Optimization Model for Berth and Transshipment Scheduling

PhD student Marwa Samrout
LMAH, FR CNRS 3335, ISCN, Normandie Univ, UNIHAVRE, 76600 Le Havre, France
marwa.al-samrout@etu.univ-lehavre.fr

Professor Adnan Yassine
LMAH, FR CNRS 3335, ISCN, Normandie Univ, UNIHAVRE, 76600 Le Havre, France

Professor Abdelkader Sbihi
Brest Business School, 29200 Brest, France

DOI: http://dx.doi.org/10.25673/85956

Abstract
Several research works have focused on Berth Allocation Problem (BAP) with the consideration of transshipment of ship-to-ship. However, to the most of our knowledge, there are no studies, that decide whether direct transshipment service should be made according to vessel's berthing time in the continuous (BAP) variant where vessels can berth anywhere alongside quays. To fill this gap, this study introduces the continuous (BAP) and formulates a novel mathematical model, which deals with the question of direct transshipment. The aim is to minimize the dwell times and the penalty accrued by tardy vessels [1]. Firstly, a mixed integer linear program is implemented to schedule incoming vessels to berthing positions and decide the transshipment method needed for each couple of vessels. Secondly, a genetic algorithm is proposed to solve large-scale problem instances. Numerical experiments were conducted, and the results are analysed and compared on a set of randomly generated instances. The designed solution approach provides near optimal solutions for comparable real size problems in a reasonable amount of time.

1. Introduction:
Optimizations of berth allocation and transshipment activities have received so far, a larger attention in the scientific literature in the last few years (Zhen et al. 2011 [2], 2016 [3]; Lee and Jin 2012 [4]; Tao and Lee 2015 [5], Schepler et al. 2017 [6]). [2] integrate both the berth allocation problem and container storage space allocation problem (SSAP) including the assignment of quay cranes at the tactical level in transshipment hubs. [3] focused on the terminal assignment problem considering fuel consumption, ITT, and storage cost. [4] schedule template for feeder vessels to reduce workload congestion in transshipment terminals. A memetic heuristic was developed to solve large instances. [5] studied the berth and yard allocation problem for transshipment hubs to minimize the total distance of exchanging containers between mother vessels and feeders. [6] used restrict-and-fixed heuristics to solve terminal allocation problem (TAP) for multi-terminal systems and minimize weighted turnaround times. In the literature, although there are many papers that applied continuous berth layout approach (Ganji et al. 2010 [7]; Lee et al. 2010[8]; Park and Kim 2002[9]), few of them consider a berth allocation problem with direct transshipment consideration. In their model, Liang et al. (2012) [10] assumed that the direct transshipment is only used between two vessels berthed at the same berth. However, they raised additional requirements for the berthing positions of the involved vessels and the operations of the QCs. They proposed a Genetic Algorithm (GA) and the solution of an example instance confirmed that direct transshipment can accelerate the service process. [9] studied a BAPC with an objective that minimizes the costs of delayed departures of ships due to the undesirable service order and those of additional complexity in handling containers when ships are served at non-optimal mooring locations in port. Their work is more practical than the aforementioned BAPC research works in that the factors assessed in the objective depend on the quay locations of ships. [7] proposed a GA-based heuristic which found near optimal solutions for small-sized instances with 3 vessels and was able

to solve larger instances with 30 vessels in 6 min. In [8] two greedy randomized adaptive search procedures (GRASP) are developed for the BAPC. Finally, in [11], a bi-objective optimization integration model of tactical and operational planning in container terminal operations is presented. It consists of tactical berth allocation problem, specific quay crane assignment problem, tactical yard allocation planning and quay crane scheduling problem.
In this paper, a novel mono-objective integration model is proposed to deal with the continuous variant of BAP and the direct and indirect transshipment tasks. We extend the relative position formulation of [1]. An exact method and an adapted (GA) are proposed to solve the problem. In our model, the transshipment consideration is similar to [10], however the authors dealt with a discrete variant and used a different modelling strategy.

2. Model and resolution approach

In this section, we present a mixed linear programming formulation to solve the berth allocation problem with the consideration of transshipment of ship-to-ship. Before that, we introduce some terminology and sets, which will be used in all the formulation later.

2.1. Problem Description

Arriving at the terminal, a ship must fulfil several tasks. Generally, part of its goods must be transferred directly to another ship, another part must be stored temporarily in the terminal yard while the rest of the cargo can be unloaded and stored without transshipment. The proposed mathematical model extends the research done by [1]. We assume quay is partitioned into a determined number B of equal sized docks. Each position can be occupied by at most one ship at a time. In addition, since the number of containers carried by a mega-ship is high, the processing time required to process such a ship may exceed one day. Therefore, we choose to discretize time using relatively long periods of 3 or 4 hours. We are interested in a set of ships whose arrival dates are known in advance, with $V = M \cup F$ where M and F represent respectively all mother and feeder ships. For each ship $i \in V$, We define: l_i length of the ship i, p_i: processing time of the ship i, a_i : arrival date of vessel i and d_i: estimated departure date of vessel i. Given the information vector $\{l_i , p_i, a_i , d_i\}$ of a ship i arriving at the terminal, the optimization problem then consists of to determine the berthing position b_i of this vessel $(1 \leq b_i \leq B - 1)$ where B is the set of berths), the date of berthing t_i, develop the duty order sequence and assign the start time of each handling operation for each vessel i. The problem is suitable under the following assumptions:

- The vessel must be serviced without disruption from its arrival at the terminal until its departure.
- Container transshipment operations between ships only exist between mother ships and "feeder" ships and they can be carried out between several ships at the same time. Thus, mother ships can trade containers with up to six "feeder" ships, while "feeder" ships can be assigned to up to three mother ships.
- Incoming ships can dock at the quay without any physical restrictions.

The sets, parameters and decision variables used to build the two models are as follows:
c_i: the earliest time that vessel i can depart
d_i: due time of vessel i (where $d_i \leq t_i + p_i$).
f_i: lateness penalty of vessel i.
h_i: length of vessel i measured in number of required berth sections.

$$w_{ij} \begin{cases} 1 & \text{if there is a transshipment operation between vessel i and vessel j} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} \begin{cases} 1 & \text{if vessel j berths after vessel i departs} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij} \begin{cases} 1 & \text{if vessel j berths completely above vessel i on the time} - \text{space diagram} \\ 0 & \text{otherwise} \end{cases}$$

$$D_{ij} \begin{cases} 1 & \text{if there is a direct transshipment between vessel i and vessel j} \\ 0 & \text{otherwise} \end{cases}$$

$$I_{ij} \begin{cases} 1 & \text{if there is an indirect transshipment between vessel i and vessel j} \\ 0 & \text{otherwise} \end{cases}$$

2.2. Mixed integer linear formulation

$$\text{Min } f = \sum_{i \in F \cup M}(c_i - a_i) + \sum_{i \in F \cup M} f_i . (c_i - d_i)^+ \quad (1)$$

$x_{ij} + x_{ji} + y_{ij} + y_{ji} \geq 1 \; \forall\, i, j \in F \cup M \; i < j$ (2)
$x_{ij} + x_{ji} \leq 1 \quad \forall\, i, j \in F \cup M \;\; i < j$ (3)
$y_{ij} + y_{ji} \leq 1 \quad \forall\, i, j \in F \cup M \;\; i < j$ (4)
$tj \geq ci + (x_{ij} - 1).M \;\; \forall\, i, j \in F \cup M \; i \neq j$ (5)
$b_j \geq b_i + hi + (y_{ij} - 1).M \;\; \forall\, i, j \in F \cup M \;\; i \neq j$ (6)
$t_i \geq a_i \qquad \forall\, i \in F \cup M$ (7)
$c_i \geq t_i + p_i \qquad \forall\, i \in F \cup M$ (8)
$b_i \leq B - (hi + 1) \qquad \forall\, i \in F \cup M$ (9)
$b_i \geq 1 \qquad \forall\, i \in F \cup M$ (10)
$Dij + l_{ij} = 1 \; \forall\, i \; \forall\, j \in F \cup M \; w_{ij} = 1, \; i \neq j$ (11)
$| t_i - t_j | \geq l_{ij} \;\; \forall\, i \; \forall\, j \in F \cup M \; w_{ij} = 1 \;\; i \neq j$ (12)
$| t_i - t_j | \leq l_{ij}.M \; \forall\, i \; \forall\, j \in F \cup M \; w_{ij} = 1 \; i \neq j$ (13)
$x_{ij}, y_{ij}, D_{ij}, l_{ij} \in \{0, 1\} \; \forall\, i \; \forall\, j \in F \cup M \; i \neq j$ (14)

Constraints (2)-(4) ensure that no vessel rectangles overlap. Constraints (5) and (6) ensure that the selected berthing times and berthing positions are consistent with the definitions of x_{ij} and y_{ij}, where M is a large positive scalar. Constraint (7) and (8) force berthing time to occur no earlier than arrival time, and departure time to occur no earlier than service completion time. Constraints (9) and (10) guarantee that all vessels fit on the berth. Constraint (11) ensures that a couple of incoming vessels must work with only one handling operation including non-transshipment or transshipment containers. Constraints (12) and (13) ensure that only the couple of vessels arriving at the same time should have a direct transshipment.

2.3. Genetic algorithm

Our BAP variant is clearly NP-hard [1]; To validate the model we used CPLEX to find optimal solutions for small instances. To solve large size instances, we adapted a Genetic Algorithm (GA) by combining it with techniques based on the 2D collision detection problem to select feasible solutions. We choose the GA approach since our problem structure is suitable for this metaheuristic, therefore its implementation is much easier. The first decision to make while implementing a GA is to properly define each individual. This step associates to each point of the search space a specific data structure, called chromosome. It has been observed that improper representation of solutions, having an improper definition of the mappings between the phenotype and genotype can lead to poor performance of the GA. In this problem, each solution is represented by a artificial chromosome (row) composed of a set of genes indicating the coordinates of each ship on a time-space diagram as well as their transshipment relationships. A binary representation is used for the binary variables x_{ij}, y_{ij}, D_{ij} and I_{ij} where the genotype consists of a bit strings of $4 \times (N_1 + N_2)^2$ elements, where the k^{th} element indicates whether the decision k is chosen (=1) or not (=0). For continuous variables (namely b_i, t_i and c_i), the genes are defined using a real valued representation of $3 \times (N_1 + N_2)$ elements. In Fig 1, we illustrate an explanatory example of the proposed coding. Such a representation offers certain facilities. Indeed, the genotype becomes structured and can be broken down into different identifiable parts, thus allowing its exploitation by crossbreeding and mutation operators.

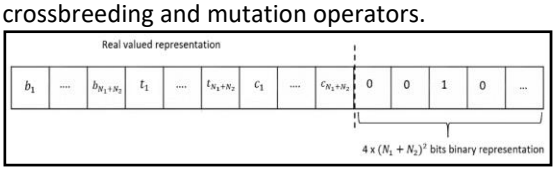

Figure 1: Representation of an individual's chromosome (genotype).

In our algorithm, the population is defined as a two-dimensional array of – size population, having two main properties: the position in the search space and the cost. The diversity of the population should be maintained otherwise it might lead to premature convergence. For this purpose, we populate the initial population with completely random solutions. We note that the population size was decided by trial. GA strongly depends on the step of selecting parents. However, when the problem has several constraints, it becomes difficult to find a randomly feasible solution. For this reason, we have designed technique based on the 2D collision detection problem [12] to select feasible parents. A feasible solution of the BAP is called a berth schedule x. Any such x can be depicted on a time-space diagram where the horizontal axis measures time and the vertical axis represents berth sections [1]; see Fig 2. A vessel i is modeled by a rectangle whose length is its processing time p_i and height is its length h_i. To determine berthing section b_i and berthing time period t_i for each vessel i, we select pairs of known solutions as parents from the current population and we ensure there is no gap between any of the 4 sides of the rectangles. Any gap means a collision does not exist

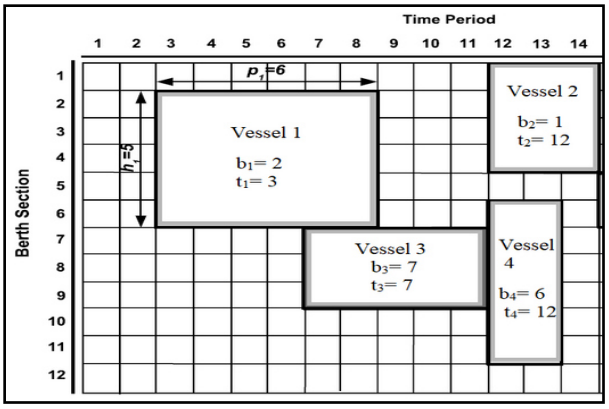

Figure 2: Representation of a berth schedule on a time-space diagram. Figure taken from [1] and readjusted.

2.3.1. Crossover operators

In the proposed problem, the resulting vessel and container transshipment scheduling plans may not be feasible if the crossover operator is not properly designed. In our model, two main types of variables are used: the binary and the continuous variables. For each type different crossover operators have been used to deal with the proposed problem. For the binary variables one of these 3 operators is randomly chosen:

- Single- Point Crossover: a random crossover point is selected and the tails of the two parents are swapped to get new off-springs.
- Double- Point Crossover: 2 random crossover points are selected, and the middle segments are swapped to get new off-springs.
- Uniform Crossover: treats each gene separately and decides which genes are inherited from the first parent, and which one are inherited from the second parent.

Let
$x_1 = (x_{11}, x_{12}, x_{13}, \dots, x_{1n})$ and
$x_2 = (x_{21}, x_{22}, x_{23}, \dots, x_{2n})$
be two parents and
$y_1 = (y_{11}, y_{12}, y_{13}, \dots, y_{1n})$ and
$y_2 = (y_{21}, y_{22}, y_{23}, \dots, y_{2n})$be the two offsprings of this crossover. We use the following formulae
$y_{1i} = \alpha_i . x_{1i} + (1 - \alpha_i). x_{2i}$
$y_{2i} = (1 - \alpha_i). x_{1i} + \alpha_i . x_{2i}$
Where
$\alpha = (\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n)$ & $\alpha_i \in \{0, 1\}$.
The crossover process practiced in GA for the continuous variables is similar to the binary uniform crossover process. Let
$x_1 = (x_{11}, x_{12}, x_{13}, \dots, x_{1n})$ and
$x_2 = (x_{21}, x_{22}, x_{23}, \dots, x_{2n})$ be two parents and
$y_1 = (y_{11}, y_{12}, y_{13}, \dots, y_{1n})$ and
$y_2 = (y_{21}, y_{22}, y_{23}, \dots, y_{2n})$be the two offspringsobtained in this crossover operation. This works by taking a set $\beta = (\beta_1, \beta_2, \beta_3, \dots, \beta_n)$ where $\alpha_i \in \{0, 1\}$ and by using the following formulae
$y_{1i} = \alpha_i . x_{1i} + (1 - \alpha_i). x_{2i}$
$y_{2i} = \alpha_i . x_{2i} + (1 - \alpha_i). x_{1i}$

2.3.2. Individual Feasibility Check

The crossover operation may result in solutions or chromosomes that are not feasible. It is therefore necessary to check and correct them. The basic verification rules are:

- no vessel rectangles overlap.
- all vessels fit on the berth.

Thereby, we propose 2D collision detection algorithm [12].

2.3.3. Mutation operators

A reproduction using only the crossover operator can be trapped in local optima. The genes of the children are limited by the genes of the parents, and if a gene is not present in the initial population (or if it disappears due to reproductions), it can never develop in the descendants. It consists of modify one or more genes of an individual selected by the selection operator. In this problem, we propose two mutation strategies for each kind of variables:

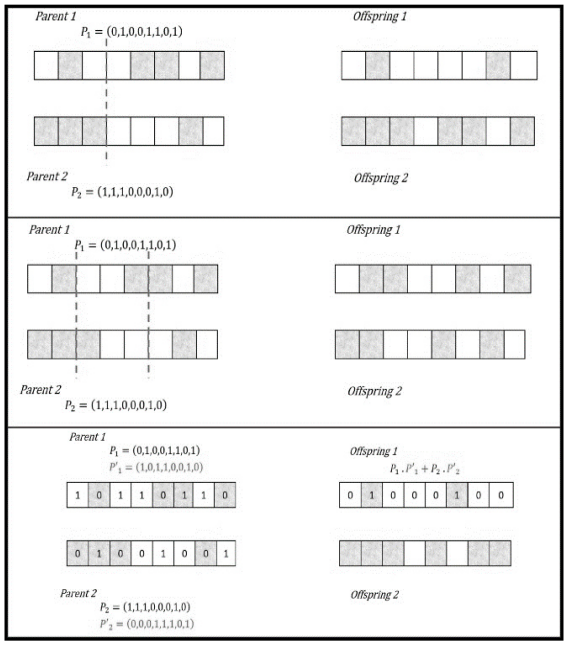

Figure 3: The figure show (from top to bottom) the single point crossover, the double point crossover and the uniform crossover process.

2.3.4. Mutation M1 (for binary variables)

Given the chromosome $x = (x_1, \dots, x_n)$. A random gene x_i ($\in \{0,1\}$) is selected from x , and a uniform random value is assigned to it. Let j be a

random integer index ∈ $\{1,\dots,n\}$. Mutation is carried out by

$$x_i' = \begin{cases} x_i & \text{if } i \neq j \quad i = 1,\dots,n \\ 1 - x_i & \text{if } i = j \end{cases}$$

2.3.5. Mutation M2 (for continuous variables)

Given the chromosome $x = (x_1,\dots,x_n)$. A random gene x_i ($\in \{0,1\}$) is selected from x. Let j be a random integer index ∈ $\{1,\dots,n\}$. The main mechanism of M2 consists of changing value by adding random noise drawn from normal distribution. M2 requires two parameters: the mean ξ and the standard deviation σ. Mutations then are realised by adding some Δxi to each xi, where the Δxi values are randomly drawn using the given $N(0,\sigma^2)$ distribution, with the corresponding probability density function.
$x_i' = x_i + N(0,\sigma^2)$

The mean of Δxi is equal to zero and the standard deviation is equal to σ^2 which represents the mutation step size.

3. Results and Discussion

To test the quality and performance of the GA and the proposed model, we consider two types of instances. The first type consists of a set of instances small random numbers that can be solved exactly by the optimization software CPLEX. Small instances are considered in this chapter to compare the results obtained by the GA method with those provided by CPLEX. The second type is based on a set of medium and large size randomly generated.

3.1. Experimental environment and setting parameters

The used approaches for the resolution of the small instances of this problem have been implemented with Java using NetBeans IDE 8.2. For large scale instances, GA has been implemented using MATLAB. By default, all experiments were conducted on a Intel® Core ™ i5-4570 CPU @ 3.20 GHz, RAM 4 GB. The final adjustment of the parameters of the proposed algorithm is shown in Table 1. The values presented in this chart are the result of several intensive studies we conducted to refine the GA. We note that the population size is set alternately at 25, 50, 100 and 150, which is an appropriate size for decision-makers to exchange solutions among the population. In our preliminary experiments, we tried to tune different combinations of probability of crossing (pc) and probability of mutation (pm) on a set of instances, while keeping the other parameters. For each instance and each combination, 20 independent analyses were performed. We can confirm that the effects of the mutation probability show that a small (pm) is likely to improve the values of the solutions obtained when (pc) is large. Therefore, we set pc= 0.9 and pm = 0.05 as final parameters.

Table 1: Parametric configuration of GA

Parameter description	Values
Population size (nPop)	$\in \{25,50,100,150\}$
Crossover Percentage (pc)	0.9
Extra Range Factor for Crossover (gamma)	0.4
Mutation Percentage (pm)	0.05
Mutation Rate 1 (mu)	0.05
Mutation Rate 2 (mub)	0.02
Maximum Number of Iterations	200

3.2. Instance generation

First, we report, in Table 3 (in Appendix), the parameters relating to the set of vessels V considered in this work. For each instance "DG" indicated in the first column, a number N of mixed-size vessels must be transported (N = N1 + N2, where N1 indicates the number of mother vessels and N2 the number of feeder vessels). The set of berths B, N1 and N2 are displayed in the second column. The third column indicates the values of a_i, d_i, f_i, h_i, p_i. The transshipment ship pairs in each instance are reported in the last column. Six sets of test problems were used, each containing 4 different generated instances. The first three of these sets include relatively small problems for a terminal with B = 12, and N = 10; 12; and 14 vessels respectively. The next three problem sets contain larger instances with B = 20 and with N = 20; 25; and 30 vessels. The same ranges of parameter values from [1] are used to control the generation of independent random parameter sets in the computational experiments (Table 2).

Table 2: Ranges of parameters used in the computational experiments

Parameters	Intervals	
h_i	[2,6]	
p_i	[1,4] if $h_i = 2$ [1,5] if $h_i \in \{3,4\}$ [1,6] if $h_i \in \{5,6\}$	
a_i	[1,10] for small instances	[1,20] for large instances
d_i	$a_i + K.p_i$	where $K \in$ [1,3]
f_i	[3,5]	

3.3. Results

According to Table 4, we can confirm the validity of our mathematical model proposed for all the small instances studied. In fact, the values obtained by minimizing f have reached the theoretical lower bound. We observe that CPLEX has enabled to optimize the majority of the cases of the small instances studied in a few seconds. Table 6 contains the identifier of the instances studied "Inst", the columns "CPLEX" & "GA" represent, respectively, the results obtained by the exact method and the meta-heuristic applied on the instances of Table 5.. The values under the heading "Obj Val" indicate the best solution found by these two methods. The CPU corresponding to the execution of each instance in seconds is reported in the "CPU(s)" columns. The values under the "Dev" header indicate the deviation of CPLEX and GA results from the best solution. The value of the corresponding objective function indicated in the "Obj Val" represents the optimal solution, or the best bound found within 3600 seconds. The results in Table 6 confirm the performance of the GA adapted to the proposed model. They clearly show the effectiveness of this approach in relation to small instances size, thus offering optimal solutions.

4. Conclusion and future work

This paper deals with a new mathematical model for the integration, of the Berth Allocation Problem (BAP) and the container transshipment problem. A mixed integer linear program is implemented in Netbeans (using Java language and CPLEX library) to schedule incoming feeders and mother vessels along the terminal quay to berthing sections and decide the best transshipment method to use (direct/indirect) for each couple of vessels. A MATLAB implementation of a genetic algorithm is also proposed to solve large-scale problem instances. Numerical experiments were accrued on a set of randomly generated instances. The numerical results are analyzed and compared. The designed solution approaches provides optimal solutions for small and medium size problems in a reasonable amount of time. Encouraging results have been obtained. There is still more work to be done: an application of the proposed genetic algorithm on large size instances of this problem must also be provided. An analytical comparison between the single point, double point and the uniform crossover will be encountered in the future as well. Finally, we intend to extend the model by considering multiple objectives. It would be worthwhile to integrate another decision problem such as the Yard Allocation Problem (YAP) or the Quay Crane Assignment Problem (QCAP) and to use other resolution strategies and metaheuristic approaches.

5. References

[1] Ak, A. (2008). Berth and quay crane scheduling: problems, models and solution methods. Georgia Institute of Technology.

[2] Zhen, L., Chew, E. P., & Lee, L. H. (2011). An integrated model for berth template and yard template planning in transshipment hubs. Transportation Science, 45(4), 483-504.

[3] Zhen, L., Wang, S., & Wang, K. (2016). Terminal allocation problem in a transshipment hub considering bunker consumption. Naval Research Logistics, 63, 529–548.

[4] Lee, D. H., Jin, J. G., & Chen, J. H. (2012). Terminal and yard allocation problem for a container transshipment hub with multiple terminals. Transportation Research Part E: Logistics and Transportation Review, 48(2), 516-528.

[5] Tao, Y., & Lee, C. Y. (2015). Joint planning of berth and yard allocation in transshipment terminals using multi-cluster stacking strategy. Transportation Research Part E: Logistics and Transportation Review, 83, 34-50.

[6] Schepler, X., Balev, S., Michel, S., Sanlaville. (2017). Global planning in a multi-terminal and multi-modal maritime container port. Transportation Research Part E: Logistics and Transportation Review, 100: 38-62.

[7] Ganji, S. R. S., Babazadeh, A., & Arabshahi, N. (2010). Analysis of the continuous berth allocation problem in container ports using a genetic algorithm. Journal of Marine Science and Technology, 15, 408–416.

[8] Lee, D. H., Chen, J. H., & Cao, J. X. (2010). The continuous berth allocation problem: A greedy randomized adaptive search solution. Transportation Research Part E, 46, 1017–1029.

[9] Park, K. T., & Kim, K. H. (2002). Berth scheduling for container terminals by using a sub-gradient optimization technique. Journal of the operational research society, 53(9), 1054-1062.
[10] Liang, C., Hwang, H., & Gen, M. (2012). A berth allocation planning problem with direct transshipment consideration. Journal of Intelligent Manufacturing, 23(6), 2207-2214.
[11] Prayogo, D. N., & Hidayatno, A. (2021, February). Bi-objective optimization model for integrated planning in container terminal operations. In IOP Conference Series: Materials Science and Engineering (Vol. 1072, No. 1, p. 012022). IOP Publishing.
[12] https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection

Appendix

Table 3: Description of small sized instances

Instance name & no	(N1- N2- B)	Parameters values	Transshipment vessels pairs
DG1	(4-6-12)	$a_i = [1,1,1,7,7,10,10,8,10,8]$ $h_i = [2,6,3,4,4,5,6,3,3,2]$ $p_i = [3,6,4,3,3,5,3,5,5,3]$ $f_i = [3,5,3,4,4,4,5,3,3,3]$ $d_i = [7,13,13,10,10,15,16,18,20,14]$	(1,2) , (1,3) , (2,3) , (4,5) , (6,9)
DG2	(4-6-12)	$a_i = [1,3,2,1,7,10,7,7,10,10]$ $h_i = [3,4,2,2,4,6,5,4,4,5]$ $p_i = [5,5,3,4,5,4,6,5,2,4]$ $f_i = [5,5,3,4,4,5,5,5,3,4]$ $d_i = [6855,12,14,13,12,12,14]$	(1,3) , (7,8)
DG3	(4-6-12)	$a_i = [1,1,3,1,3,5,7,10,10,10]$ $h_i = [2,3,4,6,6,6,5,6,2,5]$ $p_i = [3,2,4,2,2,5,5,6,2,6]$ $f_i = [3,3,3,3,3,4,4,5,3,5]$ $d_i = [4,3,7,3,5,10,12,16,12,16]$	(1,2)
DG4	(4-6-12)	$a_i = [1,2,4,1,1,4,6,9,9,8]$ $h_i = [2,2,2,5,2,6,4,5,3,3]$ $p_i = [2,2,2,3,3,4,3,3,3,2]$ $f_i = [3,3,3,4,3,5,4,5,3,4]$ $d_i = [7,4,6,4,7,8,9,15,12,10]$	(1,2) , (1,3) , (1,5) , (2,3) , (2,4) , (3,6) , (3,7) , (4,5) , (4,6) , (5,6) , (6,8) , (6,10) , (7,9) , (8,9)
DG5	(4-8-12)	$a_i = [1,1,1,7,7,10,10,8,10,8,5,5]$ $h_i = [2,6,3,4,4,5,6,3,3,2,4,5]$ $p_i = [3,6,4,3,3,5,3,5,5,3,5,6]$ $f_i = [3,5,3,4,4,4,5,3,3,3,4,5]$ $d_i = [7,13,13,10,10,15,16,18,20,14,10,11]$	(1,2) , (1,3) , (1,11) , (1,12) , (2,3) , (4,5), (6,9) , (12,6) , (8,1) , (9,1)
DG6	(4-8-12)	$a_i = [1,3,2,1,7,10,7,7,10,10,5,5]$ $h_i = [3,4,2,2,4,6,5,4,4,5,4,5]$ $p_i = [5,5,3,4,5,4,6,5,2,4,5,6]$ $f_i = [5,5,3,4,4,5,5,5,3,4,4,5]$ $d_i = [6,8,5,5,12,14,13,12,12,14,10,11]$	(1,2) , (1,3) , (1,11) , (1,12) , (2,3) , (4,5), (6,9) , (12,6) , (8,1) , (9,1)
DG7	(4-8-12)	$a_i = [1,1,3,1,3,5,7,10,10,10,3,7]$ $h_i = [2,3,4,6,6,6,5,6,2,5,6,5]$ $p_i = [3,2,4,2,2,5,5,6,2,6,4,3]$ $f_i = [3,3,3,3,3,4,4,5,3,5,5,5]$ $d_i = [4,3,7,3,5,10,12,16,12,16,7,10]$	(1,2) , (1,3) , (1,11) , (1,12) , (2,3) , (4,5), (6,9) , (12,6) , (8,1) , (9,1)
DG8	(5-9-12)	$a_i = [1,1,1,7,7,10,10,8,10,8,5,5,1,1]$ $h_i = [2,6,3,4,4,5,6,3,3,2,4,5,2,6]$ $p_i = [3,6,4,3,3,5,3,5,5,3,5,6,3,6]$ $f_i = [3,5,3,4,4,4,5,3,3,3,4,5,3,5]$ $d_i = [7,13,13,10,10,15,16,18,20,14,10,11,7,13]$	(1,2) , (1,3) , (1,11) , (1,12) , (2,1) , (2,3), (4,5) , (4,13), (6,9)
DG9	(5-9-12)	$a_i = [1,3,2,1,7,10,7,7,10,10,5,5,2,1]$ $h_i = [3,4,2,2,4,6,5,4,4,5,4,5,2,2]$ $p_i = [5,5,3,4,5,4,6,5,2,4,5,6,3,4]$ $f_i = [5,5,3,4,4,5,5,5,3,4,4,5,3,4]$ $d_i = [6,8,5,5,12,14,13,12,12,14,10,11,5,5]$	(1,2) , (1,3) , (1,11) , (1,12) , (2,1) , (2,3), (4,5) , (4,13), (6,9)
DG10	(5-9-12)	$a_i = [1,1,3,1,3,5,7,10,10,10,3,7,3,5]$ $h_i = [2,3,4,6,6,6,5,6,2,5,6,5,6,6]$ $p_i = [3,2,4,2,2,5,5,6,2,6,4,3,2,5]$ $f_i = [3,3,3,3,3,4,4,5,3,5,5,5,3,4]$ $d_i = [4,3,7,3,5,10,12,16,12,16,7,10,5,10]$	(1,2) , (1,3) , (1,11) , (1,12) , (2,1) , (2,3), (4,5) , (4,13), (6,9)

Table 4: Numerical results of the adaptation of the exact method to solve small size instances of the Table 3

Instance name	Total (root+branch &cut)	Solution status	Optimal value	Continuous variables	Direct Transshipment vessel pairs	Indirect Transshipment vessel pairs
DG1	0.14 sec. (56.33 ticks)	Optimal	68.99999999999639	b_i =[9,1,7,5,1,1,1,6,8,9] t_i =[5,1,1,7,710,15,10,15,8] c_i = [8,7,5,10,10,15,18,15,20,11]	(9,7)	(1,3), (1,4), (2,4), (3,2) (4,6), (6,10)
DG2	0.72 sec. (280.89 ticks)	Optimal	168.0	b_i =[1,5,9,9,1,5,6,1,7,1] t_i =[1,3,5,1,16,16,8,7,14,12] c_i =[6,8,8,5,21,20,14,12,16,16]	(1,4)	(2,3), (7,9)
DG3	0.16 sec. (78.49 ticks)	Optimal	128.000000000000216	b_i =[9,7,7,3,1,1,1,1,1,6] t_i =[1,4,6,1,3,5,12,17,10,10] c_i = [4,6,10,3,5,10,17,23,12,16]	(10,9)	(1,3), (2,4), (2,6), (3,8), (3,9), (4,6), (4,7), (5,3), (5,7), (5,10), (6,5), (6,8), (6,9), (7,3), (7,6), (8,4), (8,7), (8,10), (9,6)
DG4	0.14 sec. (72.23 ticks)	Optimal	37.0	b_i =[3,9,1,1,7,5,1,1,8,5] t_i =[5,2,4,1,1,4,7,10,9,8] c_i = [7,4,6,4,4,8,10,13,12,10]		(1,3), (1,4), (1,6), (2,4), (2,5), (3,2), (3,7), (3,8), (4,3), (4,6), (4,7), (5,2), (5,7), (6,4), (6,5) , (6,9), (7,4), (7,10), (8,7), (8,10), (9,8), (10,7)
DG5	26.95 sec. (21775.71 ticks)	Optimal	206.000000000000023	b_i =[5,1,8,1,1,1,5,1,8,4,7,6] t_i =[7,1,1,10,7,18,10,13,19,13,5,13] c_i = [10,7,5,13,10,23,13,18,24,16,10,19]	(2,3)	(1,3), (1,4), (1,12), (2,4), (4,6), (6,10), (9,7), (11,2), (12,2)
DG6	93.52 sec. (66906.63 ticks)	Optimal	330.000000000000114	b_i =[1,7,5,5,7,1,1,7,7,2,1,6,1,3,2,5] t_i =[15,11,19,8,13,15,6,20,15,11,19, 8,13,15,6,20] c_i = [6,8,5,9,20,1525,13,15,19,11,26]		(1,3), (1,4), (1,12), (2,3) , (2,4), (4,6), (6,10), (9,7), (11,2), (12,2)
DG7	0.72 sec. (425.34 ticks)	Optimal	268.0	b_i =[9,1,7,5,1,1,1,1,9,6,1,1] t_i =[3,1,6,1,3,18,12,23,10,12,5,9] c_i = [6,3,10,3,5,23,17,29,12,18,9,12]		(1,3), (1,4), (1,12), (2,3) , (2,4), (4,6), (6,10), (9,7),

						(11,2), (12,2)
DG8	2196.97 sec. (1903801.18 ticks)	Optimal	334.0000000000016	$\boldsymbol{b_i}$ =***[7,1,8,1,5,6,5,8,8,9,1,1,9,2]*** $\boldsymbol{t_i}$ =***[1,1,4,7,8,14,11,19,24,8,10,15,1,21]*** $\boldsymbol{c_i}$ = [***4,7,8,10,11,19,14,24,29,11,15,21,4,27,***]	(1,13)	(1,3) , (1,4) , (1,12), (2,4), (3,2), (4,6), (4,14), (6,10), (9,7), (11,2), (12,2), (14,5)
DG9	3205.16 sec. (2548575.38 ticks)	Optimal	383.00000000000153	$\boldsymbol{b_i}$ =***[1,5,5,9,6,5,1,1,7,1,1,6,7,9]*** $\boldsymbol{t_i}$ =***[1,5,2,5,22,12,20,11,10,16,2,1]*** $\boldsymbol{c_i}$ = [***6,10,5,9,27,16,26,16,12,20,11,22,5,5***]	(2,4)	(1,3), (1,4), (1,12), (1,13), (3,2) , (4,6) , (4,14) , (6,10) , (9,7) , (11,2) , (12,2) , (14,5)
DG10	3.17 sec. (1777.45 ticks)	Optimal	452.9999999999999	$\boldsymbol{b_i}$ =***[7,1,7,4,1,5,1,1,9,6,1,1,1,5]*** $\boldsymbol{t_i}$ =***[3,1,6,1,3,30,14,24,10,12,7,11,5,19]*** $\boldsymbol{c_i}$ = [***63,10,3,5,35,19,30,12,18,11,14,7,24***]	(2,4)	(1,3), (1,4), (1,12), (1,13), (3,2) , (4,6) , (4,14) , (6,10) , (9,7) , (11,2) , (12,2) , (14,5)

Table 5: Description of medium sized instances

Instan-ce	(N1, N2, B)	Parameters values	Transshipment vessels pairs
Inst1	(9, 11, 20)	a_i = [1,6,7,2,1,6,1,4,9,10,12,13,9,17,17,17,17,20,20,20] h_i = [4,5,3,2,6,4,6,6,3,4,6,4,3,4,3,55,5,3] p_i = [4,5,3,4,5,5,3,5,3,5,5,4,4,3,3,5,3,6,6,5] f_i = [3,4,3,3,5,4,3,4,3,4,5,4,4,3,3,4,4,4,4,3] d_i = [5,11,10,6,6,12,4,9,12,16,17,17,13,20,20,22,20,26,26,26]	(2,4) , (2,6) , (14,15), (15,16), (16,13)
Inst2	(13,12, 20)	a_i = [1,1,1,1,1,1,4,4,4,4,4,4,7,7,7,7,7,7,10,10,10,10,10,10,13] h_i = [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3] p_i = [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3] f_i = [4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4] d_i = [10,10,10,10,10,10,7,7,7,7,7,7,10,10,10,10,10,10,13,13,13,13,13,13,16,]	(1,7) , (2,3)
Inst3	(17,13, 20)	a_i = [1, 1, 1, 1, 1, 2, 4, 4, 4, 4, 4, 4, 7, 7, 7, 7 ,7 ,7, 10, 10, 10, 10, 10 ,10 ,13, 13, 13, 13, 13, 13] h_i = [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,] p_i = [3, 3 ,3 ,3 ,3 ,2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3 ,3 ,3 ,3, 3, 3, 3, 3, 3] f_i = [4 ,4, 4, 4, 4, 4, 4, 4, 4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4 ,4, 4, 4, 4, 4, 4, 4, 4] d_i = [10, 10, 10 ,10, 10, 10, 7, 7, 7, 7 ,7 ,7 ,10, 10, 10, 10, 10, 10, 13, 13, 13, 13, 13, 13, 16, 16, 16, 16,	(1,7) , (2,3)
Inst4	(17,13, 20)	a_i = [1,1,1,1,1,1,1,1,1,3,3,3,3,3,3,3,3,3,5,5,5,5,5,5,5,5,9,9,9,9,9,9,9,9,9,13,13,13] h_i = [2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2] p_i = [2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2] f_i = [3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3] d_i = [5,3,3,3,3,3,3,3,3,7,7,7,7,7,7,7,7,7,11,11,11,11,11,11,11,11,11,15,15,15]	(1,20)

Table 6: Numerical results of the adaptation of the exact method and the GA on the instances of the Table 5

Instance no & name	CPLEX			GA		
	Obj Val	CPU(s)	Dev	Obj Val	CPU(s)	Dev
Inst1	87.0	0.98	0	87	5.609928	0
Inst2	90.0	0.16	0	90	1.149060	0
Inst3	75.0	0.16	0	75	0.607894	0
Inst4	60	0.17	0	64.2296	2.996758	4.2296