 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.

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. 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. 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.

Gracias por su

atencion