A GENETIC ALGORITHM AND NEW MODELING TO SOLVE CONTAINER LOCATION PROBLEM IN PORT

Riadh MOUSSI [1] Ndèye Fatma NDIAYE [1] Adnan YASSINE [1, 2]

[1] Laboratory of Applied Mathematics of Le Havre (LMAH), Le Havre University.
25 rue Philippe Lebon – B.P. 540 - 76058 Le Havre cedex, France
E - mail: moussi.riadh@gmail.com, farlou@live.fr

[2] Superior Institute of Logistics Studies (ISEL).
Quai Frissard – B.P. 1137 - 76063 Le Havre cedex, France.
E - mail: adnan.yassine@univ-lehavre.fr

Abstract

Container terminals are essential intermodal interfaces in the global transportation network. The efficient handling to container at terminals is very important for reducing transportation costs and keeping shipping schedules. In this paper, we study the storage of containers in the storage yards of terminals. We model the seaport system with the objective of determining the optimal storage strategy for various container-handling schedules. A new container location model (CLM) is developed, with an objective function designed to minimize the total distance of transport containers between the vessel berthing locations and their storage positions. Due to the inherent complexity of the problem, a genetic algorithm is designed to attain solutions very close to optimal solutions. This problem is solved also by an exact method which is Branch and Bound using the commercial software ILOG CPLEX. The optimal solutions for small-scale problems given by Branch and Bound are used to prove the efficiency of the proposed genetic algorithm. Computational results on real dimensions taken from the terminal of Normandy, in Le Havre port, in France, show the good quality of the solutions obtained by the Genetic Algorithm (GA).

Keywords: Port container terminal, Storage containers, Genetic Algorithm (GA).

1. Introduction

Container terminals play an important role in marine transportation; they constitute transfer stations to multimodal transport. The container storage is one of the most important services in a container terminal. To increase the efficiency of a container terminal, containers are optimally stacked in the storage areas in the form of stacks. The maximum height of stacks is fixed by the port authorities, based on the equipments used. The container stacks are arranged in rows aligned, called sides. A set of sides form a block. Each storage area consists of many blocks (see Figure 01).

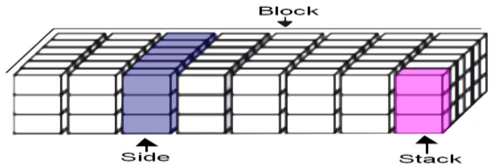

Figure 01. Blocks in port.

Generally, a container terminal is operated with three types of material handling equipment: Quay Cranes (QCs); Yard Cranes (YCs) and Trucks. After unloading of containers from ships by the QCs, containers will be allocated to blocks for temporary storage. In a storage zone, the YCs are engaged to select and to stack containers in blocks. The trucks provide the transfer of containers to accommodate the scheduling of QCs and YCs. We have also another type of container terminal which is operated with only two types of material handling equipments: QCs and Straddle Carrier (SC). Each SC carries a container and stores it in a well defined stack in a side of a block of storage area.

To achieve optimal transfer containers in a port container terminal, the global problem is divided into two dependent problems:

1. Vehicles Routing Optimization (routing of trucks): the goal is to optimize the routing of trucks to organize the transfer of containers in order to minimize the total travel time of all trucks, respecting the time window of every container and every trucks.
2. Optimization of storage of containers in the storage area.

We treated and solved the first problem in previous articles (see 1 and 2 in previous works). In this work, we address the second problem.

This paper examines the method employed in the storage of containers at a port container terminal. The focus is to utilize the storage area in a more optimal manner; thus reducing the time required for the transfer of containers. The objective of the model is to determine the optimal storage strategy for containers. A new container location model is designed to address the objective of the research. This model is based on three major constraints: (1) consider the state of the storage area before the arrival of containers, (2) for each stack, containers are stored in the decreasing order of their departure time from the yard, (3) containers are stored by respecting the constraint of compatibility in each stack (containers dimension). The goal of this work is to minimize the unloading time of a number of containers and to determine an optimal storage strategy.

The problem treated is known to be NP-hard category. Thus, GAs are proposed to solve this problem. In order to prove the efficiency of the algorithms proposed, we compare them to an exact method (Branch and Bound method) using the software ILOG CPLEX with small-scale problems. The rest of this paper is organized as follows: a brief overview and related literature is presented in Section 2. We provide a description of the problem treated in Section 3. The mathematical model representing the concrete problem is formulated in Section 4. We propose two genetic algorithms (one-cut-point and two-cut-point) which are developed in Section 5. The results of numerical simulations of all the implemented algorithms and a comparative study are presented in Section 6, and finally Section 7 is devoted to conclusion.

2. Literature Review

Various aspects of the operational problems in the container terminal are developed in the literature. Stahlbock et al. (2008) have presented the current state of the art in container terminal operations and operation research. Steenken et al. (2004) have described and classified the main logistic processes and operations in container terminals. Operational problems of the container terminal have been divided into several problems such as:

1. Berth assignement: Imai et al. (2001) have discussed the problem of determining a dynamic berth assignment to ships.

2. Scheduling of QCs: Lee et al. (2008) have focused their research on the QCs scheduling problem in order to determine a handling sequence of holds for QCs assigned to vessels.
3. Scheduling of YC: Lee et al. (2007) have discussed the scheduling of two YCs systems which serve the loading operations of one QC at two different container blocks in order to minimize the total loading time at stack area.
4. Scheduling of trucks: Kap et al. (2004) have discussed how to dispatch trucks using information about locations and times of future delivery tasks.
5. Storage container problem which is the problem treated in this paper.

Storage containers are a critical resource in container terminals. In the terminals, the loading sequence of export containers affects significantly the productivity of port operations. However, the optimal allocation of containers allows a sequence of optimal loading. It affects the efficiency of delivery and loading operations**.** Peter. P. et al. (2001) developed a model to determine optimal storage strategies and container handling schedules. They proposed a heuristic method with a GA. In (2006), they treated the same problem with an improved algorithm. Indeed, they developed a GA, a Tabu Search algorithm (TS) and a hybrid algorithm between TS and GA. Mohammad. B. et al. (2009) solved an extended Storage Space Allocation Problem (SSAP) in a container terminal by an efficient GA. The SSAP is defined as the temporary storage of the inbound and outbound containers of the storage blocks. The objective of the SSAP developed is to minimize the time of storage and retrieval time of containers. Changkyu. P. et al. (2009) focused on the planar storage location assignment problem (PSLAP). The PSLAP can be defined as the assignment of the inbound and outbound containers of the storage area in order to minimize the number of obstructive moves. The PSLAP is resolved by a GA. Chuqian. Z. et al. (2003) treated the SSAP by using a rolling-horizon approach. For each planning horizon, the problem is decomposed into two levels: At the first level, they defined for each period the number of containers to be placed in each storage block. At the second level, they found out the number of containers stored in each block at each period associated with each vessel. The objective of the work of Chuqian. Z. et al. is to minimize the total distance to transport containers between their storage blocks and the vessel berthing locations.

There is another problem developed by some academic researchers to treat the storage container problem which is the storage of inbound containers and outbound containers. The storage location assignment problem for outbound containers is treated by, Lu. C. et al. (2009), the objective of the problem is to minimize the rehandling operations by cranes in order to maintain the stability of the ship. The problem is decomposed into two stages. In the first stage, the numbers of locations in each yard bay are determined by a mixed integer programming model. In the second stage, the exact storage location for each container is determined by a hybrid sequence stacking algorithm. The same problem is discussed by Kap. H. K. et al. (2003). They formulated a basic model as a mixed-integer linear program and they suggested two heuristics algorithms to solve the problem. The storage location assignment problem for inbound containers is one of the problems developed to solve the storage container problem. Kap. H. K. et al (2007) discussed a method of determining the optimal price schedule for storing inbound containers. Kap. H. K. et al. (1999) proposed a mathematical model to allocate storage space for import containers using the segregation strategy in order to minimize the number of rehandles.

3. Context

When ships arrive at ports, they remain inactive during the operations of loading and unloading containers. Each ship follows a calendar provided by the port authorities. Based on this calendar and the various constraints of the terminal, the port authorities prepare the dates of loading and unloading containers and precise the storage positions of containers. One of the major problems of a terminal is to store containers in an optimal way. We treat the storage of containers in the port container terminal. The goal of this work is to minimize the unloading time of containers and to determine an optimal storage strategy. In our case, the rehandling operations are not accepted, i.e., when a container is stored, it is not moved from its position until the departure time.
We model the problem with a new mathematical model that reflects reality and takes into account most of the constraints imposed by port authorities. This model treats the following hypotheses:

1. We don’t mix on the same block and in the same period the loading and the unloading containers. Blocks which are used to receive unloading containers are fixed before the beginning of each period.
2. Before the beginning of each period, we know the state of the storage area. For each stack, we know: the number of container stored, the departure time of every container and the type of the stack (dimension of containers in the stack).
3. For each stack, containers are stored in the decreasing order of their departure time from the yard.
4. Containers are stored by respecting the constraint of compatibility. We know the type of stack which is simply the dimension of its containers. All containers stored in the same stack have the same dimension. You cannot store two containers of different dimensions in the same stack.
5. The maximum number of container stored in each stack is fixed to 3 containers.

4. Mathematical Model

In this section, the container location problem in a port is formulated as a new and original mathematical programming model. This model is applied on each period in order to minimize the unloading time of containers and to determine an optimal storage strategy based on the following assumptions. Note that p represents stacks, i is the index of the empty position in a stack and k represents containers.
N: Represents the number of containers.
N_P: Represents the number of stacks.

c_p*: Represents the number of empty position for stack p.*
r_p*: Represents the type of stack p.*
t_p*: Represents the date of stack p. The date of a stack is equal to the departure time of the container stored on the top else if the stack is empty* $t_p = M$*, M is a big number.*
R_k*: Represents the type of container k.*
T_k*: Represents the departure time of container k.*
d_{pk} *: Represents the shortest way between the position of the ship of container k and the stack p.*

$$\lambda_{pik} : \begin{cases} 1 & \textit{if container } k \textit{ is stored in a position } i \textit{ of stack } p. \\ 0 & \textit{otherwise.} \end{cases}$$

The problem treated in this paper can be formulated as the following linear mathematical model:

$$Min = \sum_{p=1}^{N_p} \sum_{i=1}^{c_p} \sum_{k=1}^{N} \lambda_{pik} d_{pk} \qquad (1)$$

$$\sum_{p=1}^{N_p} \sum_{i=1}^{c_p} \lambda_{pik} = 1, \quad k = 1, \ldots \mathrm{N}. \qquad (2)$$

$$\lambda_{pik} + \lambda_{pik'} \leq 1, \quad k = 1, \ldots, N, \quad k' = 1, \ldots, N, \; k \neq k', \quad p = 1, \ldots, N_p, \quad i = 1, \ldots, c_p. \qquad (3)$$

$$\sum_{k=1}^{N} \lambda_{pik} \geq \sum_{k'=1}^{N} \lambda_{pi+1k}, \quad p = 1, \ldots, N_p, \quad i = 1, \ldots, c_p. \qquad (4)$$

$$(r_p - R_k)\lambda_{pik} = 0, \quad k = 1, \ldots, N, \quad p = 1, \ldots, N_p, \quad i = 1, \ldots, c_p. \qquad (5)$$

$$T_k \leq M(1 - \lambda_{pik}) + \lambda_{pik} t_p, \quad k = 1, \ldots, N, \quad p = 1, \ldots, N_p, \quad i = 1, \ldots, c_p. \qquad (6)$$

$$M(1-\lambda_{pik})+\lambda_{pik}T_k \geq \lambda_{pi+1k'}T_{k'},\quad k=1,\dots,N,\quad k'=1,\dots,N,\quad p=1,\dots,N_p,\quad i=1,\dots,c_p. \qquad (7)$$

The objective of this model is to minimize the distance between the ship of each container and their storage position in the storage zone, see constraint (1). The second objective of the model is to determine an optimal storage strategy which is developed by the following assumptions. Each constraint is more explained by a figure (Figure 02 to Figure 07). Each figure represents a case that cannot be accepted by one of the constraint developed. Constraints (2) ensure that, each container is stored in one storage position. For example in Figure 02, container *k=1* cannot be stored on the same time in position *i=3* of stack *p=1* and position *i'=1* of stack *p'=2*. Each position in each stack receives only one container. This idea is developed by constraint (3). Containers *K=1* and *K'=2* cannot be stored in position *i=1* in stack *p=2*, see Figure 03. Constraint (4) ensures that, an empty intermediate positions between containers stored in the same stack are not accepted. For example, in Figure 04, Containers *k=1* and *k'=2* are stored respectively in position *i=1* and *i=3* in stack *p=2* and position *i=2* is empty; this case is not accepted. Each container is stored in a stack with the same type; this idea is developed in Constraint (5). In Figure 05, container *k=1* has a type *R=1*. Then, it cannot be stored in stack *p=2* with type *r=2*. Constraint (6) and (7) ensure that containers must be stored in a stack in the decreasing order of their departure time from the yard. Constraint (6) takes into account the container stored before the beginning of the new period. Container *k=1* has a departure time *T=7* and cannot be stored in stack *p=2* because the departure time of the container stored on the top of this stack (*t=2*) is inferior than the departure time of *k=1*, see Figure 06. In constraint (7), we compare the date of containers which will be stored in the same stack p and they must be stored in the decreasing order of their departure time. In Figure 07, containers *k=1* and *k=2* should be stored on the decreasing order of departure time; that means *k=1* should be stored before *k=2*.

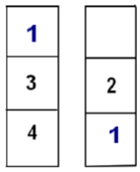

Figure 02. Constraint (2).

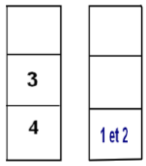

Figure 03. Constraint (3).

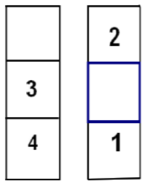

Figure 04. Constraint (4).

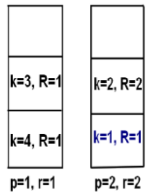

Figure 05. Constraint (5).

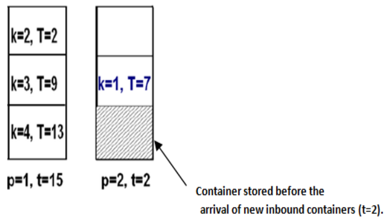

Figure 06. Constraint (6).

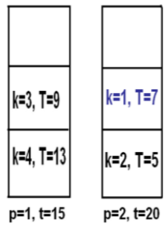

Figure07. Constraint (7).

5. Genetic Algorithm Implementation

5.1. Introduction

The container storage problem is formulated as a linear integer programming problem. The problem is known to be NP-hard and the computation complexity increases exponentially. This makes it difficult his resolution in reasonable time with exact method techniques (Branch and Bound Method, for example). This implies that, the solution of large instances requires the use of approximate methods (heuristics and meta-heuristics). The choice to apply a GA was justified by the good results provided by genetic algorithms applied on similar problems treated in the literature. M. Bazzazi et al. (2009) applied a GA to solve an extended storage space allocation problem (SSAP), E. Kozan. et al. (2001and 2006) developed a GA to solve the Container Location Problem (CLP). C. Park. et al. (2009) focused on the planar storage location assignment problem and solved the problem with a GA.

A GA has been developed by J. Holland in the 1970's to understand the adaptive processes of natural systems. The GA was applied to several problems such as: the travelling salesman problem (TSP), vehicle routing problem (VRP) and many other problems. Their operation is as follows: we begin with an arbitrary initial population of potential solutions (chromosomes). We estimate their relative performance. We create a new population of potential solutions using simple evolutionary operators: selection, crossover and mutation. This cycle is repeated until the obtaining of a satisfactory solution. We develop two GAs to solve the container location problem. The first is based on the one-cut-point as a crossover method and the second is based on the two-cut-point as a crossover method. To see more variants of choice of crossover operators, mutation and selection, you can review the thesis of S. Bourazza (2006) and the book of T. El-Ghasali (2009).

5.2. Chromosomes

Each chromosome is represented as a table with two lines (see Figure 08). The first line contains the empty positions of each stack and the second line contains containers assigned to each stack. The number of columns is equal to the number of the empty positions of all stacks.

1	1	2	3	3	3
3		1	4	2	

Figure 08. Chromosome.

For example, in Figure 08, there are 3 stacks with 6 empty positions (for example, stack 2 has only one empty position) and 4 containers stored. For example, containers 4 and 2 are stored, respectively, in stack 3. Container 4 is stored before container 2 and the third position remains empty. Containers are stored respecting all the constraints developed in Section 4**.**

5.3. Fitness evaluation

The problem treated is a minimization problem. Thus, the smaller objective function value must be the higher fitness value. This paper defines the fitness function as the reciprocal of the objective function.

5.4. Selection method

The selection strategy means how to choose the chromosomes in the current population that will create offspring for the next generation. The most common method for the selection mechanism is the ‘‘roulette wheel”, in which each chromosome is assigned a slice of a circular roulette wheel and the size of the slice is proportional to the chromosome’s fitness. The roulette wheel is used in this work in order to select parents to create a new generation.

5.5. Crossover operators

Crossover operates on two chromosomes and generates offspring by recombining current genes. Crossover is usually accomplished by either the one-cut-point method or the two-cut-point method. In this paper, we develop the two methods of crossover.

5.5.1. One-cut-point

In one-cut-point method, we choose two chromosomes and fix a cutting position (see Figure 09). The cutting position divides the two parents into two segments. In order to obtain the new children, we permute the two segment situated on the right of the cutting position between the two parents (see Figure 10). Crossover may generate infeasible children that do not satisfy constraints. In order to keep the feasibility, the crossover-repair operation is performed in the following manner. The first problem is that we found a duplicate container and then we keep the first affectation of each duplicate container and remove the second one. After, removing a container, we should verify the 4th constraint of the model. If the child obtained has an empty intermediate position, then, we apply the procedure developed in Section 5.6. (see Figure 15). The second problem is missing containers, for example, in child 1, container 2 is absent, and then we add missing containers on the first stack that can receive it. Finally, we obtain valid children (see Figure 11).

5.5.2. Two-cut-point

In two-cut-point method, we choose two chromosomes and fix two cutting

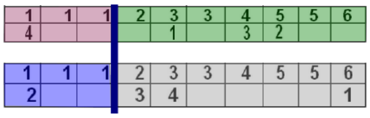

Figure 09. One-cut-point.

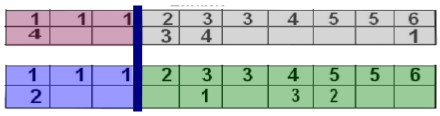

Figure 10. Invalid children.

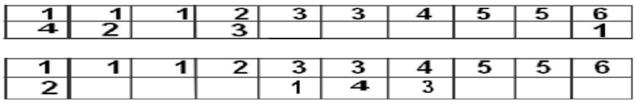

Figure 11. Valid children.

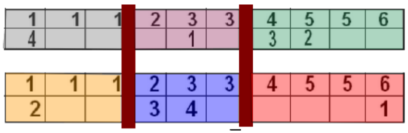

Figure 12. Two-cut-point.

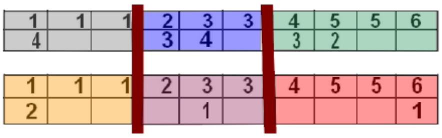

Figure 13. Invalid children.

positions for each one (see Figure 12). In order to obtain children, we permute the two parts situated between the two cutting position (see Figure 13). Crossover may generate infeasible children that do not satisfy constraints. We apply the same correction used with one-cut-point.

For one-cut-point and two-cut-point after correction, we obtain two valid children and we introduce to the next generation the best one.

5.6.Mutation

Mutation introduces random changes to the chromosomes by altering the value to a gene with a user-specified probability called mutation rate. In our case, we choose randomly a container and we affect it to another stack respecting all constraints (see Figure 14). Mutation may generate infeasible offspring that do not satisfy the 4^{th} constraint. Then, we must correct it (see Figure 15). For example, we choose container 4, which is stored in stack 3 in position 1, then, we can store it in stack 1 in position 2. We remove container 4 to the new position but we note that, the first position in stack 3 becomes empty and the second one is occupied by container 2 then constraint 4 is not respected. We move container 2 from the 2^{nd} position to the 1^{st} one and update the capacity and the date of stack 3.

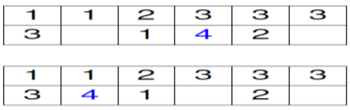

Figure14. Mutation.

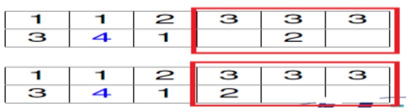

Figure15. Correction

5.7. Summary of parameter values

All GA runs used in this paper have the following standard characteristics:

One-cut-point parameters:

- Crossover rate: 0.75
- Mutation rate: 0.025
- Population size: 30
- Number of generation: 1000.

Two-cut-point parameters:

- Crossover rate: 0.47
- Mutation rate: 0.05
- Population size: 40
- Number of generation: 1000.

6. Numerical Results

In this section, firstly, we compare the two GAs to Branch and Bound with small-scale problems. Second, we generate some large-scale problems which are solved by the two genetic algorithms proposed.

6.1. Small-scale problems

In this section, the two GAs proposed are compared to an exact method which is Branch and Bound developed by ILOG CPLEX with small scale problems (cannot exceed 100 containers). This comparison is carried out in order to verify the quality of the GAs proposed. A comprehensive set of test problems randomly generated based on real-life terminal operations. For each container and each stack, we have three types [20 feet, 40 feet, 45 feet], a departure time is uniformly distributed in [10, 80] for each container and we calculate the distance between its ship and the different stacks. For each stack, the capacity is fixed from [1, 2, 3] and a departure time of the container situated on the top of each stack is uniformly distributed in [40, 100] else if the capacity of the stack is equal to 3, then, its departure time is equal to M.

In Table01, we present 31 instances generated that can be solved by Branch and Bound and GAs. The first column of each table contains: N_p = number of stacks, N = number of containers, and P_d = percentage of free position in stacks which is calculated by: $\frac{\sum offree\ position}{3 \times N_p} \times 100$. The second, third and fourth columns represents, respectively, the value of the objective function for each instances founded by the one-cut-point, the two-cut-point and the optimum results given by Branch and Bound. The last column represents the percentage deviation of the two GAs from the optimum results. The percentage deviation is calculated by this formula: $\frac{GA\ results - optimum}{optimum} \times 100$.

Table 01: Comparison between GAs and Branch and Bound.

Instances			GA with one-cut-point	GA with two-cut-point	Branch and Bound	Percentage deviation	
N_p	N	P_d				one-cut-point	two-cut-point
20	10	80%	4350	4500	4150	**4.81%**	8.43%
35	25	67.61%	17800	17150	15350	15.96%	**11.72%**
40	20	68,33%	12650	12950	10500	**20.47%**	23.33%
	25	73.33%	19700	20500	17650	**11.61%**	16.14%
45	30	57.03%	24800	25100	22750	**9.01%**	10.32%
	35	66.66%	27950	27400	21250	31.52%	**28.94%**
50	30	60%	26000	25250	21700	19.81%	**16.35%**
	35	66.66%	33500	36050	31500	**6.34%**	14.44%
	40	71.33%	34050	34500	27750	**22.70%**	24.32%
55	40	66.66%	37900	42100	31100	**21.86%**	35.36%
60	40	65%	41300	42100	32200	**28.26%**	30.74%
	45	64.44%	47450	50550	43250	**9.71%**	16.87%
65	40	64.61%	41450	41500	33850	**22.45%**	22.59%
70	45	67.14%	54150	55250	44200	**22.51%**	25%
	50	67.14%	61450	61250	53150	15.61%	**15.25%**
	55	73.33%	70200	70550	56050	**25.24%**	25.86%
	60	61.42%	81450	85700	72850	**11.8%**	17.63%
75	60	67.11%	80400	84800	66200	**21.45%**	28.09%
80	60	65%	75750	81300	60850	**24.48%**	33.6%
	65	68,33%	100400	101050	84400	**18.95%**	19.72%
85	60	68,62%	80800	83100	60950	**32.56%**	36.34%
90	65	69.62%	90400	89200	64400	40.37%	**38.5%**
	70	64.44%	102550	107000	87150	**17.67%**	22.77%
95	70	70.17%	106000	117850	78300	**35.37%**	50.51%
	75	66.31%	113300	122450	91750	**23.48%**	33.46%
	80	61.4%	131800	135650	112000	**17.67%**	21.11%
100	80	64%	145550	152100	122600	**18.71%**	24.06%
	85	63%	144200	143550	111300	29.55%	**28.97%**
105	95	69.2%	168550	167000	132750	26.96%	**25.8%**
110	95	66.96%	180200	185450	145600	**23.76%**	27.36 %
	90	65.75%	159450	162950	124100	**28.48%**	31.3%
Average percentage deviation						**21.26%**	24.67%

In Table01, 31 instances were generated and solved by Branch and Bound and the two GAs developed before. The two GAs are compared and we note that, in 24 cases one-cut-point is more efficient than two-cut-point.
To have an idea of the efficiency of the one-cut-point, we calculate the average of percentage deviation which is defined as: $\frac{\sum percdev}{number\ of\ instances}$. The average of percentage deviation is equal to 21.26%. The one-cut-point is more efficient than the two-cut-point.

6.2. Large-scale problems

In order to generate some large-scale problems, we keep the same notation presented before with small instances and we change only the interval of departure time of containers which is uniformly distributed in [200, 1000] and for stacks is uniformly distributed in [500, 2000].

Table 02: GAs with large-scale.

Instances			GA with one-cut-point	GA with two-cut-point
N_p	N	P_d		
300	100	64.22%	**395450**	399250
	200	66.66%	**905550**	914250
	300	66.66%	**1489350**	1534850
400	200	66.5%	**1123200**	1453000
	300	50.16%	**1816600**	1880850
	400	66.33%	**2684350**	2735400
	500	66.33%	**3674450**	3753800
500	100	66.53%	597650	**585150**
	200	67.13%	**1351150**	1397600
	300	65.93%	**2171400**	2224450
	400	67.2%	**3099950**	3157300
600	100	66.66%	**669800**	710500
	200	66.72%	**1590000**	1610400
	300	67.94%	**2586500**	2697350
	400	67.77%	**3559350**	3634900
700	100	67.42%	**755700**	798950
	200	67.52%	**1895900**	1930900
	300	67.14%	2956350	**2939200**
	400	66.61%	**4108250**	4175000
	500	66.09%	**5355300**	5354950

We note that, the GA with one-cut-point proposed is able to solve the storage of container in port with a set of test problems randomly generated based on real-life terminal operations.

7. Conclusion

In this paper, a new container location model is designed to minimize the unloading time of containers and to determine an optimal storage strategy. This model is based on three major constraints: (1) consider the state of the storage area before the arrival of containers (2) for each stack, containers are stored in the decreasing order of their departure time from the yard (3) containers are stored by respecting the constraint of size compatibility in each stack. It also respects the other standard constraints required by the port authorities. The problem is NP-Hard. This requires the use of meta-heuristics methods to find an approximated optimal solution for the large instances where it is impossible to determine the optimal solution by exact methods. The good results obtained by GAs applied on similar problems have motivated us to apply a GA to solve this problem. We applied two variants of a GA, the first uses the crossover operator "one-cut-point" and the second uses the crossover operator "two-cut-point".

By comparing the results obtained by these two variants of GA with the exact results provided by ILOG CPLEX on problems of small dimensions, we found that, the results of the variant "one-cut-point" are more efficient than the variant "two-cut-point". In addition, GA with one-cut-point is more satisfactory than using two-cut-point on the problems of large dimensions. The novelty of this work is that, we propose a new model to solve the Container

Location Problem in ports. The model developed is based on real-life terminal operators taken from the terminal of Normandy Le Havre port, France. In order to solve the model, we propose a GA based on two types of crossover operators. The one-cut-point is the best one and its efficiency is proved by the quality of solution obtained with large-scale problems (real problem). But, the major inconvenient of this algorithm that is far to the optimum results (The average of percentage deviation is equal to 21.26%). For these reasons, our perspective is to find another strategy to improve outcomes obtained in this work and find solutions very close to the optimum.

8. References

1. Moussi. R, Yassine. A, Kansou. A, Galinho. T, "Scheduling of lifting vehicles with time windows in an automated port container terminal", LOGISTIQUA 2011, 4th International Conference on logistics, IEEE Xplore, p.p. 55-61, 2011.
   http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5939403
2. Moussi. R, Yassine. A, Kansou. A, Galinho. T, "Scheduling of vehicles under containers time windows in port", Journal of Informatics and Mathematical Sciences (on reviewing).
3. Bourazza. S. Variants of genetic algorithms applied to scheduling problems. PhD Thesis, University of Le Havre, France, (2006).
4. Changkyu. P, Junyong. S, "Mathematical modeling and solving procedure of the planar storage location assignment problem", Computers & Industrial Engineering 57: 1062–1071, (2009).
5. Chuqian. Z, Jiyin. L, Yat-wah. W, Katta. M, Richard. L, "Storage space allocation in container terminals", Transportation Research 37: 883–903, (2003).
6. El-Ghazali. T, Metaheuristics from design to implementation, John Wiley & Sons, (2009).
7. Erhan. K, Peter. P, "Mathematical modeling of container transfers and storage locations at seaport terminals", OR Spectrum 28: 519–537, (2006).
8. Imai. A, Nishimura. E, Papadimitriou. S, "The dynamic berth allocation problem for a container port", Transportation Research, 35 (4): 401-417, (2001).
9. Kap. H. K., Hong. B. K, "Segregating space allocation models for container inventories inport container terminals", Int. J. Production Economics 59: 415-423, (1999).
10. Kap. H. K, Jong. W. B, "A Look-Ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals", Transportation Science, 38 (2): 224-234, (2004).

11. Kap. H. K, Kang. T. P, "A note on a dynamic space-allocation method for outbound containers", European Journal of Operational Research 148: 92–101, (2003).
12. Kap. H. K, Ki. Y. K, "Optimal price schedules for storage of inbound containers", Transportation Research 41: 892–905, (2007).
13. Lee. D. H, Cao. Z, Meng. Q, "Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm", International Journal of Production Economics, 107(1):115-124, (2007).
14. Lee. D. H, Wang. H. Q, Miao. L. X, "Quay crane scheduling with non-interference constraints in port container terminals", Transportation Research 44(1): 124-135, (2008).
15. Lu. C, Zhiqiang. L, "The storage location assignment problem for outbound containers in a maritime terminal", Int. J. Production Economics 48:991-1011, (2010).
16. Mohammad. B, Nima. S, Nikbakhsh. J, "A genetic algorithm to solve the storage space allocation problem in a container terminal", Computers & Industrial Engineering 56: 44–52, (2009).
17. Peter. P, Erhan. K, "An approach to determine storage locations of containers at seaport terminals", Computers & Operations Research 28: 983-995, (2001).
18. Stahlbock. R, Voß. S, "Operations research at container terminals: a literature update", OR Spectrum 30:1–52, (2008).
19. Steenken. D, Voß. S, Stahlbock. R, "Container terminal operation and operations research–a classification and literature review", OR Spectrum 26: 3–49, (2004).