# A Branch-and-Cut Algorithm to Solve the Container Storage Problem

Ndèye Fatma Ndiaye
University of Le Havre
Le Havre, France
e-mail: farlou@live.fr

Adnan Yassine
Superior institute of logistic studies
Le Havre, France
e-mail: adnan.yassine@univ-lehavre.fr

Ibrahima Diarrassouba
University Institute of Technology
Le Havre, France
e-mail: diarrasi@univ-lehavre.fr

*Abstract*— **We study the Container Storage Problem in port terminal (CSP), which consists to effectively manage the storage space so as to increase the productivity of port. When a ship arrives, the inbound containers are unloaded by Quay Cranes (QC) and then placed on quays. So, they are collected by Straddle Carriers (SC). Each is able to carry one container at a time, and store it in its storage location. In order to reduce the waiting times of ships, we propose a mathematical model which minimizes the total distance traveled by SC between quays and container yards. In this paper, we take into account additional constraints which are not considered previously. We also propose an effective branch-and-cut algorithm (BC-CSP), which is an optimal resolution method. Performed simulations prove the effectiveness of our algorithm.**

***Keywords-Container Storage Problem; complexity; branch-and-bound; CPLEX.***

## I. INTRODUCTION

In a seaport, the container terminal manages all actions concerning containers. Generally, three types of containers are distinguished: outbound, inbound, and transshipment containers. All these containers are temporarily stacked in the container yard, before leaving the port. Outbound containers are brought by External Trucks (ET), also picked up by SC which store them in their storage location, and then loaded onto vessels. Inbound containers are unloaded from vessels by QC, transported to their storage locations by SC, and then recuperated later by ET. Nowadays, the competition between ports is very high. Therefore, each of them tries to improve continuously the quality of its service in order to attract more customers. The most important criteria to measure service level include the waiting time of ET which collect inbound containers. In fact, when an ET arrives at port and claims a specific container, it waits during all the time required to retrieve it. If the desired container is under others, it may be necessary to move these containers in first. This kind of movements, named reshuffles, are unproductive and require so much time. Therefore, it is very important to optimally store containers so as to avoid them. Another important criterion to measure the quality of service is the time required to unload ships. The importance of this factor is justified by the fact that it is more beneficial for both the port and the customers to shorten the stay of vessels. First, it is better for the port to quickly free the berths in order to allocate them to others incoming vessels. Second, generally, ship-owners rent vessels; therefore, they tend to minimize the berthing durations in order to reduce rental costs. These two issues are addressed in this paper. We consider a modern container terminal, which uses SC instead of Internal Trucks (IT). The advantage of a SC is that it is able to lift and to store a container itself; therefore it is not necessary to use Yard Cranes (YC). A storage yard is composed of several blocks. In order to allow good circulation of SC, each block is composed of bays which are separated by small spaces. In every bay, there are stacks wherein containers are stored. A stack must have a height inferior or equal to the limit fixed by the port authorities. Fig. 1 shows an example of block wherein circulate straddle carriers.

Fig. 1 Straddle carriers circulating in a containers yard

We propose in this paper an effective method to solve the container storage problem. For this, we propose a new mathematical model which determines the optimum storage plan and minimizes the time required to transport containers between quays and storage areas. For the numerical solution

 ISBN: 978-1-61208-319-3 

of the model, we propose an effective branch and cut algorithm (BC-CSP), which is an exact method.

The remainder of the paper is organized as follows: a literature review is given in the second part. A detailed description of the addressed problem is exposed in the third part. The complexity of the problem is discussed in the fourth section, while the proposed mathematical model is explained in the fifth section. The branch-and-cut algorithm is itemized in the sixth section; the numerical results are presented in the seventh section. Our conclusion is given in the eighth section.

## II. LITERATURE REVIEW

There are many papers addressing the storage of outbound containers than inbound containers. However, there are some papers which consider both simultaneously. Zhang et al. [2] considered in addition to these two categories, those which are in transition, that means containers which are unloaded from some vessels and are waiting for being loaded onto others. They used the rolling-horizon approach [2] to solve the storage space allocation problem. For each planning horizon, they solved the problem in two steps formulated as mathematical programs. In the first step, they determined the total number of containers assigned to each block at a period so that the workloads of loading and unloading of each vessel are balanced. Then, in the second step they determined the number of containers associated to every vessel in order to minimize the total distance traveled to transport these containers from quays to the storage blocks. Bazzazi et al. [3] proposed a genetic algorithm to solve an extended version of the Storage Space Allocation Problem (SSAP). It consists to allocate temporarily locations to the inbound and outbound containers in the storage yard according to their types (regular, empty, and refrigerated). They aimed to balance the workloads between blocks with the goal to minimize the time required to store or to retrieve containers. Park and Seo [4] dealt the Planar Storage Location Assignment Problem (PSLAP), in which only planar movements are allowed. The purpose of PSLAP is to store inbound and outbound containers so as to minimize the number of moving obstructive objects. The authors made a mathematical formulation of PSLAP and proposed a genetic algorithm to solve it. Lee et al. [5] combined the truck scheduling and the storage allocation problems. They considered inbound and outbound containers, and tend to minimize the weighted sum of total delay of requests and the total travel time of yard trucks. For numerical resolution, they proposed a hybrid insertion algorithm. Kozan and Preston [6] developed an iterative search algorithm using a transfer model and an assignment model. At first, the algorithm determined cyclically the optimum storage locations for inbound and outbound containers, and secondly, it found the corresponding handling schedule. They solved the problem using a genetic algorithm, a tabou search algorithm and a hybrid algorithm.

Concerning inbound containers, most of papers deal with the management of reshuffles. Sauri and Martin [7] proposed three different strategies to store inbound containers. The purpose of their work is to determine the best strategy which minimizes re-handles in an import container yard. For this, they developed a mathematical model based on probabilistic distribution functions to evaluate the number of reshuffles. Kim and Kim [8] considered a segregation strategy to store inbound containers. This method does not allow placing newly arriving containers over those which arrived earlier. Therefore, storage spaces are allocated to each vessel so as to minimize the number of reshuffles expected during the loading operations. Jinxin et al. [9] proposed an integer programming model, which addressed the trucks scheduling and the storage of inbound containers. They minimized at the same time the number of congestions, the waiting time of trucks, and the unloading time of containers. The authors designed a genetic algorithm to solve the model, and another heuristic algorithm which gave them best results. Yu and Qi [10] treated the storage problem of inbounds containers in a modern automatic container terminal. They aimed to minimize reshuffles in two steps. For this, they first resolved the block space allocation problem for newly arriving inbound containers, and then after the retrieving of some containers they tackled the re-marshaling processes in order to re-organize the block space allocation. They suggested three mathematical models of storage containers, the first is a non-segregation model, the second is a single-period segregation model, and the last is a multiple-period segregation model. They conceived a convex cost network flow algorithm for the first and the second models, and a dynamic programming for the third. They found that the re-marshaling problem is NP-hard and then designed a heuristic algorithm to solve it. We considered in [1] a container terminal wherein reshuffles are not allowed. We proposed a new mathematical model to allocate storage spaces to inbound containers in such a way that no reshuffles will be necessary to retrieve them later. We designed a hybrid algorithm including genetic algorithm and simulated annealing to solve it.

In most container terminals, the departure times of inbound containers are unknown. K. H. Kim and K. Y. Kim considered in [11] a container terminal in which there is limited free time storage for inbound containers, beyond which customers have to pay storage costs per time unity. The authors proposed a mathematical model to find the optimal price schedule.

Papers dealing with the storage problem of outbound containers have generally different goals. Preston and Kozan [12] proposed a Container Location Model (CLM) to store outbound containers in such a way that the time service of container ships is minimal. They designed a genetic algorithm for the numerical resolution. Kim et al. [13] developed a dynamic programming model to determine the storage locations for outbound containers according to their weight. They minimized the number of relocations expected during the ships loading. They also made a decision tree using the set of optimal solutions to support real-time decisions. Chen and Lu [14] addressed in two steps the storage space allocation problem of outbound containers. In the first step, they used a mixed integer programming model to calculate the number of yard bays and the number of

 ISBN: 978-1-61208-319-3 

locations in each of them. So, in the second step, they determined for each container the exact location where it must be stored. Woo and Kim [15] proposed a method to allocate storage spaces to groups of outbound containers. They reserved for each group of containers having the same attributes, a collection of adjacent stacks. At the end, the authors proposed a method to determine the necessary size of the storage space expected for all the outbound containers. Kim and Park [16] gave two linear mathematical models to store outbound containers. In the first, they considered a direct transfer, and so, in the second, they dealt with an indirect transfer system. They designed two heuristic algorithms to solve these models. The one is based on the duration-of-stay of containers, and then, they used the sub-gradient optimization technique in the other.

Among the few papers dealing only transshipment containers include that of Nishimura et al. [17]. They developed an optimization model to store temporarily transshipment containers in the storage yard, and proposed a heuristic based on lagrangian relaxation method for the numerical resolution.

## III. CONTEXT

When a container ship arrives at port, QCs unload containers and place them on quays. So, they are picked up by SCs, which carry and store them in the container yard. The firsts containers which are placed on quays are the first picked up. In order to avoid congestion on quays, which could increase the time required to unload ships, we minimize the total distance traveled by SCs between quays and the container yard. In this study, we consider the following five main hypotheses:

(1) Reshuffles are not allowed,
(2) In each stack, containers are arranged according to:
  (2.a) the same order that they are unloaded from ships,
  (2.b) and the descending order of their departure time,
(3) In a stack, all containers have same dimensions,
(4)We take into account containers which are already present in the storage areas before the start of the new storage period,
(5) We respect the maximum capacity of each stack.

Excepted (2.a), these hypotheses are considered in [1].

## IV. COMPLEXITY OF THE PROBLEM

In this section, we study the complexity of the CSP. In particular, we show that it is equivalent to the Bounded Coloring Problem (BCP); therefore, is NP-hard in the general case.

### A. Some reminders about the BCP

Let us begin by recalling some concepts and definitions that will be useful for the following.

*1) Preliminary notions*: Let *G(V,E)* be an undirected graph, *V* is the set of vertices and *E* is the set of edges.

*G* is a *comparability graph* if and only if there are a sequence of vertices $v_1,...,v_n$ of *V* such that for each *(p, q, r)* checking $1 < p < q < r < n$, if $(v_p, v_q) \in E$ and $(v_q, v_r) \in E$ then $(v_p, v_r) \in E$.

A *co-comparability* graph is the complement of a comparability graph.

An undirected graph *G = (V,E)* is a permutation graph if and only if there are a sequence of vertices $v_1,...,v_n$ of *V* and a permutation $\sigma$ of the vertices such that for all *i* and *j* satisfying $1 \le i \le j \le n$, $(v_i, v_j) \in E$ if and only if $\sigma(i) \ge \sigma(j)$.

*Theorem 1*: A graph *G* is a permutation graph if and only if *G* and its complement are comparability graphs [18].

*2) The bounded coloring problem:* Given an undirected graph *G = (V,E)*, a set of s colors $l_1,...,l_s$, an integer *H* and a vector that gives the weight of assigning a color $l_i$ to a vertex of the graph. The bounded coloring problem with minimum weight consists to determine a minimum weight coloring of *G* using at most *s* colors in such a way that a color is assigned to at most *H* vertices.

*Theorem 2*: [19] The bounded coloring problem with minimum weight is NP-hard for the class of permutation graphs for all $H \ge 6$.

### B. Equivalence between the CSP and the BCP

We show that the CSP is NP-hard. For this, we introduce an undirected graph *G(N,O,T) = (V,E)* constructed from an instance of the CSP, where *N* is the set of containers and *O* and *T* are vectors, which give respectively the unloading order and the departure times of each container. The graph *G* is constructed as follows. A vertex of the graph corresponds to a container. To simplify notation, the index *k* is used to denote both a container and the vertex of the graph which corresponds to it. There is an edge between two vertices *k* and *k'* if and only if $O_k < O_{k'}$ and $T_k < T_{k'}$. We have the following lemma.

*Lemma 1*: The graph *G(N,O,T)* obtained from a instance of CSP is a permutation graph.

*Proof*: To prove that the graph *G(N,O,T)* is a permutation graph, it suffices to show that it is a comparability graph as well as its complement (see Theorem 1).

First, we show that *G(N,O,T)* is a comparability graph. The vertices are ordered according to the same order that the unloading of the corresponding containers from ships. If two containers *k* and *k'* are unloaded from ships at the same time (that is to say if $O_k = O_{k'}$), then the vertices *k* and *k'* are ordered in the ascending order of their departure times. If

 ISBN: 978-1-61208-319-3 

$O_k = O_{k'}$ and $T_k = T_{k'}$ then the vertices are ordered in the lexicographical order. Without loss of generality, we consider that the vertices are numbered in the order that is previously determined. Now, consider any three vertices *k*, *k'* and *k''* of the graph such that $k < k' < k'', (k,k') \in E$ and $(k',k'') \in E.$ We will prove that necessarily $(k,k'') \in E$. As $(k,k') \in E$ and $(k',k'') \in E$, we have $O_k < O_{k'}$ and $T_k < T_{k'}$, and we have also $O_{k'} < O_{k''}$ and $T_{k'} < T_{k''}$. We thus obtain that $O_k < O_{k'} < O_{k''}$ and $T_k < T_{k'} < T_{k''}$, which implies that the graph *G(N,O,T)* has an edge between vertices *k* and *k''*. So *G(N,O,T)* is a comparability graph.

Now, we will prove that the complement of *G(N,O,T)*, denoted $\overline{G}(N,O,T)$ is also a comparability graph. First, note that there is an edge between two vertices *k* and *k'* of $\overline{\overline{G}}(N,O,T)$ if and only if there is no edge in *G(N,O,T)* between *k* and *k'* in other words $O_k < O_{k'}$ and $T_k > T_{k'}$. The vertices of $\overline{G}$ are ordered in the same order as those of G. As before, for any three vertices *k*, *k'*, and *k''* of the graph $\overline{G}(N,O,T)$ such that $k < k' < k'', (k,k') \in E$ and $(k',k'') \in E,$ we have $O_k < O_{k'} < O_{k''}$ and $T_k > T_{k'} > T_{k''}$. So, $O_k < O_{k''}$ and $T_k > T_{k''}$, and then there is an edge between *k* and *k''* in $\overline{G}(N,O,T)$. Therefore, $\overline{G}(N,O,T)$ is also a comparability graph.

Now, it is easy to see that a solution of the container storage problem is a solution of the corresponding bounded coloring problem. In fact, a similar result is given in [20]. Consider an instance *ICSP = (N,O,T,* $N_p$ *,H,r,R,d)* of the CSP and the graph *G(N,O,T)* associated. Now, consider an H-coloring of *G(N,O,T)* that has s colors. Each color of the bounded coloring problem is matched to a stack of the CSP. Indeed, as all vertices having the same color form a stable set, in other words they are not connected by any edge, therefore any two containers corresponding to two vertices of this stable set satisfy these two inequalities $O_k < O_{k'}$ and $T_k \geq T_{k'}$. The unloading order as well as the departure times of containers corresponding to the vertices of a stable set are compatible; thereby, they can be stored in a same stack if it has enough empty slots. In addition, there are at most *H* vertices in this stable set. So, the number of containers assigned to the corresponding stack is inferior or equal to *H*. Therefore, an H-coloring corresponds to a valid assignment for the CSP. Similarly, it is easy to see that a solution of the CSP is a solution of the H-bounded coloring problem in the graph *G(N,O,T)*. We have the following lemma.

*Lemma 2:* Let *ICSP = (N,O,T,* $N_p$ *,H,r,R,d)* an instance of the container storage problem. The CSP has a solution for this instance if and only if the bounded H-coloring problem on the graph *G(N,O,T)* has a solution.

We now give the main result of this section.

*Theorem 3*: The container storage problem is equivalent to the bounded coloring problem with minimum weight.

*Proof:* To establish this result, we prove that an instance of the CSP is equivalent to an instance of the BCP and vice versa. Let *ICSP = (N,O,T,* $N_p$ *,H,r,R,d)* an instance of the storage container problem and *G(N,O,T)* the permutation graph associated. Consider *IBCP = (G(N,O,T),H,* $N_p$ *,d)* an instance defined on the graph *G(N,O,T)*, where *Np* is the number of colors, *H* is the bound, and *d* a matrix containing the weights. According to the Lemma 2 a solution of the CSP is a solution of the BCP, and similarly a stack of CSP corresponds to a color of BCP and vice versa. It follows then that the cost $d_p^k$ of assigning a container $k \in N$ to the stack $p \in N_p$, is the same as the assignment of the vertex *k* to the color corresponding to the stack *p*. So, the cost of H-coloring in the graph *G(N,O,T)* is the same as the cost of the solution of the corresponding CSP and vice versa. Therefore, we can find the optimal solution of the CSP if and only if we find the optimal solution of BCP.

According to the Theorem 2 the bounded coloring problem is NP-hard for the class of permutation graphs if $H \geq 6$. It therefore follows from Theorem 3 that the CSP is NP-hard if $H \geq 6$.

*Corollary 1*: The container storage problem is NP-hard if the maximum capacity of every stack is superior or equal to six.

## V. MATHEMATICAL MODELING

In the mathematical model, we use the following indices:

*p*: stack,
*k*: container.

The data of the problem are:

*N* : number of containers,

$N_p$ : number of stacks,

$c_p$ : number of empty slots in the stack p,

$r_p$ : type of container which can be placed in the stack *p*,

$t_p$ : departure time of the container which was on the top of the stack p at the begin of the new storage period.

$R_k$ : type of the container k,

$T_k$ : departure time of the container k,

$O_k$ : unloading order of the container k from ships.

$d_p^k$ : traveled distance to transport the container k from quay to stack p,

*G(V,E)*: a graph, where *V* is the set of vertices and *E* the set of edges. Every vertex represents a container, and |V| = N. There is an edge between two vertices $v_k$ and $v_{k'}$ if and only if $T_k < T_{k'}$ and $O_k < O_{k'}$, this means that container *k* and *k'* can't be assigned to a same stack.

The decision variables are defined as follows:

$$x_p^k = \begin{cases} 1 & \text{if container } k \text{ is assigned to stack } p \\ 0 & \text{otherwise} \end{cases}$$

We propose the following mathematical model:

$$Minimize \sum_{k=1}^{N}\sum_{p=1}^{N_p} d_p^k x_p^k \quad (1)$$

$$\sum_{p=1}^{N_p} x_p^k = 1, \quad \forall k = 1,...,N \quad (2)$$

$$x_p^k + x_p^{k'} \le 1, \quad \forall (k,k') \in E, p = 1,...,N_p \quad (3)$$

$$\sum_{k=1}^{N} x_p^k \le c_p, \quad \forall p = 1,...,N_p \quad (4)$$

$$\sum_{1 \le k \le N, T_k > t_p or R_k \ne r_p} x_p^k = 0, \quad \forall p = 1,...,N_p \quad (5)$$

$$x_p^k \in \{0,1\} \quad \forall p = 1,...,N_p, k = 1,...,N \quad (6)$$

The objective function (1) minimizes the total distance traveled between ships and the container yard. Constraints (2) require that each container is assigned to a single stack. Constraints (3) ensure that the containers of each stack are arranged following the same order that they were unloaded from ships, and the decreasing order of their departure times. Constraints (4) enforce the stack capacity. Constraints (5) secure the compatibility between containers and stacks.

Let k a vertex of the graph ($1 \le k \le N$), *N(k)* the set of its neighbors, *p'* a stack ($1 \le p' \le N_p$). Constraints (3) lead to the following neighborhood inequality as in [21].

$$\sum_{k' \in N(k)} x_{p'}^{k'} + |N(k)| x_{p'}^k \le |N(k)| \quad (7)$$

*Proposition 1:* For an integer solution, the inequalities (3) and (7) are equivalents.

*Proof:* It suffices to prove that (7) implies (3), because the reverse is highlighted by the definition of (7).

Since $x_{p'}^k$ is a binary variable, then it can be equal to either 0 or 1.

- If $x_{p'}^k = 1$ then $\sum_{k' \in N(k)} x_{p'}^{k'} = 0$. Therefore, for all *k'* neighbor of *k*, we have $x_{p'}^{k'} = 0$. Thereby $x_{p'}^k + x_{p'}^{k'} = 1, \forall k' \in N(k)$.
- If $x_{p'}^k = 0$ then $\sum_{k' \in N(k)} x_{p'}^{k'} \le |N(k)|$ which means that for all *k'* belonging to *N(k)*, $x_{p'}^{k'}$ can be equal to either 0 or 1. Thus, $x_{p'}^k + x_{p'}^{k'} \le 1, \forall k' \in N(k)$.

## VI. BRANCH-AND-CUT ALGORITHM

The branch-and-cut is an improvement of the branch-and-bound, which is an exact resolution method. Each of these two methods uses a search tree to explore the solution space. To do this, the search space is divided into smaller subsets, each representing a node of the search tree. So, the problem is solved by considering one by one all subsets. This strategy is called *divide and conquer*.

To build the search tree, we first create the root node; it corresponds to the released problem. Other nodes of the tree are obtained by making connections.

In the branch-and-cut, unlike the branch-and-bound, at each node of the search tree, some constraints called *valid inequalities* are added to the released problem so as to improve the solution.

### A. Relaxation of the problem

After the relaxation of integrity constraints (6), we find that the total number of constraints of the mathematical model remains great. Therefore, since the adjacency constraints (3) are equivalent to the neighborhood inequalities (7), so we delete them from the model knowing that the admissibility of solutions will be ensured by the gradual addition of valid inequalities along the branch-and-cut algorithm.

### B. Preprocessing

The number of variables increases depending on the number of stacks ($N_p$) and containers *(N)*. In the case where all stacks were empty at the beginning of the storage period, we can reduce the number of variables. We consider that all containers are equidistant to the stacks. Knowing

 ISBN: 978-1-61208-319-3 

that, generally, $N_p \geq N$, we only use the $N$ stacks which are more near to quays. This allows to significantly reduce the number of variables and to speed up the computation.

### C. Upper bound

To find an upper bound, we solve the bounded vertex coloring problem on the graph defined in section V. Each color corresponds to a stack. We use a heuristic algorithm which colors vertices one by one following the descending order of their number of uncolored neighbors. For each vertex, it chooses among the admissible colors the one that fits to the nearest stack. The eligible colors are those not assigned to a vertex which is a neighbor of the considering vertex, and correspond to the stacks which are not full. Whenever a vertex is colored, the number of empty slot of the stack corresponding to the used color is reduced.

### D. Branchings rules

We use the classical branching rule. At each node of the search tree, we create two branches by rounding the largest fractional variable. Let $x_p^k$ this variable. We put $x_p^k = 0$ in a branch; it means that container $k$ will not be assigned to stack $p$ in this branch. Then, in the other branch, we put $x_p^k = 1$, which means that container $k$ will inevitably be assigned to stack $p$ in this branch.

### E. Separation method

At each node of the search tree, before creating branches, we use a simple heuristic algorithm to look for neighborhood inequalities which are violated. To do this, we treat one by one all variable which is superior to 0.5 in the optimal solution of the current node. Let $x_p^k$ one of these variables and $S$ an integer initialized to zero. We calculate the number $|N(k)|$ of neighbors of the vertex $k$. Then, we add to $S$ the value of $x_p^k$ multiplied by $|N(k)|$. And we seek all variables $x_{p'}^{k'}$ such that $k$ and $k'$ are neighbors and $p=p'$, and we add the sum of their values to $S$. If $S > |N(k)|$ then there is a violated inequality therefore we add to the sub-problem a constraint to avoid this.

### F. Description of the algorithm

1: We begin by solving the problem using a heuristic algorithm to find an upper bound named BS.
2: Then, we create the root node of the search tree which represents the released problem.
3: We solve the sub-problem using the CPLEX solver.
4: Then, we seek all neighborhood inequalities that are not satisfied by the solution of the current node, and then we add them to the released problem.
5: We solve the problem again using the CPLEX solver.
6: If the solution is integer and inferior to BS then we update BS.
7: If the solution is fractional and inferior to BS then we do:
  7.a: Perform connections,
  7.b: Choose an unexploited node,
  7.c: Go back to 3.

## VII. NUMERICAL RESULTS

In this section, we present the numerical results of our branch-and-cut algorithm. For the implementation, we use SCIP, which is a framework allowing a total control of the solution process. The experiments were performed using a computer DELL PRECISION T3500 with an Intel Xeon 5 GHz processor.

To test the effectiveness of our algorithm, we naturally compare it to CPLEX version 12.5. Performed tests on several instances prove that BC-CSP is very fast and it is able to solve large instances which can not be solved by CPLEX because requiring a lot of memory.

In Table I, we note the execution times of BC-CSP and of CPLEX for various instances.

— means that the execution is interrupted because it lasted more than 3 hours.

--- means that the computer memory is insufficient to resolve this instance.

TABLE I. NUMERICAL RESULTS

| $N$ | $N_p$ | BC-CSP | CPLEX |
|---|---|---|---|
| 100 | 500 | 0 sec | 2 min 58 sec |
| 150 | 500 | 1 sec | 15 min 45 sec |
| 100 | 700 | 0 sec | 4 min 11 sec |
| 100 | 1500 | 0 sec | 11 min 16 sec |
| 50 | 200 | 0 sec | 2 sec |
| 200 | 200 | 3 sec | 14 min |
| 80 | 100 | 0 sec | 1 min 5 sec |
| 90 | 100 | 0 sec | 1 min 29 sec |
| 100 | 100 | 0 sec | 2 min 11 sec |
| 150 | 200 | 1 sec | 53 min 50 sec |
| 100 | 3500 | 0 sec | --- |
| 100 | 3500 | 0 sec | --- |
| 200 | 3500 | 3 sec | --- |
| 300 | 3500 | 14 sec | --- |
| 400 | 3500 | 41 sec | --- |
| 500 | 3500 | 1 min 36 sec | --- |
| 600 | 3500 | 6 min 13 sec | --- |
| 700 | 3500 | 5 min 57 sec | --- |
| 800 | 3500 | 9 min 49 sec | --- |
| 900 | 3500 | 15 min 35 sec | --- |
| 1000 | 3500 | 1 h 41 min 26 sec | --- |
| 1100 | 3500 | 1 h 43 min 47 sec | --- |
| 1200 | 3500 | 2 h 5 min 4 sec | --- |
| 1300 | 3500 | 2 h 21 min 20 sec | --- |
| 1400 | 3500 | — | --- |

 ISBN: 978-1-61208-319-3 

In Table I, we remark that, in the most cases, the resolution of a great instance requires more time than the resolution of a small instance. However, in some cases, we observe the reverse. This phenomenon can be justified by the influence of the values of parameters like the departure times, the unloading order, etc. In fact, in some cases the search tree can have too lot of nodes; therefore its exploration may require more time. But, even with these instances, our branch-and-cut algorithm is faster than CPLEX.

The mathematical model of our container storage problem has too many variables, especially when there are a lot of empty stacks in the terminal. Therefore, the elimination of the farthest stacks reduces the size of the problem and improves the resolution. Fig. 2 shows that preprocessing reduces the execution times.

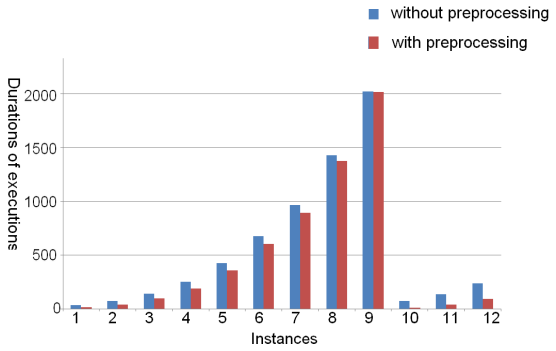


Fig. 2 Comparisons of execution times

As can be seen in Fig. 2, the preprocessing is more efficient when the number of stacks is superior to the number of containers.

## VIII. CONCLUSION AND FUTURE WORK

In this paper, we studied the container storage problem. We widely improve the work that we did in [1] by considering additional constrains in order to avoid reshuffles at quays. We take into account the order in which containers are unloaded from vessels, and we minimize the total distance traveled by SC between quays and container yards in order to shorten the berthing times of ships. The major contribution of this paper is the effective branch-and-cut algorithm, which is very fast and is able to solve great instances. This is an exact resolution method, unlike the hybrid algorithm proposed in [1], which has an average percentage deviation equal to 10.22%. It may be possible to improve our branch-and-cut algorithm; therefore we prospect to design more effective branching rules and separation methods. We also plan to adapt our approach to container terminals which use modern equipments, such as automatic guided vehicles.

## REFERENCES

[1] R. Moussi, N. F. Ndiaye, and A. Yassine, “Hybrid Genetic Simulated Annealing Algorithm (HGSAA) to Solve Storage Container Problem in Port,” Intelligent Information and Database Systems, Lecture Notes in Computer Science, Pan. Chen. Nguyen. Berlin, Intelligent Information and Database Systems, Lecture Notes in Computer Science , vol. 7197, March. 2012, pp. 301-310.

[2] C. Zhang, J. Liu, Y. W. Wan, K. G. Murty, and R. J. Linn,“ Storage space allocation in container terminals,” Transportation Research Part B: Methodological, vol. 37, December. 2003, pp. 883-903.

[3] M. Bazzazi, N. Safaei , and N. Javadian, “A genetic algorithm to solve the storage space allocation problem in a container terminal,” Computers & Industrial Engineering, vol. 56, February. 2009, pp. 44-52.

[4] C. Park and J. Seo, “Mathematical modeling and solving procedure of the planar storage location assignment problem,” Computers & Industrial Engineering, vol. 57, October. 2009, pp. 1062-1071.

[5] D. Lee, J. X. Cao, Q. Shi, and J. H. Chen, “A heuristic algorithm for yard truck scheduling and storage allocation problems,” Transportation Research Part E: Logistics and Transportation Review, vol. 45, September. 2009, pp. 810-820.

[6] E. Kozan and P. Preston, “Mathematical modeling of container transfers and storage locations at seaport terminals,” OR Spectrum, vol. 28, October. 2006, pp. 519-537.

[7] S. Sauri and E. Martin, “Space allocating strategies for improving import yard performance at marine terminals,” Transportation Research Part E: Logistics and Transportation Review, vol. 47, November. 2001, pp. 1038-1057.

[8] K. H. Kim and H. B. Kim, “Segregating space allocation models for container inventories in port container terminals,” International Journal of Production Economics, vol. 59, March. 1999, pp. 415-423.

[9] C. Jinxin, S. Qixin, and D. Lee, “A Decision Support Method for Truck Scheduling and Storage Allocation Problem at Container,” Tsinghua Science & T echnology, vol. 13, October. 2008, pp. 211-216.

[10] M. Yu and X. Qi, “Storage space allocation models for inbound containers in an automatic container terminal,” European Journal of Operational Research, vol. 226, April. 2013, pp. 32-45.

[11] K. H. Kim and K. Y. Kim, “Optimal price schedules for storage of inbound containers,” Transportation Research Part B: Methodological, vol. 41, October. 2007, pp. 892-905.

[12] P. Preston and E. Kozan, “An approach to determine storage locations of containers at seaport terminals,” Computers & Operations Research, vol. 28, September. 2001, pp. 983-995.

[13] K. H. Kim, Y. M. Park, and K. Ryu, “Deriving decision rules to locate export containers in container yards,” European Journal of Operational Research, vol. 124, July. 2000, pp. 89-101.

[14] L. Chen and Z. Lu, “The storage location assignment problem for outbound containers in a maritime terminal,” International Journal of Production Economics, vol. 135, January 2012, pp. 73-80.

[15] Y. J. Woo and K. H. Kim, “Estimating the space requirement for outbound container inventories in port container terminals,” International Journal of Production Economics, vol. 133, September 2011, pp. 293-301.

[16] K. H. Kim and K. T. Park, “A note on a dynamic space-allocation method for outbound containers,” European Journal of Operational Research, vol. 148, July. 2003, pp. 92-101.

[17] E. Nishimura, A. Imai, G. K. Janssens, and S. Papadimitriou, “Container storage and transshipment marine terminals,” Transportation Research Part E: Logistics and ransportation Review, vol. 45, September. 2009, pp. 771-786.

[18] B. Dushnik and E. W. Miller, “Partially ordered sets, American Journal of Mathematics”, 1941, vol. 63, pp. 600-610.

[19] K. Jansen, “The mutual exclusion scheduling problem for permutation and comparability graphs,” STACS 98, Lecture Notes in Computer Science, vol. 1373, February. 1988, pp. 287-297.

 ISBN: 978-1-61208-319-3 

[20] F. Bonomo, S. Mattia, and G. Oriolo, “Bounded coloring of co-comparability graphs and the pickup and delivery tour combination problem,” Theoretical Computer Science, vol. 412, October. 2011, pp. 6261-6268.

[21] I. Mindez-Diaz and P. Zabal, A Branch-and-cut algorithm for graph coloring, Discret Applied Mathematics, vol. 154, April. 2006, pp. 826-847.

 ISBN: 978-1-61208-319-3 