IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 10, No. 4, December 2021, pp. 919~929
ISSN: 2252-8938, DOI: 10.11591/ijai.v10.i4.pp919-929

A self-adaptation algorithm for quay crane scheduling at a container terminal

Esam Taha Yassen[1], Masri Ayob[2], Alaa Abdalqahar Jihad[3], Mohd Zakree Ahmad Nazri[4]
[1,3]Computer Center, University of Anbar, Al Anbar, Iraq
[1,2,4]Data Mining and Optimization Research Group (DMO), Centre for Artificial Intelligent (CAIT), Universiti Kebangsaan Malaysia, Selangor, Malaysia

Article Info

Article history:

Received Feb 24, 2021
Revised Jun 5, 2021
Accepted Aug 20, 2021

Keywords:

Adaptive selection mechanism
Local search algorithms
Quay crane scheduling
Self-adaptation heuristic

ABSTRACT

Quay cranes scheduling at container terminals is a fertile area of study that is attracting researchers as well as practitioners in different parts of the world, especially in OR and artificial intelligence (AI). This process efficiency may affect the accomplishment and the competitive merits. As such, four local search algorithms (LSs) are utilized in the current work. These are hill climbing (HC), simulated annealing (SA), tabu search (TS), and iterated local search (ILS). The results obtained demonstrated that none of these LSs succeeded to achieve good results on all instances. This is because different QCSP instances have different characteristics with NP-hardness nature. Therefore, it is difficult to define which LS can yield the best outcomes for all instances. Consequently, appropriate LS selection should be governed by the type of problem and search status. The current work proposes to achieve this, the self-adaptation heuristic (self-H). The self-H is composed of two separate stages: The upper (LS-controller) and the lower (QCSP-solver). The LS-controller embeds an adaptive selection mechanism to adaptively select which LS is to be adopted by the QCSP-solver to solve the given problem. The results revealed that the self-H outperformed others as it attained better results over most instances and competitive results.

Corresponding Author:

Esam Taha Yassen
Computer Center
University of Anbar
Al Anbar, Iraq
Email: co.esamtaha@uoanbar.edu.iq

1. INTRODUCTION

Container terminals are regarded as one of the most significant modern sea freights means of transportation, as they represent the essential connection between local markets and global customers and contrariwise [1]. Thus, their effectiveness is essential to the achievement of the supply chain. In accordance with the speedy yearly growth of transported container volume, no wonder that container terminal now represents one of the predicaments in the international chain of supply [2], and container terminal efficiency a significant issue for liner transportation corporations as they work on reducing their cost [3]. The primary objective of container terminals is to attain fast movement of containers at the minimum cost possible [4]. Therefore, the time the vessel requires to load or unload is generally the terminal's uppermost precedence; the mooring time of a vessel is its turn time [5].

Quay crane scheduling problem (QCSP) is among the most significant operations related to seaport terminals. Its aim is to identify the order of tasks to be made by each quay crane (QC) [1]. Improving the efficiency of this operation will improve the vessel turn time [6]. Concerning QCSP, there is a common

belief that the allocations of cranes to vessels are already accomplished. All QCs are made to move on a railway line that runs in a parallel direction with the vessel where each QC cannot cross over to other QCs. The vessel is divided in a longitudinal way to produce several bays where the containers are stored. Containers that are stored in these bays usually are aligned together with their features and criteria, such as weight, size, and origin and destination ports [1], [5]. Based on Meisel and Bierwirth, QCSP is classified into three basic problems that are most prevalent in the research community [7]:

- QCSP with container groups
- QCSP with complete bays
- QCSP with bay areas

The problem addressed in the present work is a QCSP with container groups that involve the most significant complexity among the other two classes [5], [7]. Each task, in such problems, is situated at the vessel bay location. Some pairs of tasks situated in the same bay have precedence relations that should be considered in processing these pairs. Thus, these tasks cannot be processed concurrently [1], [8]. In order to eliminate issues of congestion at the yard blocks, simultaneous processing of certain tasks may be disallowed [7], [8]. Each QC is associated with its initial bay position and ready time. During operations, each QC should not be crossed by other QCs. Moreover, two QCs must not work simultaneously at the same bay place, and a safe distance, usually in units of specified bays, must be preserved [1], [7].

2. RELATED WORK

The early works of Daganzo [9] initiated the study of QCSP. The work involved an investigation of problems with multiple vessels at a berth with cranes that are allowed to move freely. Exact approaches were suggested for handling small issues. A branch-and-bound algorithm proposal was made by [10] to solve actual size issues. Nevertheless, these studies never made necessary considerations for the interference of the quay cranes [1]. Another formulation was made by Kim and Park [11], which is based on a mixed-integer programming model. The proposal made by that study is a Branch and Bound algorithm that seeks to reduce the search space with a lower object connected to define a function value that will lead to an optimal schedule. The high computation time required in Branch and Bound with large instances results in certain limitations when applied in real situations; Thus, the researchers modified a greedy randomized adaptive search procedure (GRASP) that allows for the attainment of good quality schedules within considerably shorter computational times. Moccia *et al.* [12] conducted an extended analysis of Kim and Park's [11] model and discovered certain situations whereby there is a possibility of interference between the quay cranes. This work provided a revised mathematical formulation to provide solutions for the weakness demonstrated. At the same time, numerous families with solid inequalities were applied to a branch and cutalgorithmm to provide solutions for large instances. Obtained results showed the positive performance of the suggested branch and cutalgorithmm in comparison with the branch and boundalgorithmm developed by Kim and Park [9]. Kasm and Diabat [13] developed a mixed-integer program and a two-stage exact solution methodology to solve the QCSP. Sun *et al.* [14] address the QCSP with vessel stability constraints, an exact algorithm based on logic-based benders decomposition.

Sammarra *et al.* [15] suggested that QCSP has a routing problem and, therefore, unable to fix the tasks performed by each quay crane. It also suffers a scheduling problem to determine when to start a task and when to finish. And then, a tabu search has been developed for solving the routing issue. The neighbourhood structure utilized a highly potential promising movement to minimize the volume of makespan obtainable within the present schedule. Moreover, a descriptive graph was used to evaluate the schedule obtained during the search process. The tabu search's performance can be compared with the branch and cut that is proposed by [12], [15]. Bierwirth and Meisel [8] disclosed that the mathematical model proposed by [11] is not correct because it cannot be used to find out schedules of interference among quay cranes at all times. This paper developed a revised formulation of the QCSP by creating a suitable time distance between the two tasks in the design. It offered heuristics known as unidirectional scheduling (UDS) that is used on tree search. The UDS achieved the best-known schedules necessary for benchmarking suites utilized in previous studies within limited computation time. Chung and Choy [16] suggested a Genetic Algorithm for the QCSP. Starting from an initial randomly generated population sample, varied mutation and crossover operators were utilized to obtain high-quality schedules. Legato *et al.* [17] offered a new approach for QCSP, which adopted numerous practical issues in the form of quay cranes ready times and due date, and individual cranes processing time. Developments were made in this paper based on the UDS extension that was suggested by [10], which is known as the LTM method. Meisel and Bierwirth [7] offered a platform with the intention of comparing models of optimization and approaches for QCSP by generating a benchmark suite. The suggested generation scheme adopted the comparability principle, reproductively and unbiasedness, with a group of parameters that feature services of container vessels. Izquierdo *et al.* [18]

conducted a study of the suitability of estimation of distribution algorithms (EDA) in the course of solving QCSP. Based on the EDA, there are several methods for solving QCSP. A hybrid estimation of data algorithm was proposed by [4], and it adopted a local search for solving the QCSP. Such a scheme was based on a priori view of the problem to attain high-quality schedules. Nguyen *et al.* [1] creates a new form of priority-based schedule construction for generating quay crane schedules. Such a procedure also brought about the development of two hybrid evolutionary computation approaches that adopted genetic programming and genetic algorithm formulated for QCSPs with container groups that allow for easy location of near-optimal schedules with reasonable time for computations. Yue *et al.* [19] proposed the two-phase scheduling optimization model to optimize the joint scheduling problem between dual-trolley quay crane and AGV.

According to the survey conducted by [20]-[22], most of the previous studies conducted to solve QCSP utilized evolutionary or genetic algorithms because these algorithms are simply designed and easily applied. Yet, these algorithms were reported to obtain solutions with limited quality. Currently, understanding QCSP has become much better, hence simplicity of the design and easy application are no longer basic considerations in selecting the QCSP solver. Under the established understanding, this work proposes a self-adaptation heuristic, which has been proven to have high efficacy when it comes to solving different combinational optimization problems to produce more efficient results [23], [24]. The basic thing this work aims to achieve is to design a self-adaptation heuristic (*self-H*) that adopts the best fitting local search algorithm that can be implemented in the course of search as a remedy to the QCSP problems.

3. PROBLEM DESCRIPTION

QCSP with container group can be described as [7]: A set of tasks $\Omega = \{1,2,\ldots,n\}$ has been assigned to the QCSP together with a set of uniform QCs $Q = \{1,2,\ldots,q\}$. Each of the tasks $i \in \Omega$ is used to represent the loading or unloading activities of a given container group which a QC must undertake without any pre-emption. The tasks have distinct processing times p_i, and they are situated at bay positions l_i inside a vessel of b bays. For the tasks within the same bay, the stacking-dependent accessibility can call for the incorporation of relations between precedence. In the case of a QCSP with container groups, Φ is used to represent precedence-constrained task pairs set, while Ψ is used to represent the set of pair tasks that are permitted for simultaneous processing, for instance, to evade bottleneck at yard blocks used for holding containers for these pair tasks.

Each QC $k \in Q$ has a ready time r^k and an initial bay location l_0^k. Between two neighbouring bays, all QCs can move in the same travel time t. It is presupposed that two QCs cannot operate simultaneously at the same bay. Moreover, they cannot cross each other and should maintain a safety margin s, estimated in units of bays as shown in Figure 1. The issue is to define the feasible time to complete the task on the cranes in terms of the constraints defined. The main goal is to reduce the vessel handling time, which refers to the completion time for the most recently completed task. This problem is referred to as the NP-hard, see [6]. For more details on QCSP, consider [9], [10].

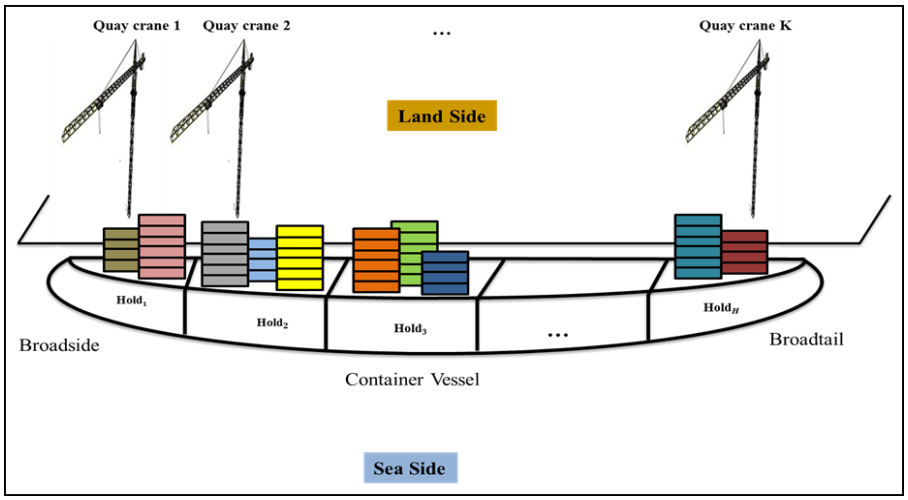

Figure 1. The QCSP illustration

4. THE PROPOSED METHOD

Due to the importance of the QCSP, on the one hand, and the inability of current approaches to operate well all over its instances, on the other hand, the need to advance a new algorithm that has the necessary ability to operate well on all available instances is still pressing matter. The meta-heuristic approaches were effectively utilized to address various optimization problems. Of them are university timetabling [25], multi-objective optimization [26], healthcare [27], data mining [28], multi-robot searching system [29], structural design [30] and others [31]-[34]. Recently, self-adaptation heuristics comprise a group of approaches that are motivated by the aim of automating the design of heuristic methods in order to solve difficult computational search problems and across different problem instances [23], [24], [35]. Accordingly, in this paper, a self-adaptation heuristic (self-H) is proposed. The self-H consists of two separate levels: a higher and a lower level. The Multi-armed bandit selection mechanism (MAB) [36]-[37] selects the most appropriate LS to be executed at the current decision point whose basis is the search status. The lower level is the number of LSs that are managed by the MAB to solves the given problem instance as shown in Figure 2. The *self-H* is executed for a certain number of iterations (*MaxItr*), aiming to improve the quality of the QCSP solution iteratively. The *self-H* will hopefully work effectively during the search. This is due to the ability of the suggested *self-H* to apply different LSs for a different problem in an online manner).

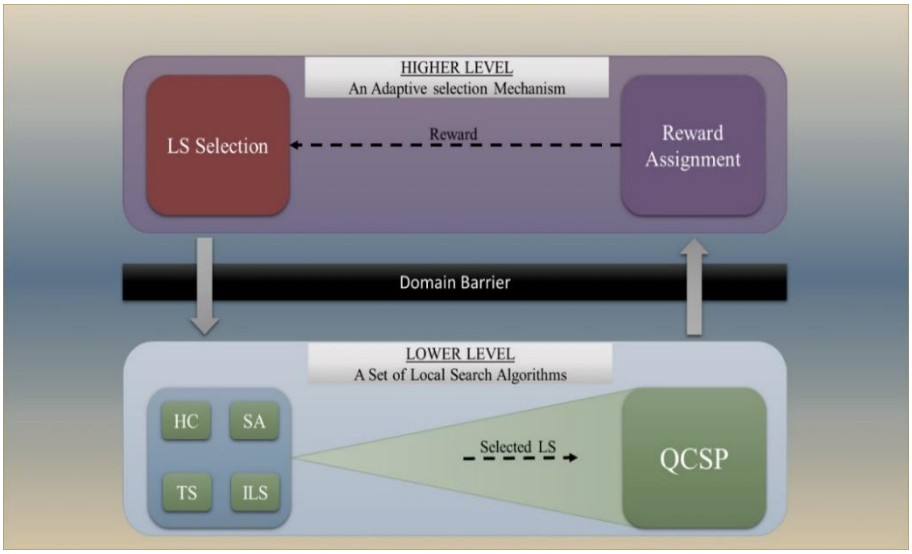

Figure 2. The self-H framework

4.1. The higher level of self-H

The high-level heuristic for the self-H is the adaptive selection mechanism (MAB) which attempts to adaptively select different LSs for different problem instances depend on the status of the search. The adaptive selection mechanism has two components which are the local search impact evaluation and the local search selection mechanism [37].

4.1.1. Local search algorithm impact evaluation component

In this work, each LS is connected with empirical quality estimation ($q'_{ls,i}$) that indicates the average reward gained by an LS from the first iteration up to the current iteration *i* of the search process. The empirical quality estimation is computed using (1) [37], and it is updated throughout the search process.

$$q'_{ls,i+1} = \left(\frac{(n_{ls,i} - 1) \times q'_{ls,i} + r_{ls}}{n_{ls,i}} \right) \quad (1)$$

where $n_{ls,i}$ refers to the number of time the LS_{ls} is used, $q'_{ls,i}$ is the total reward of the LS_{ls} before the i^{th} iteration, and r_{ls} refers to the recent reward of the LS_{ls} if it is used in the present iteration (which is calculated by using (2).

$$r_{ls} = \frac{(f(S) - f(S'))}{|f(S)|} \quad (2)$$

4.1.2. The local search selection mechanism

MAB takes the empirical quality estimation ($q'_{ls,i}$) of each LS and intentionally chooses the one that returns the maximum value by using (3) [37]:

$$Select\ arg\ max_{ls=1..L}\left(q'_{ls} + C \times \sqrt{\frac{2 \times log(\sum_{l=1}^{L} n_l)}{n_{ls}}}\right) \quad (3)$$

L refers to the number of LSs, and *C* refers to the scaling factor that attains exploration-exploitation balance during the search process. In this equation, the right-hand term enhances the exploration ability by giving chances to LSs that recorded low improvements history. The left-hand term of this equation enhances the exploitation ability by favouring the LS, which has the best empirical quality. The *C* factor is very important as it balances between the search exploration and exploitation abilities. Thus, the MAB takes into consideration the improvements achieved by each LS during the search and how many times it is applied. The MAB algorithm process is illustrated in algorithm 1 [37].

Algorithm 1 Multi-Armed Bandit Mechanism

```
Start
While Not Terminated do
if there is a single-based meta-heuristics not utilized yet, then
    ls ← randomly choice the LS that has not been utilized
    else
    ls ← arg arg max ls=1..L ( q'ls + c × sqrt( 2×log(Σ_{j=1}^{L} nj) / nls ) )
    end if
apply LSls and calculate its improvement strength
rls ← CreditAssignment.GetReward(ls)
    nls = nls + 1
    q'ls ← ( ((nls − 1) × q'ls + rls) / nls )
end while
End
```

4.2. The lower level of self-H

In this work, the lower level of self-H is a set of local search algorithms that contains four well-known local search algorithms used to solve the QCSP. The major difference among the utilized local search algorithms is how they escape local optima. These algorithms are Hill climbing algorithm (HC), simulated annealing (SA), tabu search (TS) and iterated local search (ILS) [38]-[41]. In this study, an initial solution is generated based on the priority-based schedule construction algorithm [1]. The criterion for the termination of local search algorithms is adopted in accordance with the number of non-improved iterations.

In order to generate neighbouring solutions, two neighbourhood structures have been adopted. To provide the *self-H* with more opportunities to explore the search space, the neighbour solution is generated through the *Move* neighbourhood structure. In this operator, the last task within a particular crane route is deleted from the current route and then reinserted in the first position of the next route. In case the last route is selected, the first task will be moved to the last position of the previous route. In terms of ILS, perturbation operator is utilized to make a great change in the current solution [40]. Instead of one task, this operator moves the last segment (a group of subsequent tasks) from a route that is selected randomly to the subsequent one. In case the last route is selected, the first segment will be moved to the previous route.

5. EXPERIMENTAL DESIGN

In this study, Meisel and Bierwirth benchmark [7] is employed to assess the performance of the proposed method. This section examines the aspects of this benchmark and self-H parameter settings.

5.1. QCSP benchmark

The adopted Meisel and Bierwirth benchmark contains 400 instances [7] which are generated using QCSPgen generation procedure-which is available at http://prodlog.wiwi.uni-halle.de/qcspgen. The generation process is controlled by the following eight parameters [7]:

a. *s* = Number of container groups.
b. *b* = Number of bays.
c. *c* = Bay's capacity, referring to maximum number of containers per bay.
d. *f* = Handling rate, referring to the percentage of container groups as handled in a service in contrast with the capacity of the vessel.
e. *loc* = Location, in charge of the container groups' distribution over the vessel. One of three supported values are assigned to *loc*:

– *uni*: the unified distribution of container groups over the vessel's bays.
– *cl1*: practically, the common distribution containers are sometimes stored in the immediate vicinity over a vessel. In order to achieve that, a Gaussian distribution is adopted.
– *cl2*: in this case, the container groups are clustered in two areas of the vessel. Two different Gaussian distributions are adopted for this purpose.

f. *d* = Precedence density, in charge of generating the relations of precedence among the groups of containers.
g. *g* = Density of non-simultaneous, in charge of generating the non-simultaneous relations among the groups of containers.
h. *seed= Random seed, in charge of initiating a pseudorandom number generator which leads to start at an arbitrary point in a random sequence.*

These instances are distributed into seven sets according to various crane scheduling issues, such as: number of tasks, number of bays, bays' capacity, handling rate, container groups' distribution, precedence and non-simultaneity relations among container groups and number of cranes. These sets are known as A, B, C, D, E, F and G. The sets A, B, and C refer to small, medium and large vessels with two, four and six cranes, respectively. The rest sets D, E, F and G refer to medium vessels with various container groups' distribution, precedence's density, number of cranes and safety requirements. The features of these sets are illustrated in Table 1.

Table 1. The features of QCSP sets A-G

Set	Vessel size	*b*	*n*	*q*	*f*	*c*	*d*	*loc*	Vessel size
A	Small	10	10-40	2	0.5	200	1.0	*uni*	Small
B	Medium	15	45-70	4	0.5	400	1.0	*uni*	Medium
C	Large	20	75-100	6	0.5	600	1.0	*uni*	Large
D	Medium	15	50	4	0.2,0.8	400	1.0	*cl1, cl2, uni*	Medium
E	Medium	15	50	4	0.5	400	0.8-1	*uni*	Medium
F	Medium	15	50	2-6	0.5	400	1.0	*uni*	Medium
G	Medium	15	50	4	0.5	400	1.0	*uni*	Medium

5.2. Experimental setup

The proposed algorithms (HC, SA, TS, ILS and self-H) are all programmed in Java and executed by using PC with windows7, Intel processor Quad CPU 2.33 GHz, RAM 2.00 GB. In this section, preliminary test or as suggested by previous works, are appropriately used to determine the appropriate values for all proposed algorithm parameters. During the preliminary test, the adopted heuristics are executed 10 runs on seven instances-which are selected based on seven groups founded in the benchmark. The parameter settings used in this work are summarized in Table 2.

Table 2. The parameter settings of the *self-H*

Parameter	Algorithm	Value
MaxItr	*Self-H*	100
NonItr	HC	200
T_max	SA	50
T_min	SA	0.05
β	SA	0.99
NonItr	TS	200
N_neighbors	TS	10
TLS	TS	12
NonItr	ILS	200
C	MAB	0.01

6. EXPERIMENTAL RESULTS AND COMPARISONS

To assess the performance of the suggested self-H for tackling QCSP, two experimental tests are carried out in this work. The first one compared the results of the self-H with that of four well-known LSs; HC, SA, TS and ILS in section 6.1. Fourteen instances are selected in this experimental (two instances from each group):

- A1: Set A, n=10 A2: Set A, n=10
- B1: Set B, n=45 B2: Set B, n=50
- C1: Set C, n=75 C2: Set C, n=80
- D1: Set D, f=0.2, Loc=cl1 D2: Set D, f=0.2, Loc=cl2
- E1: Set E, d=0.8 E2: Set E, d=0.85
- F1: Set F, q=2 F2: Set F, q=3
- G1: Set G, s=0 G2: Set G, s=1

In the second experimental test, a comparison was carried between the results of self-H and those obtained in former studies, shown in section 6.2.

6.1. Comparisons of the self-H with the LSs

This study investigated self-H performance in comparison with that of four well-known LSs; HC, SA, TS and ILS. The obtained results are tabulated in Tables 3 and 4; these results involved Best and Avr. The high-quality solutions are marked in bold. In terms of the best values, the self-H obtained better results than the others. The results tabulated in Table 3 show that the self-H got the best results on thirteen instances out of fourteen, while HC, SA, TS and ILS got the best results on 3, 5, 8 and 9 instances, respectively. Based on the Avr values, Table 4 shows that the self-H outperformed the others as it obtained the best Avr results on eleven instances out of fourteen, while HC and ILS gained the best Avr results on one and two instances.

Table 3. The best results of self-H compared to HC, SA, TS and ILS

Instances	HC	SA	TS	ILS	*self-H*
A1	599	599	599	599	599
A2	561	561	517	561	517
B1	798	863	776	798	768
B2	959	871	786	818	818
C1	1507	1443	1320	1295	1295
C2	1381	1183	1319	1273	1183
D1	373	388	373	373	373
D2	336	339	329	336	329
E1	959	840	840	818	818
E2	959	951	887	818	818
F1	1515	1515	1515	1515	1515
F2	1240	1029	1036	1029	1029
G1	960	820	820	820	820
G2	959	840	818	818	818

Table 4. The average results of *self-H* compared to HC, SA, TS and ILS

Instances	HC	SA	TS	ILS	*self-H*
A1	599	599.00	606.20	599	599
A2	561	561.00	533.03	561	527.45
B1	981.55	932.10	920.13	942.48	854.06
B2	1099.16	1005.48	984.60	903.06	915.55
C1	1991.35	1720.55	1489.93	1488.87	1324.58
C2	2030.26	1617.71	1554.17	1457.32	1370.65
D1	505.06	453.29	424.60	419.87	388.77
D2	363.39	412.94	368.97	375.58	367.35
E1	1099.16	992.16	1009.20	941.84	842.71
E2	1099.16	975.58	960.43	929.94	928.97
F1	1515	1515.00	1515.00	1515.00	1515.00
F2	1333.42	1326.84	1268.33	1226.55	1233.87
G1	1114.00	996.16	962.70	926.94	846.87
G2	1099.16	998.19	971.13	921.29	919.77

For the purpose of assessing the statistical significance of the gained results, a statistical test is conducted as: At the beginning, the Shapiro-Wilk normality test with 0.05 critical level is carried to verify whether the distribution of obtained results is normal or not normal. The test revealed that the obtained results significantly deviate from a normal distribution (the p-value is less than 0.05). Consequently, two

types of non-parametric tests have been utilized; these are the Friedman and Wilcoxon tests. Table 5 provides the ranking of the HC, SA, TS, ILS and self-H in lieu of the Friedman test (where the lower the value, the higher the rank). The self-H ranked first as it had the lowest value. Whilst, ILS, TS, SA and HC ranked 2nd,3rd, 4th and 5th, respectively. It can be noticed that the p-value of the Friedman and Iman-Davenport statistical tests demonstrated the existence of a significant difference between the obtained results (p-value < 0.05, i.e. p-value=0.00). Consequently, a post-hoc statistical test with critical level 0.05 is employed to obtain the regulated p-values for each comparison between the control algorithm (self-H) and the others; HC, SA, TS and ILS. Table 6 shows the adjusted p-values of Holm and Hochberg statistical tests. Table 5 makes it clear that the self-H is far better statistically than HC, SA and TS (adjusted p-value < 0.05), yet it is not statistically significant compared to ILS.

Table 5. The ranking of HC, SA, TS, ILS and *self-H*

Algorithm	Ranking	Index
self-H	1.4643	(1)
ILS	2.3929	(2)
TS	3.0714	(3)
SA	3.75	(4)
HC	4.3214	(5)
Friedman test(p-value)	0.000011	
Iman-Davenport(p-value)	0.000000169853	

Table 6. The adjusted p-value obtained through the application of the post hoc test

self-H vs	unadjusted P	PHolm	PHochberg
HC	0.000002	0.000007	0.000007
SA	0.000131	0.000393	0.000393
TS	0.007161	0.014322	0.014322

Table 7 illustrates the results obtained by the standard self-H for all the instances in the adopted benchmark (sets A to G). In this table, each cell comprises two values, upper and lower. The upper value refers to the average vessel handling times, whereas the lower value refers to the average running time as measured in seconds. The results obtained in this work demonstrate that the suggested self-H is an efficient solution technique for the QCSP. This may result from the ability of the self-H to select different LSs for a different problem in an online manner). By utilizing different LSs during the search, the self-H can deal with various problem instances and cope with any changes that may occur during optimizing a solution. Based on these results, the hypothesis raised above is accepted and proved to be true.

Table 7. the average vessel handling times for instances in sets A-G

Instance set							
A	*n= 10*	*n= 15*	*n= 20*	*n= 25*	*n= 30*	*n= 35*	*n= 40*
	536.8	549.5	516.7	525.2	511.7	521.9	513.9
	< 1	< 1	< 1	< 1	< 1	< 1	< 1
B	*n= 45*	*n= 50*	*n= 55*	*n= 60*	*n= 65*	*n= 70*	
	787	797.5	815.4	819.1	783.6	798.9	
	11.41	4.29	5.32	5.80	3.72	8.93	
C	*n= 75*	*n= 80*	*n= 85*	*n= 90*	*n= 95*	*n= 100*	
	1208.1	1153.6	1141.6	1152.6	1188.7	1134.1	
	5.53	3.00	4.58	10.17	3.41	4.06	
D	*f= 0.2*	*f= 0.2*	*f= 0.2*	*f= 0.8*	*f= 0.8*	*f= 0.8*	
	loc= cl1	*loc= cl2*	*loc= uni*	*loc= cl1*	*loc= cl2*	*loc= uni*	
	339.8	322.6	329.3	1239.9	1264.2	1258.1	
	2.49	< 1	7.22	< 1	1.23	1.56	
E	*d= 0.80*	*d= 0.85*	*d= 0.90*	*d= 0.95*	*d= 1.00*		
	816.8	806.7	815	800.6	821.2		
	2.29	2.06	3.90	0.93	0.25		
F	*q= 2*	*q= 3*	*q= 4*	*q= 5*	*q= 6*		
	1524.1	1037.5	812	676	574.8		
	< 1	< 1	1.13	< 1	< 1		
G	*s= 0*	*s= 1*	*s= 2*	*s= 3*	*s= 4*		
	816.7	806.7	988.1	1152.6	1325.4		
	< 1	< 1	< 1	< 1	< 1		

6.2. Comparison of self-H with the state of the art methods

In this experiment, self-H results are compared with the best-known existing results in the available works of literature (CPLEX and UDS), which are collected from [1]. There are numerous studies that have been proposed in the literature for QCSP. Three of these recent studies achieved the best-known results, so they were selected for comparison with the proposed method. These studies are:

- GA: a genetic algorithm proposed by [5]
- HGA and HGP: two hybrid evolutionary computation methods proposed by [1]

Table 8 reports the GAP the best results obtained by self-H and the other compared algorithms. The aforementioned tabulated results reveal that self-H attained competitive results as compared to others. This comparison demonstrates that the self-H excelled USD, HGA and HGP on D, F and G datasets. Even though self-H did not match the best-known existing results for all datasets, the obtained results for these datasets remain very competitive.

Table 8. The GAP the best results obtained by self-H and the other compared algorithms

	CPLEX	UDS	GA	HGA	HGP
A	3.35	3.20	2.88	3.20	3.21
B	6.20	3.91	-	3.55	3.66
C	15.96	5.95	-	5.65	5.76
D	-	-1.45	-	-1.61	-1.56
E	-	5.50	-	-	5.33
F	-	-2.90	5.25	-2.99	-2.95
G	-	-1.56	-	-1.62	-1.62

"−" indicates that the compared method did not report the best result
Instance set

7. CONCLUSION

The researchers in this study proposed a Self adaptation heuristic (self-H) to solve QCSP. The self-H involves two separate levels, the upper level (LS-controller) and the lower level (QCSP-solver). This enables the self-H to deal with different problems and the changes that may happen throughout the course of optimizing a solution. The LS-controller is based on an adaptive selection mechanism that adaptively selects which LS will be adopted by the QCSP-solver to solve the given problem. The QCSP-solver embeds a group of adopted local search algorithms. The effectiveness of the self-H was verified by using the standard QCSP benchmark, suggested by Meisel and Bierwirth (2011). The experimental results showed that the f-H is an efficient solution method for the QCSP as it is able to inherit the most important features of the QCSP-solver to select the most suitable local search algorithm to be applied through the search process. In future work, the proposed algorithm (self-H) can be applied to solve more practical problems such as university timetabling and multi-objective optimization of machine learning model. This application will identify the generality characteristic of the proposed algorithm.

ACKNOWLEDGEMENTS

This research was granted by the Universiti Kebangsaan Malaysia (UKM) GUP-2020-091.

REFERENCES

[1] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Hybrid evolutionary computation methods for quay crane scheduling problems," *Computers and Operations Research*, vol. 40, no. 8, pp. 2083-2093, 2013, doi: 10.1016/j.cor.2013.03.007.

[2] M. E. Petering and K. G. Murty, "Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal," *Computers and Operations Research*, vol. 36, no. 5, pp. 1711-1725, 2009, doi: 10.1016/j.cor.2008.04.007.

[3] Q. Zeng and Z. Yang, "Integrating simulation and optimization to schedule loading operations in container terminals," *Computers and Operations Research*, vol. 36, no. 6, pp. 1935-1944, 2009, doi: 10.1016/j.cor.2008.06.010.

[4] C. ExpóSito-Izquierdo, J. L. González-Velarde, B. Melián-Batista, and J. M. Moreno-Vega, "Hybrid estimation of distribution algorithm for the quay crane scheduling problem," *Applied Soft Computing*, vol. 13, no. 10, pp. 4063-4076, 2013, doi: 10.1016/j.asoc.2013.05.006.

[5] N. Kaveshgar, N. Huynh, and S. K. Rahimian, "An efficient genetic algorithm for solving the quay crane scheduling problem," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13108-13117, 2012, doi: 10.1016/j.eswa.2012.05.091.

[6] Z. Lu, X. Han, L. Xi, and A. L. Erera, "A heuristic for the quay crane scheduling problem based on contiguous bay crane operations," *Computers and Operations Research*, vol. 39, no. 12, pp. 2915-2928, 2012, doi: 10.1016/j.cor.2012.02.013.
[7] F. Meisel and C. Bierwirth, "A unified approach for the evaluation of quay crane scheduling models and algorithms," *Computers and Operations Research*, vol. 38, no. 3, pp. 683-693, 2011, doi: 10.1016/j.cor.2010.08.001.
[8] C. Bierwirth and F. Meisel, "A fast heuristic for quay crane scheduling with interference constraints," *Journal of Scheduling*, vol. 12, no. 4, pp. 345-360, 2009, doi: 10.1007/s10951-009-0105-0.
[9] C. F. Daganzo, "The crane scheduling problem," *Transportation Research Part B: Methodological*, vol. 23, no. 3, pp. 159-175, 1989, doi: 10.1016/0191-2615(89)90001-5.
[10] R. I. Peterkofsky and C. F. Daganzo, "A branch and bound solution method for the crane scheduling problem," *Transportation Research Part B: Methodological*, vol. 24, no. 3, pp. 159-172, 1990, doi: 10.1016/0191-2615(90)90014-P.
[11] K. H. Kim and Y.-M. Park, "A crane scheduling method for port container terminals," *European Journal of operational research*, vol. 156, no. 3, pp. 752-768, 2004, doi: 10.1016/S0377-2217(03)00133-4.
[12] L. Moccia, J. F. Cordeau, M. Gaudioso, and G. Laporte, "A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal," *Naval Research Logistics (NRL)*, vol. 53, no. 1, pp. 45-59, 2006, doi: 10.1002/nav.20121.
[13] O. Abou Kasm, and A. Diabat, "Next-generation quay crane scheduling," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 694–715, 2020, doi: 10.1016/j.trc.2020.02.015.
[14] D. Sun, L. Tang, R. Baldacci, and A. Lim, "An exact algorithm for the unidirectional quay crane scheduling problem with vessel stability," *European Journal of Operational Research*, vol. 291, no. 1, pp. 271–283, 2020, doi: 10.1016/j.ejor.2020.09.033.
[15] M. Sammarra, J.-F. Cordeau, G. Laporte, and M. F. Monaco, "A tabu search heuristic for the quay crane scheduling problem," *Journal of Scheduling*, vol. 10, no. 4, pp. 327-336, 2007, doi: 10.1007/s10951-007-0029-5.
[16] S. H. Chung and K. L. Choy, "A modified genetic algorithm for quay crane scheduling operations," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4213-4221, 2012, doi: 10.1016/j.eswa.2011.09.113.
[17] P. Legato, R. Trunfio, and F. Meisel, "Modeling and solving rich quay crane scheduling problems," *Computers AND Operations Research*, vol. 39, pp. 2063-2078, 2012, doi: 10.1016/j.cor.2011.09.025.
[18] C. E. Izquierdo, J. L. G. Velarde, B. Melián-Batista, and J. M. Moreno-Vega, "Estimation of distribution algorithm for the quay crane scheduling problem," in *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, 2011, pp. 183-194, doi: 10.1007/978-3-642-24094-2_13.
[19] L. Yue, H. Fan, and C. Zhai, "Joint configuration and scheduling optimization of the dual trolley quay crane and AGV for automated container terminal," *Journal of Physics: Conference Serie*, vol. 1486, no. 7, 2020.
[20] C. Bierwirth and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, vol. 202, no. 3, pp. 615-627, 2010, doi: 10.1016/j.ejor.2009.05.031.
[21] C. Bierwirth and F. Meisel, "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, vol. 244, no. 3, pp. 675-689, 2015, doi: 10.1016/j.ejor.2014.12.030.
[22] D. Kizilay, and D. T. Eliiyi, "A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals," *Flex Serv Manuf J*, vol. 33, pp. 1–42, 2021, doi: 10.1007/s10696-020-09385-5.
[23] G. Kendall and J. Li, "Competitive travelling salesmen problem: A hyper-heuristic approach," *Journal of the Operational Research Society*, vol. 64, no. 2, pp. 208-216, 2013, doi: 10.1057/jors.2012.37.
[24] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 942-958, 2010, doi: 10.1109/TEVC.2010.2041061.
[25] S. Abdullah, and M. Alzaqebah, "A hybrid self-adaptive bees algorithm based disruptive selection for examination timetabling problems," *Applied Soft Computing*, vol. 13, no. 8, pp. 3608-3620, 2013, doi: 10.1016/j.asoc.2013.04.010.
[26] S. Sabzevari, and S. Abdullah, "Gene selection in microarray data for multi-objective perspective," *3rd Conference on Data Mining and Optimisation*, 2011, pp. 199-207, doi: 10.1109/DMO.2011.5976528.
[27] L. Z. Long, A. R. Hamdan, and A. A. Bakar, "Anomaly based on frequent-outlier for outbreak detection in public health surveillance," *International Journal of Medical and Health Sciences*, vol. 7, no. 3, pp. 151-158, 2013, doi: 10.5281/zenodo.1327684.
[28] M. Mousavi, A. A. Bakar, and M. Vakilian, "Data stream clustering algorithms: A review," *Int J Adv Soft Comput Appl*, vol. 7, no. 3, pp. 1-15, 2015.
[29] B. Nakisa, M. N. Rastgoo, M. Z. Ahmad Nazri, and M. Nordin, "Target searching in unknown environment of multi-robot system using a hybrid particle swarm optimization," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 13, pp. 4055-4065, 2018.
[30] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Metaheuristic applications in structures and infrastructures," Newnes, 2013.
[31] M. A. Dulebenets, "An Adaptive Island Evolutionary Algorithm for the berth scheduling problem," *Memetic Computing*, vol. 12, no. 1, pp. 51–72, 2019, doi: 10.1007/s12293-019-00292-3.

[32] G. D'Angelo, R. Pilla, C. Tascini, and S. Rampone, "A proposal for distinguishing between bacterial and viral meningitis using genetic programming and decision trees," *Soft Computing*, vol. 23, no. 22, pp. 11775–11791, 2019, doi: 10.1007/s00500-018-03729-y.
[33] H. Zhao, and C. Zhang, "An online-learning-based evolutionary many-objective algorithm," *Information Sciences*, vol. 509, pp. 1–21, doi: 10.1016/j.ins.2019.08.069.
[34] N. Panda, and S. K. Majhi, "How effective is the salp swarm algorithm in data classification," In *Computational Intelligence in Pattern Recognition*, pp. 579-588, 2020, doi: 10.1007/978-981-13-9042-5_49.
[35] N. R. Sabar and G. Kendall, "Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems," *Information Sciences*, vol. 314, pp. 225-239, 2015, doi: 10.1016/j.ins.2014.10.045.
[36] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, pp. 4-22, 1985.
[37] Á. Fialho, "Adaptive operator selection for optimization," Université Paris Sud-Paris XI, 2010.
[38] L. Di Gaspero, A. Schaerf, M. Cadoli, W. Slany, and M. Falaschi, "Local Search Techniques for Scheduling Problems: Algorithms and Software Tool," Tesi Online, 2003.
[39] A. Bassel, H. M. Haglan, and A. Sh. Mahmoud, "Local search algorithms based on benchmark test functions problem," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 3, pp. 529-534, doi: 10.11591/ijai.v9.i3.pp529-534.
[40] E.-G. Talbi, "Metaheuristics: from design to implementation," John Wiley & Sons, 2009.
[41] S. Alani, A. Baseel, M. M. Hamdi, and S. A. Rashid, "A hybrid technique for single-source shortest path-based on A* algorithm and ant colony optimization," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, pp. 356-363, 2020, doi: 10.11591/ijai.v9.i2.pp356-363.