HEURISTIC-BASED MODEL FOR CONTAINER STACKING PROBLEM

M. Kefi1, O. Korbaa1, K. Ghedira1, P. Yim2

1LI3, Ecole Nationale des Sciences de l'Informatique, Manouba University, 2010 Manouba-2010, Tunis, Tunisia

2 LAGIS, Ecole Centrale de Lille

BP 48, Scientific City, F59651Villeneuve d'Ascq, Lille, France

Abstract

The paper presents a container stacking model that simulates, solves and optimizes the container stacking activity within a fluvial or maritime port. This activity constitutes a real world problem that is omnipresent in logistic and transportation domain. Given container arrivals in a container port terminal, the objective is to assign a slot to each one in a storage area at least cost with respect to pre-defined constraints. The cost is expressed in term of expected relocation movement number. The constraints to respect are stack height, stack number, and departure dates. The aim of the proposed model is to help operative management personnel by giving a stacking plan for each container. To this end, two centralized algorithms were presented: the first is an uninformed search algorithm and the second is an informed search algorithm based on heuristics. By comparing them, it is shown that the latter performs much better than the former thanks to the adopted heuristics.

Keywords:
Transportation, Optimization, Maritime Port, Container Stacking, Heuristics.

1 INTRODUCTION

Recently, seen the large competition between ports, the improvement of customer service became a serious important problem within port container terminals which led to several sub-problems [7] . One of the performance measures is the time spent by vessels in the port quays which is referred by 'berthing time'.

This time is composed of the container unloading/ loading time in major part. In order to reduce the loading time of load, it is necessary to determine adequate slots for arriving containers to be efficiently loaded on corresponding vessels or trucks.

Explicitly, it is about assigning nearly in real time, convenient slots to new arriving containers or disturbed containers when departing of others, among empty ones. This assignment should minimize the expected unproductive/ parasite movements when one wants to place a new container, to bring closer a distant container, or to reach a container being under others container to be loaded on vessel, train or truck. Predefined spatio—temporal constraints like: stack height, stack number and container departure dates should be also respected.

This problem is known in literature as 'Container stacking problem'. It is a NP-complete and an intractable highly combinatorial optimization problem.

Few studies are dealing with this problem. [2] proposed dynamic programming to attain an ideal configuration while minimizing the number of containers to move and the follow-on travelled distance. The problem is divided into two sub-problems: Bay matching and move planning problem, and moving tasks sequencing problem. A mathematical model is used for the resolution of each sub-problem.

It is noted worthy that [6] used genetic algorithms in their study to reduce container transfer and handling times and thereafter the berthing time of ship at port quays.

An interesting work, propounded by [3], dealt with the problem of determination of the best storage slot for containers with the aim of minimizing the number of expected relocation movements in loading operation. They deployed the weight of the container as criteria to define certain priority between the different containers to be stacked in the storage yard.

[1] provided an empiric and comparative survey of the different strategies of container storage in a terminal port according to several parameters.

Another work [4] proposed a cost model for determining the optimal storage position and the optimal number of cranes used for container handling.

A recent work [5] have used dynamic programming algorithm based on different laws of stochastic arrival of barges to solve in real time the problem of unloaded container assignment within fluvial port while minimizing parasite movement number.

These works are based on mathematical and stochastic models to solve the underling problem. They turn out to be very limited seen its complexity and dynamicity.

Indeed, this paper presents two centralized algorithms: the first is an uninformed search algorithm proposed to simulate and solve the container stacking process, and the second is an informed search algorithm using developed heuristics proposed to optimize this process.

The paper is organized as follows: After an outline of the problem settings that have been identified as being important to develop a convenient model for container stacking, it describes exhaustively the proposed basic model using an uninformed search algorithm. Next, it details the extended model which uses an informed search algorithm via specific heuristics. These models are then compared on the basis of randomly instances generation. Finally, concluding remarks are given.

2 PROBLEM SETTINGS

To clarify the problem raised, its parameters, hypotheses, constraints and objectives are presented next.

2.1 Parameters

The following parameters were taken into account in our study:

- A storage yard composed of *M* Slots.
- *N* ISO Containers stocked in stacks.
- P_{max} : the maximal number of stacks in a storage area.
- H_{max}: the maximal height of stacks.
- *initConfig*: an initial arrangement of the *N* containers in the *M* available slots.
- *finConfig*: a final possible arrangement of deranged containers after the departure of some ones. This arrangement must be attained from *initConfig*.

2.2 Hypotheses

The following hypotheses are adopted:

- Container arrival and a departure cannot coincide,
- The departure order of different containers is known, while their arrival order is unknown,
- Only one container arrives at once,
- The containers of different dimensions (1 TEU, 2 TEU etc.) are stacked on two different storage yards,
- A movement consists on removing a container and putting it somewhere,
- Vessels are classified as *A*, *B*, *C*, *D*, *E*, and *F* according to the chronological order of their departure dates which are known in advance,
- Vessels of which the exact departure dates are beyond *F* are called *Y*.
- Vessels of which the departure dates is completely unknown are called *Z*,
- Every container is associated the vessel whereby it is going to leave (*A*, *B*, *C*, *D*, *E*, *F*…*Y* & *Z*) which determines implicitly its departure date *t.* For example, for two different containers *C* and *C'*, if *C* is going to leave on vessel A and C' is going to leave on vessel *B*, *C*, *D*,.. or *Y* then *t* (*C*) <*t* (*C* '),
- The containers are categorized identically to the associated vessels. For example a container is of category A if it is going to leave on vessel *A*,
- One container is prioritized to another container if the category of the former is less than the category of the latter (*A*<*B*<*C*<*D*<*E*<*F*).

2.3 Objectives

The objective of the present study is to determine firstly a feasible sequence of successive unstacking–stacking operations, and then approximate sequence that minimises container rehandles, considering temporal and spatial constraints. To achieve this objective, a basic centralized model that uses an uninformed search algorithm is developed first. Then, an extended centralized model that uses an informed search algorithm based on heuristics is developed. These models will be detailed subsequently.

3 BASIC MODEL

This model is based on an uninformed algorithm in order to simulate and solve the container stacking activity at the moment of departure of prior containers with category A. This algorithm aims determining randomly new slots to disturbed containers of other categories taking into account the arriving containers between two departure times. These are placed first randomly. To explicit this idea, the basic model is described by giving the solving process and the associated algorithm.

3.1 Solving process

It is a question of determining first a list *L* of containers which are departing the first ones, i.e., which category is A (placed in each stack at k^{th} level). Then, unstacking the containers *A* in *L* (sequentially and randomly). After, to determine the list *L'* containing the containers which are on those belonging to *L*. Finally, each container in *L'* is assigned to a new slot chosen randomly, i.e. this container is stacked to a random stack with respect to the maximal height and the total stack number. Once all containers of category A left, the previous steps are applied again to next categories *B*, *C*, *D*, etc. The last step is detailed drawing on Random_assign (see figure 1).

3.2 Algorithm

```
Procedure Random_assign (Pmax, Hmax, initConfig)
Begin
While (cat < catNbr+1) do
  nonEmbarassed_ contAList ←Ø
    Embarassed_contAList =Ø
    For i from 1 to Pmax do
      For j from 1 to Hi do
        If ((matCont (i,j).cat)  = A ) then
          If (j <> Hi) then
            Embarassed_contAList←
            Embarassed_contAList ∪ {matCont(i, j))}
          Else
            nonEmbarassed_contAList←
            nonEmbarassed_ contAList  ∪ {matCont(i, j))}
ContA_depart (nonEmbarassed_ contAList)
While |Embarassed_contAList| <>0) do
 depStack←Random_unstacking(Embarassed_contAList)
  For i from k+1 to Hl do
    destStack←Random_new_slot_determination ( )
    Push(i, destStack )
    mvtNbr = mvtNbr +1;
ContA_depart ()
Update(matCont) ;
End
```

Figure 1: Random assignment of disturbed and arriving containers procedure.

4 EXTENDED MODEL

4.1 Description

With regard to the basic model, the extended model implies heuristics in both departure of containers of category *A*, and determining the most adequate slot for disturbed containers taking into account also arriving containers. This extension is based on an informed search algorithm detailed drawing on Guided_assign procedure (see figure 2). This algorithm consists first in determining a list *L* of containers of category *A*. Then, unstacking these containers sequentially in guided way using heuristics (see figure 3). After, determining the list *L'* containing the containers which are on containers in *L*. Finally, each container in *L*' is assigned to a new adequate slot on the basis of heuristics (see figure 4), i.e. this container is arranged in the stack that minimizes the number of unproductive movements with respect to the maximal height and the total stack number. Once all containers *A* left, the previous steps are applied again to the containers of categories *B*, *C*, *D*, etc.

4.2 Definitions

In this extended model, the following definitions are adopted:

- A container *C1* is embarrassing another container *C2* if *C1* is placed on *C2* and the category of *C1* is greater than the category of *C2*.
- H_i : the height of the stack *i*.
- $f_{i,cat}$: the number of containers in stack *i* that are embarrassing and blocking a container of category *cat*.
- $Occ_{i,cat}$: the number of occurrences of the container of category *cat* in the stack *i*.
- $Dist_{i,cont}$: it is equal to the absolute value of the difference between the categories of the container which corresponds to the heap of the stack *i*, and the embarrassing container *cont* which will be removed to a new stack. The container *cont* can be a disturbed container or an arriving container between two successive departure times.

4.3 Algorithm

```
Procedure Guided_assign (Pmax, Hmax, initConfig )
Begin
While (cat < catNbr+1) do
  nonEmbarassed_ contAList ←Ø
    Embarassed_contAList =Ø
    For i from 1 to Pmax do
      For j from 1 to Hi do
        If ((matCont (i,j).cat)  = A ) then
          If (j <> Hi) then
            Embarassed_contAList←
            Embarassed_contAList ∪ {matCont(i, j))}
          Else
            nonEmbarassed_contAList←
            nonEmbarassed_ contAList  ∪ {matCont(i,
          j))}
ContA_depart (nonEmbarassed_ contAList)
While |Embarassed_contAList| <>0) do

depStack←Guided_unstacking(Embarassed_contAList)
  For i from k+1 to Hi do
    destStack←Guided_new_slot_determination ( )
    Push(i, destStack )
    mvtNbr = mvtNbr +1;
ContA_depart ()
Update(matCont) ;
End
```

Figure 2: Guided assignment of disturbed and arriving containers procedure.

4.4 Implied Heuristics

As invoked above, some heuristics are adopted and used in the determination of a departure sequence of containers A. Then, the assignment of new slots to the disturbed containers and arriving containers, with respect to spatio-temporal constraints while minimizing unproductive movement number.

Determination of departure sequence

First, Among all stacks containing embarrassed containers of category *A*, the first departure is accorded to the container *A* belonging to the stack verifying Min ($f_{i,A}$) to minimize the number of parasite movements. Then, if more than one container satisfy this criterion, the retained container corresponds to that belonging to the stack checking Max $Occ_{i,A}$ in order to free the maximum of slots. Finally, if more than one container satisfy this criterion, the lexicographic order of stacks is applied (see figure 3).

Guided_Unstacking Function (*contList*)
Begin
depStack← *0*
dep_stackList ← *depStack*
for *i* from 1 to P_{max} do
  Embarassing_contNbr← $f_{i,A}$
minDep_stackList ←Min($f_{i,A}$)
If |*minDep_stackList* | >1 then
  for *j* from 1 to Length (*minDep_stackList*) do
    Occ← $Occ_{j,A}$
  maxMinDep_stackList ← Max ($Occ_{j,A}$)
  If |*maxMinDep_stackList*| >1 then
    depStack←lexOrder(*maxMinDep_stackList*)
End

Figure 3: Guided Unstacking embarrassed containers function.

Disturbed and arriving container assignment

The disturbed containers i.e., the containers that embarrassed containers A, and the arriving containers are stacked/ assigned to the stack that satisfy these criteria in decreasing priority: First Min($Dist_{i,cont}$) to minimize the number of shifts, then Max (H_i) to maximize the available stack exploitation. Finally, the lexicographic order. (see figure 4).

Guided_new_slot_determination Function (*embaras_contList*)
Begin
destStack ← *0*
dest_stackList ← *destStack*
B=true
For *i* from 1 to P_{max} do
  Dist ← $Dist_{i,Cont}$
minDest_stackList ← Min($Dist_{i,Cont}$)
If |*minDest_stackList* | >1 then
  for *j* from 1 to Length (*minDest_stackList*) do
    H← H_j
  maxMinDest_stackList ← Max (H_j)
  If |*maxMinDest_stackList* |>1 then
    destStack←lexOrder(*maxMinDest_stackList*)
End

Figure 4: Guided assignment of embarrassing containers function.

5 EXPERIMENTAL RESULTS

To evaluate the proposed basic and extended models, JBuilder environment is used in implementing both the uninformed search algorithm and the informed search algorithm.

The following graphic (see figure 5) shows plots that compare these two algorithms. The total number of unproductive movements is represented on axis Y in function of random-generated instances mentioned on Axis X. Each instance signifies an initial configuration with a certain number of containers. It is noted that the numerical results correspond to 10-run average for each generated instance.

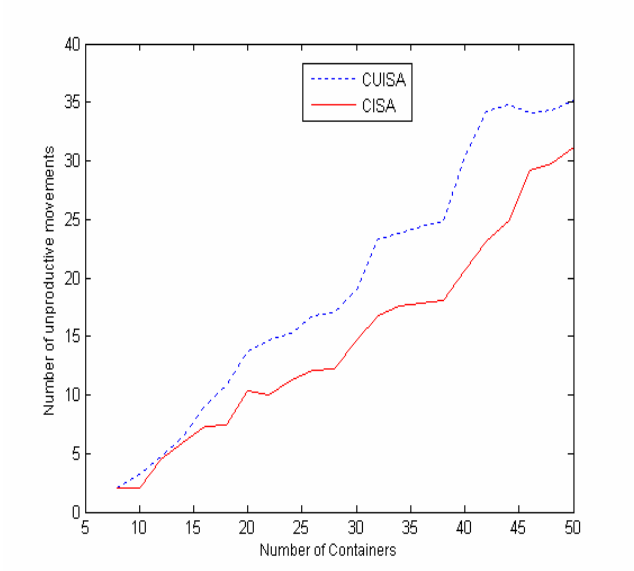

Figure 5: Comparison between the Centralized UnInformed Search Algorithm CUISA and the Centralized Informed Search Algorithm CISA «Centralized Informed Search Algorithm ".

Then, as shown, a significant improvement is noted particularly when the number of containers is increasing.

6 CONCLUSION

This paper dealt with the container stacking activity within maritime or fluvial ports. Two proposed models are described and developed: a basic model based on an uninformed search algorithm, and an extended model based on an informed search algorithm. The basic model allows simulating and solving the container stacking process. While, the extended model allows to optimize this process.

This paper provides empirical and comparative study that compares the two proposed models, and shows that the latter outperforms considerably the former via involving heuristics during the solving procedure. It supports the efficiency of using heuristics search in combinatorial optimization problem solving such as the Container stacking Problem.

7 ACKNOWLEDGMENTS

The engineer “Ayoub Mejri” at the National School of Computing Sciences of Tunisia is gratefully thanked for making the implemention of the proposed models.

8 REFERENCES

[1] Duinkerken, M.B., Evers, J.-J.M., Ottjes, J. A., 2001, A simulation model for integrating quay transport and stacking policies automated containers terminals, Proc. of 15th European Simulation Multi-conference, Prague.

[2] Kim, K.H., Bae, J.W., 1998, Re-marshaling export containers in port container terminals, C&IE, 35, 655-658

[3] Kim, K.H., Park, Y.M., Ryu, K.R., 2000, Deriving decision rules to locate export containers in container yards, EJOR, 124, 89-101

[4] Kim, K.H., Kim, H.B., 2002, The optimal sizing of the storage space and handling facilities for import containers, TR, 36, 821-835.

[5] Korbaa, O., Yim, P., 2004, Container Assignment to Stock in a Fluvial Port, S. M. C International Conf., The Netherlands.

[6] Kozan, E., Preston, P., 1999, Genetic algorithms to schedule container transfers at multimodal terminals, International Transactions in Operational Research, 6, 311-329

[7] Vis, I.F.A., De Koster, R., 2003, Transshipment of containers at a container terminal: an overview, EJOR, 147, 1-16.