Genetic algorithm to optimize unloading of large containers vessel in port of Tripoli-Lebanon

A. SKAF[1,2], S. LAMROUS[1], Z. HAMMOUDAN[2], M.-A. MANIER[1]
[1] Univ. Bourgogne Franche-Comté, FEMTO-ST Institute/CNRS, (UTBM), Belfort, 90010, France
[2] Univ. Libano-Française (ULF), Tripoli, Lebanon
(ali.skaf@utbm.fr, sid.lamrous@utbm.fr, zakaria.hammoudan@gmail.com, marie-ange.manier@utbm.fr)

Abstract— **The quay crane scheduling problem (QCSP) in the lebanese port (Tripoli) is discussed in this study. It consists in assigning each crane to a set of bays for a given vessel, while sequencing the unloading of these bays. In a previous study, we discussed two exacts methods whose main drawback is the difficulty to obtain results for large instances. That's why, we propose a genetic algorithm which enables us to overcome this and to get quickly near optimal solutions. We have tested and validated our method on real instances from the port of Tripoli.**

Keywords – **optimization, quay crane, scheduling problem, genetic algorithm, case study**

I. Introduction and related work

Port of Tripoli is the second port in Lebanon, it receives about 400 ships every year and has only two quay cranes. In 2018 for the first time, the port receive big containers vessels loaded by more than 2000 containers. Quay cranes serve to unload or load containers from the vessel to the store or to the trucks. Like the other ports, one of the challenges in the port of Tripoli-Lebanon is to address the Quay Crane Scheduling Problem (QCSP), subject to non-interference constraints for the quay cranes, assignments conditions with bays, and positioning problem. Fig.1 is an illustration of the quay cranes and the containers while Fig.2 is an illustration of the containers vessels and quay cranes assignment.

In a previous study, we proposed exact methods to solve the QCSP with non-interference constraints in the port of Tripoli-Lebanon. But solving large instances spends long execution times and reaches the out of memory exception. So the objective of this paper is to reduce the execution time by using a metaheuristic approach.

Several researchers addressed the QCSP in the literature. [8] Moghaddam et al. (2008) presented a novel, mixed integer programming (MIP) model for the quay crane scheduling and assignment problem, after that [11] Wang et al. (2009) determined the sequence of unloading operation, with the objective to minimize the weighted operation time of jobs and travel time.

[4] Chung et al. (2012) proposed a modified genetic algorithm to deal with the problem and to test the optimization reliability of the proposed algorithm. A set of well-known benchmarking problems was solved, meanwhile [13] Yi et al. (2012) aimed to minimize the total handle time of all tasks, using a polynomial time algorithm to solve the problem.

[3] Azza et al. (2014) founded the handling sequence of tasks at ship; their objective is to minimize the time spent by the vessel at berth and minimize the total cost. Mixed-integer linear programming and ant colony algorithm were used in the previous papers to solve the problem. [6] Liu et al. (2015) proposed models and algorithm for the general double-cycling problem with internal-reshuffles, where reshuffle containers are allowed to move directly from one stack to another. A branch-and-price framework is used to solve the problem.

[7] Liu et al. (2016) studied a scheduling problem with two uniform quay cranes; their objective was to minimize the turn-around time of a vessel, then [1] Alnaqbi et al. (2016) aimed to minimize the latest starting time for the vessel and its handling time. They used Mixed-integer linear programming and hybrid algorithm to solve the problem, meanwhile [2] Awar et al. (2016) proposed a solution to minimize the entire processing time for all vessels, mixed-integer linear algorithm is used to solve the problem.

[5] Haoyuan et al. (2017) proposed a simulation model of container terminal, in order to solve its quay crane scheduling problem, with the objective to optimize the shortest total delay time for all vessels, [9] Salhi et al. (2017) elaborated a model that combines three distinct problems, namely berth allocation, quay crane assignment, and quay crane scheduling that arises in container ports, genetic algorithm was developed to solve the problem and finally, [12] Xiazhong et al. (2017) aimed to find the optimal storage location of container groups while minimizing both the maximum completion time and the traveling time of trucks. They used mixed-integer linear programming and tabu search to solve the problem.

This brief review suggests that methods such as genetic algorithms are adapted to solve the Quay Crane Scheduling Problem (QCSP).

In the previous work, [10] Skaf et al. (2018) created a mixed-integer programming (MILP) model solved with CPLEX, and a dynamic programming (DP) algorithm to solve the addressed problem. Our dynamic programming algorithm finds all possible choices for the quay cranes-bays assignment while taking into consideration the non-interference constraints between quay cranes. It allows to find the optimal completion time for a containers vessel. We observed that dynamic programming provided better results than MILP in terms of CPU time and completion time for large number

of bays. Nevertheless, the inconvenient of the DP is that for larger instances, per example for 40 bays, it may takes 5 hours to give a result. For such instances, we propose in this paper a genetic algorithm to obtain optimal or near optimal results with a quick execution time, for the quay cranes scheduling problem without cranes crossing.
Later on in this paper, section 2 proposes a mathematical formulation (MILP). Section 3 provides a detailed explanation of the genetic algorithm, section 4 provides the experimental results obtained with this one, that we compare with the dynamic programming results. Finally, section 5 gives a conclusion and some perspectives.

Fig. 1. Quay cranes and containers

Fig. 2. Containers vessel

II. Mathematical Formulation

A. Assumptions

The quay crane problem we have studied has the following characteristics: containers unloading is in a single vessel, all quay cranes move on the same track and need to work without any interference between them. A crane is assigned to a single bay at a time and it cannot work in another bay until it completes unloading containers in the current bay (no preemption). Finally, each bay is handled by at most one quay crane at a time. In this study, the quay cranes travel time between bays is ignored as it is very short compared to the unloading times.

B. Notations

Q : number of quay cranes
i : index for quay crane (i=1 to Q)
B : number of bays
j : index for bay (j=1 to B)
C_j : number of containers in bay j
Tc : time to unload a container by a quay crane and put it in the storage. It is supposed to be the same for all containers in all bays.
M : big integer

C. Decision variables

$x_{(j,i)}$	$= 1$	if bay j is handled by quay crane i
	$= 0$	otherwise
$z_{(j,j')}$	$= 1$	if the unloading of bay j finishes before starting to unload bay j'
	$= 0$	otherwise
t_j		completion time of bay j
C_{max}		makespan

D. Modeling

The following is the mixed integer linear formulation we have presented in a previous work ([10] Skaf et al. (2018)). Here we use some of those equations in the fitness calculation of the genetic algorithm which we will talk about later.

Objective

$$Minimize \quad C_{max} \tag{1}$$

Subject to

$$\sum_{i=1}^{Q} x_{(j,i)} = 1 \quad \forall j \in B \tag{2}$$

$$t_j \geq (Tc * C_j) \quad \forall j \in B \tag{3}$$

$$t_j - t_{j'} + (Tc * C_{j'}) + z_{(j,j')} * M > 0 \quad \forall j, j' \in B \tag{4}$$

$$t_j - t_{j'} + (Tc * C_{j'}) - (1 - z_{(j,j')}) * M \leq 0 \quad \forall j, j' \in B \tag{5}$$

$$\sum_{i=1}^{Q} i * x_{(j,i)} + 1 \leq \sum_{i'=1}^{Q} i' * x_{(j',i')} + (z_{(j,j')} + z_{(j',j)}) * M$$

$$\forall j, j' \in B, j < j' \tag{6}$$

$$C_{max} = \max_j \ t_j \tag{7}$$

$$x_{(j,i)} = [0,1] \quad \forall j \in B, \forall i \in Q \tag{8}$$

$$z_{(j,j')} = [0,1] \quad \forall j, j' \in B, j < j' \tag{9}$$

Equation (1) is the objective function which serves to minimize the latest completion time among all bays. Constraint (2) ensures that every bay must be handled only by one quay crane. Constraint (3) ensures that the completion time is bigger than the working time in each bay (Working time = number of containers * time needed to unload a container and store it). Constraint (4) shows when bay j finishes before bay j' starts and constraint (5) shows the opposite. Constraint (6) avoid the interference between the quay cranes and finally constraint(7) defines the C_{max} value.

III. GENETIC ALGORITHM

Due to the difficulty to solve large instances, we propose in this work a genetic algorithm (GA). A genetic algorithm (GA) is a heuristic search algorithm that was first proposed by [Holland, 1975] and developed and applied by [Goldberg, 1989]. It can be easily coded and often gives good solutions. Roulette wheel selection, order crossover and swap mutation are used due to their results efficacy.
The steps of the genetic algorithm are provided as follows : first we generate an initial population of solutions, then for all generations we should make a test about non-interference constraints of quay cranes in other words we want to know if quay cranes work without crossing. If yes, we calculate the fitness value, otherwise we set the fitness value zero. After that, if the current generation is the last one, then the program is ended, otherwise we should execute the genetic algorithms steps such as roulette wheel selection, order crossover and swap mutation.

A. Chromosome representation

In a genetic algorithm, a solution is called a chromosome. In our case, it is composed of series of bays. Table I provides a chromosome and the bay number represents a gene.

TABLE I
CHROMOSOME REPRESENTATION

<u>6</u>	9	1	10	3	8	7	2	4	5

For example, the second element (second gene) in the chromosome represents the bay number 9.

B. Assignment of initial positions for quay cranes

We propose to fix the initial position of each quay crane i to the position of bay 1+(i-1)*D, (i ∈ Q) with D is a random integer between 1 and B/Q (if B/Q is a rational number, we take its natural part number). It allows a homogeneous initial distribution of the cranes along the vessel.
Let us note that with this formula, the first quay crane position should always be at the first bay.
Example: Let us suppose that we have 10 bays and 3 quay cranes, then D should be a random number between 1 and 10/3 (hence between 1 and 3). If D=3, then the main position of quay crane 1 is on bay 1, quay crane 2 is on bay 4 and quay crane 3 is on bay 7.

C. The scheduling of quay cranes

The quay crane scheduling works as explained in Algorithm 1. After determining the initial positions of the cranes, the bays are assigned to the cranes. We use a sequential procedure which considers the bays in the order of these ones in the associated chromosome.
The four following main steps are distinguished. For each one we illustrate it with the previous numerical example (with the given chromosome and the initial position assignment) with 3 quay cranes and 10 bays.

Algorithm 1 Quay cranes scheduling

```
if $QC\ nb$ = $1$ then
    QC $\leftarrow bay_j$;
    $bay_j \leftarrow -1$;
    j $\leftarrow$ j+1;
    $CP_{QC} \leftarrow CP_{QC} + t_{bay_j}$;
end
else
    if $CP_{QC_i}$ Not in $\{CP_{QC}\}/CP_{QC_i}$ then
        $CP_{QC_i}$ = Min$\{CP_{QC}\}$;
        $bay_j \leftarrow -1$;
        j = Next_bay_Index({bay});
        $CP_{QC_i} \leftarrow CP_{QC_i} + t_{bay_j}$;
    end
    else
        if $\Delta(bay) > 0$ then
            $QC_i \leftarrow bay_j$ with shorter distance;
            $bay_j \leftarrow -1$;
            j = Next_bay_Index({bay});
            $CP_{QC_i} \leftarrow CP_{QC_i} + t_{bay_j}$;
        end
    end
    else
        $QC_i \leftarrow bay_j$ with smaller index number;
        $bay_j \leftarrow -1$;
        j = Min_bay_Index({$bay_j$});
        $CP_{QC_i} \leftarrow CP_{QC_i} + t_{bay_j}$;
    end
end
```

QC : Quay crane, CP_{QC} : completion time of the quay crane
t_{bay_j} : time that quay crane took to finish working in bay j
$\Delta = |bay_i - bay_k|$
Next_bay_Index{bay} : function to get the next bay
Min_bay_Index{bay} : function to get the smaller bay number

In this example, the main completion time of the three quay cranes is initialized to 0, because no unloading operation has been performed until now (iteration init in Table II).
In the chromosome, we consider the current bay (current gene) which corresponds to an unassigned bay, and we assign it to a crane. For example, let us take the first gene of the chromosome (bay 6), and we suppose the processing time for all bays is 10.
1. Non-Crossing : The first step is a test of interference which identifies the two cranes which are candidate to unload the bay in the current gene, i.e. we keep the cranes whose last position is immediatly at the left or at the right of this current bay. For our example, quay crane 2 and quay crane 3 can handle bay 6 without interference with quay crane 1.
2. Makespan : Among these two cranes, we choose the one which is available sooner. In case of equality, that means if the two cranes have the same completion time, then we go to step 3 to discriminate them according to a distance criterion. In our example, the completion times of quay crane 2 and quay crane 3 are equal.

3. Distance : If no crane has been chosen in step 2 with the time criterion, we assign the current bay to the nearest crane. In case of equality (if the two cranes are at equal distance from the bay), we choose the crane at the left (with the smallest index). The distance between bay 6 and bay 4 (which quay crane 2 works inside) is 1 bay and the distance between bay 6 and bay 7 (which quay crane 3 works inside), is 0 bay. Then crane 3 is nearer from bay 6 than crane 2.
4. Assignement : Assign bay 6 to quay crane 3. Update the position of quay crane 3 and the associated completion time (see iteration 1 in Table II). Then update the current gene (following one) in the chromosome (gene 2 then bay 9 in our example). Repeat the four steps until all bays are assigned to one quay crane.

TABLE II
DETAILED EXAMPLE

Iteration		QC1	QC2	QC3
	bay position	1	4	7
init	partial scheduling	$\emptyset$	$\emptyset$	$\emptyset$
	completion time	0	0	0
	bay position	1	4	6
1	partial scheduling	$\emptyset$	$\emptyset$	6
	completion time	0	0	10
	bay position	1	4	9
2	partial scheduling	$\emptyset$	$\emptyset$	6, 9
	completion time	0	0	20
	bay position	1	4	9
3	partial scheduling	1	$\emptyset$	6, 9
	completion time	10	0	20
	bay position	1	4	10
4	partial scheduling	1	$\emptyset$	6, 9, 10
	completion time	10	0	30
	bay position	1	3	10
5	partial scheduling	1	3	6, 9, 10
	completion time	10	10	30
	bay position	1	8	10
6	partial scheduling	1	3, 8	6, 9, 10
	completion time	10	20	30
	bay position	7	8	10
7	partial scheduling	1, 7	3, 8	6, 9, 10
	completion time	20	20	30
	bay position	2	8	10
8	partial scheduling	1, 7, 2	3, 8	6, 9, 10
	completion time	30	20	30
	bay position	2	4	10
9	partial scheduling	1, 7, 2	3, 8, 4	6, 9, 10
	completion time	30	30	30
	bay position	2	5	10
10	completed scheduling	1, 7, 2	3, 8, 4, 5	6, 9, 10
	completion time	30	40	30

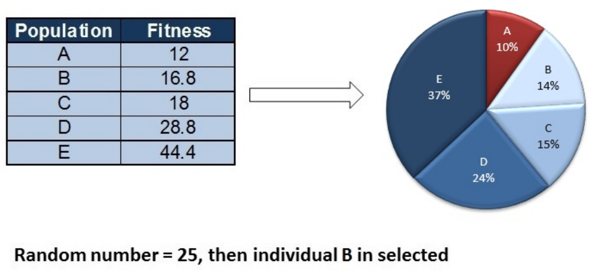
D. Fitness

The previous procedure prevents the interference between quay cranes, but nevertheless all quay cranes must be checked if they satisfy the constraints or no. Constraints (4) and (5) can be used and then the quay crane scheduling can be checked if it satisfies (6), if yes then the fitness is as shown in Eq.(10) else it will be set to zero.

$$Fitness = 1/C_{max} \tag{10}$$

E. Roulette wheel selection

Selection is a process in which individuals from a population are chosen according to the values of their cost or "fitness" function to form a new population. Individuals evolve through successive iterations of selection, called generations. Each individual is selected proportionally to their "fitness" function, so an individual with a higher fitness function will be more likely to be selected than another with a lower fitness value. This function can be considered as a measure of profit or quality that one wishes to maximize. A simple operator of selection is the technique of the weighted roulette where each individual of a population occupies a surface of the roulette proportional to the value of its function "fitness". For reproduction, candidates are selected with a probability proportional to their "fitness". For each selection of an individual, a simple rotation of the wheel gives the selected candidate. Roulette Wheel Selection works as follows :
Calculate the sum of all fitness, then generate a number between 0 and the sum, after that add the fitness values to a partial sum X starting from the top of population, and finally the chosen chromosome is the first chromosome for which X overpass the random number.
The procedure is illustrated in Fig.3.



Population	Fitness
A	12
B	16.8
C	18
D	28.8
E	44.4

Random number = 25, then individual B in selected

Fig. 3. Roulette wheel selection

F. Order crossover

Order crossover works as follows:
Select a random substring from one parent randomly, then create an offspring by copying the selected substring in their corresponding positions. After that, delete from the second parent all existing in the substring and place the genes in the empty positions of the offspring from left to right. Finally we should take in consideration the order of the sequence of the offspring creation.
The procedure is illustrated in Fig.4.

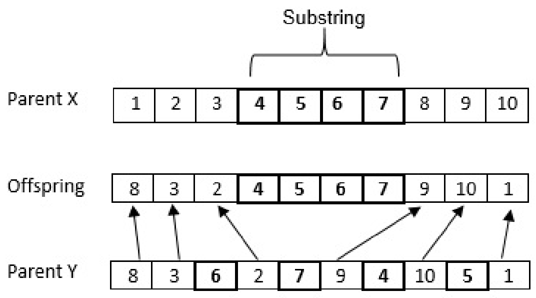


Fig. 4. Offspring

G. Swap mutation

In the mutation all individuals are tested bit by bit. It selects two positions on the chromosome randomly, then swaps the values on these positions.
The procedure is illustrated in Fig.5.

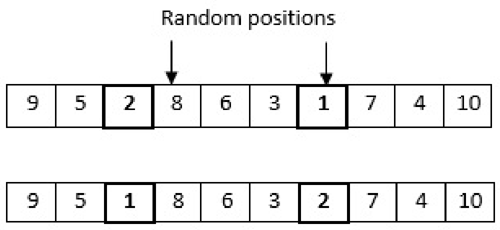


Fig. 5. Swapped bays

IV. EXPERIMENTAL RESULTS

A. Results comparison between dynamic programming and genetic algorithm

In the previous work, we solved some instances using CPLEX and Dynamic programming, small and large sizes instances were generated randomly, and containers number in each bay is also generated in a random way between 10 and 50 containers. In this paper we add the genetic algorithm results with CPU time, as shown in Table III and Table IV, results comparison is made for small and large instances with CPU time and with the GAP for makespan and CPU time.

According to the primary tests, the population size, the probability of mutation, the probability of crossover and the limit of generations are 300, 0.2, 0.25 and 1000 respectively, in these computational experiments.

Programs are executed in a HP laptop core i5 CPU 2.50 GHZ with 12GB RAM, 250GB SSD, 2GB RAM dedicated VGA, running in windows 8 professional.

TABLE III
SMALL INSTANCES

No.	(B x C)	Makespan DP (mins)	Makespan GA (mins)	GAP (%)	CPU Time DP (s)	CPU Time GA (s)
1	4 x 2	43.29	43.29	0	< 1	< 1
2	4 x 3	29.25	29.25	0	< 1	< 1
3	5 x 2	69.03	69.03	0	< 1	< 1
4	5 x 3	57.33	57.33	0	< 1	< 1
5	6 x 2	95.94	95.94	0	< 1	< 1

B x C : bays number x cranes number

Table III shows that both dynamic programming and genetic algorithm attained the optimal solution for the small instances and their CPU time is different but acceptable and realistic for the small instances.

TABLE IV
LARGE INSTANCES

No.	(B x C)	Makespan DP (mins)	Makespan GA (mins)	GAP1 (%)	CPU Time DP (s)	CPU Time GA (s)	GAP2 (%)
1	12 x 2	171.99	173.16	0.68	< 1	< 1	44.07
2	12 x 3	115.83	117.12	1.1	< 1	< 1	25
3	13 x 2	184.86	184.86	0	1.03	< 1	46.6
4	13 x 3	124.02	126.36	1.85	< 1	< 1	50.51
5	14 x 2	201.79	204.75	1.45	2.36	< 1	66.52
6	14 x 3	128.7	129.92	0.94	2.45	< 1	64.49
7	15 x 2	226.13	231.66	2.39	2.78	< 1	70.14
8	15 x 3	139.89	143.91	2.8	3.16	< 1	71.2
9	16 x 2	215.28	218.79	1.6	3.57	< 1	72.83
10	16 x 3	152.1	154.44	1.52	4.83	1.23	74.53
11	17 x 2	221.13	223.3	0.97	5.41	1.38	74.49
12	17 x 3	156.78	159.12	1.47	10.95	2.58	76.44
13	18 x 2	249.21	253.89	1.84	12.97	2.66	79.49
14	18 x 3	168.48	168.48	0	36.57	5.03	86.25
15	19 x 2	264.42	265.59	0.44	47.53	9.39	80.24
16	19 x 3	180.18	182.52	1.28	143.68	11.98	> 90
17	20 x 2	271.44	273.78	0.85	173.04	14.15	> 90
18	20 x 3	187.2	189.54	1.23	579.76	25.17	> 90
19	21 x 2	279.63	280.8	0.42	679.001	36.13	> 90
20	21 x 3	194.22	197.73	1.78	2313.81	31.18	> 90
21	22 x 2	285.48	288.99	1.21	2932.66	43.88	> 90
22	22 x 3	201.24	203.58	1.15	14862.35	44.58	> 90
23	23 x 2	NA	297.18	-	NA	56.13	-
24	23 x 3	NA	209.43	-	NA	59.58	-

B x C : bays number x cranes number
NA : Interrupt execution time after 5 hours. (No results)
GAP = ((DP makespan - GA makespan)/DP makespan) *100

As shown in Table IV, starting from instance 23 and beyond, dynamic programming was unable to provide solution after more than 5 hours of CPU processing time, meanwhile genetic algorithm provides better solution in a very reduced time.
The GAP between the optimal solutions and the near optimal solution is between 0% and 2.8%, so we can say that the proposed genetic algorithm is successful and practical for the mentioned problem.

B. Results stability of Genetic Algorithm

As shown in Table V, for the small instances and after 5 executions, the same results appears.

TABLE V

STABILITY ANALYSIS FOR SMALL INSTANCES

Execution	4x2	4x3	5x2	5x3
1	43.29	29.25	69.03	57.33
2	43.29	29.25	69.03	57.33
3	43.29	29.25	69.03	57.33
4	43.29	29.25	69.03	57.33
5	43.29	29.25	69.03	57.33

In Table VI, for the large instances, after 5 executions for the same instance, minimum 2 executions have the same near optimal result, with a percentage of 40% while the 3 others give results more or less 3% away from the near optimal result. Example for 10 bays and 2 quay cranes, first, third and fourth execution give the near optimal solution obtained by the genetic algorithm = 136.23, while the second and the fifth execution give results more or less 3% away from the near optimal solution.

Percentage calculation for second execution:
139.74 - 136.23 = 3.51
(3.51/139.74) * 100 = 2.51% from the near optimal result.

TABLE VI

STABILITY ANALYSIS FOR LARGE INSTANCES

Execution	10x2	12x3	16x2
1	136.23*	117.12*	218.79*
2	139.74	120.63	218.79*
3	136.23*	119.46	219.96
4	136.23*	119.46	218.79*
5	140.91	117.12*	221.13

*: near optimal solution obtained by the GA

C. Results comparing to port of Tripoli-Lebanon

TABLE VII

DP RESULTS VS PORT RESULTS

N.	Cranes	Bays	Containers	Port results	DP	Saved time %
1	2	6	25	40	19.5	51.25
2	2	4	62	95	44.78	52.86
3	2	3	90	103	70.2	31.84
4	2	4	84	96	49.14	48.81
5	2	5	600	480	351	26.88

In Table VII the mentioned port results are compared to the dynamic programming (DP) results while in Table VIII they're compared with genetic algorithm (GA) results.
In the first example, two quay cranes and six bays containing in total 25 containers, the time to unload a container by a quay crane is about 1.17 minutes, DP result is 19.5 minutes, then in the port, the unloading time of 25 containers is about 40 minutes, finally the DP enables us to save time by 51.25%, while GA result is 21.84 minutes and it enables to save time by 45.4%. Although the percentage of the saved time by GA is smaller than DP percentage, so the GA results are good and can be applied. The model accommodates the true conditions of the port.

TABLE VIII

GA RESULTS VS PORT RESULTS

N.	Cranes	Bays	Containers	Port results	GA	Saved time %
1	2	6	25	40	21.84	45.4
2	2	4	62	95	45.95	51.63
3	2	3	90	103	72.54	29.57
4	2	4	84	96	52.01	45.82
5	2	5	600	480	355.68	25.9

V. CONCLUSION

On large instances, this paper shows the premises of the QCSP solving by metaheuristics. The results show that the proposal is promising to have quickly good solutions. Here, the first attempt explores a search space with a genetic algorithm. To further improve the quality of solutions in terms of execution time and convergence towards the optimal solution, other types of metaheuristics deserve to be tested. A large-scale comparison of benchmarks in the literature is obvious to the following of this work.

REFERENCES

[1] B. Alnaqbi, H. Alrubaiai, and S. A. Alawi, "Combination of a dynamic-hybrid berth allocation problem with a quay crane scheduling problem," *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2016.

[2] K. A. Awar, M. Alawani, and S. A. Jaberi, "A multi-vessel quay crane scheduling problem," *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2016.

[3] L. Azza, M. E. merouani, and A. Medouri, "Ant colony system for solving quay crane scheduling problem in container terminal," *2014 International Conference on Logistics Operations Management*, 2014.

[4] S. Chung and K. Choy, "A modified genetic algorithm for quay crane scheduling operations," *Expert Systems with Applications*, 2012.

[5] L. Haoyuan and S. Qi, "Simulation-based optimization on quay crane scheduling of container terminals," *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017.

[6] M. Liu, S. Wang, and C. Chu, "A branch-and-price framework for the general double-cycling problem with internal-reshuffles," *2015 IEEE 12th International Conference on Networking, Sensing and Control*, 2015.

[7] M. Liu, F. Zheng, Y. Xu, and C. Chu, "Approximation algorithm for uniform quay crane scheduling at container ports," *Discrete Mathematics, Algorithms and Applications*, 2016.

[8] R. T. Moghaddam, A. Makui, S. Salahi, M. Bazzazi, and F. Taheri, "An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports," *Computers & Industrial Engineering*, 2008.

[9] A. Salhi, G. Alsoufi, and X. Yang, "An evolutionary approach to a combined mixed integer programming model of seaside operations as arise in container ports," *ADVANCES IN THEORETICAL AND APPLIED COMBINATORIAL OPTIMIZATION*, 2017.

[10] A. Skaf, S. Lamrous, Z. Hammoudan, and M.-A. Manier, "Exact method for single vessel and multiple quay cranes to solve scheduling problem at port of tripoli-lebanon," *2018 International Conference on Industrial Engineering and Engineering Management*, 2018.

[11] S. Wang and W. Hu, "Multi quay crane scheduling problem based on aco in container terminals," *2009 International Conference on Management and Service Science*, 2009.

[12] C. Xiazhong, Z. Ye, and H. Hongtao, "Optimization research of joint quay crane scheduling and block selection in container terminals," *2017 International Conference on Service Systems and Service Management*, 2017.

[13] D. Yi, L. GuoLong, and L. ChengJi, "Model and heuristic algorithm for quay crane scheduling at container terminal," *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, 2012.