An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports

R. Tavakkoli-Moghaddam a,*, A. Makui b, S. Salahi c, M. Bazzazi d, F. Taheri d

aDepartment of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
bDepartment of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
cDepartment of Industrial Engineering, Islamic Azad University, Science and Research Branch, Iran
dDepartment of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran

1. Introduction

The use of containers in maritime transportation is rapidly increasing during last decades. A container terminal plays an essential role in a port as an intermodal interface, in which container vessels dock on berths, load outbound (export), and unload inbound (import) containers. The main productivity of a container terminal can be measured in terms of two factors: (1) Ship operations where containers are discharged from and onto a ship; and (2) receiving and delivery operations where containers are transferred to and from outside trucks (Kim & Park, 2004). The planning process of ship operations consists of berth planning, quay crane (QC) or work scheduling, and discharge and load sequencing. During the process of berth planning, the berthing time and the berthing position of a container ship on a wharf must be determined. A QC schedule specifies the service sequence of bays in a ship by each QC and the time schedule for the services. Input data for QC scheduling consists of a stowage plan of a ship, the ready time of each QC, and a yard map showing the storage locations of a containers bound for the ship. Finally, during discharge and load sequencing, the discharge and load sequence of individual containers are determined based on a QC schedule. This paper addresses the QCSAP, which is pertinent to the second stage of ship operation planning.

By considering that the main objective of a QC scheduling problem is to minimize the makespan of vessels. Kim and Park (2004) presented a mathematical model for a QC scheduling problem. We extend their model incorporating with QC assignment problem, namely QCSAP. We apply this extended model for a container terminal located at the south of Iran, called Shahid Rajaei Terminal. In the case of any delay in the total known charge or discharge time, the container terminals must pay the related cost, called Demoraje, to vessels. Furthermore, they receive the cost of earliness in the total completion time, called Dispatch. Thus, makespan of container vessels is the latest completion time among all handling tasks of the container vessel, which is a critical success factor. On the other hand, the use of each quay crane has a fixed and variable cost for each unit of time. In this paper, we propose and develop a new mixed integer programming (MIP) model for the quay crane scheduling and assignment problem to vessels optimizing the efficiency of container terminal operations.

There are many different decision problems involved in a container terminal operation, such as berth allocation, storage space allocation, quay crane (QC) scheduling, QC allocation, location assignment, yard crane scheduling, trailer routing problem, and so on. All these decisions affect on each other. The QC scheduling problem is one of the significant issues in container terminal operations. The main goal of this problem is to determine the sequence of loading and unloading operations in such a way that the completion time of a ship operation is minimized, similar to the parallel machine scheduling problem. However, the QC scheduling

* Corresponding author. Tel.: +98 21 88021067; fax: +98 21 88013102.
E-mail address: tavakoli@ut.ac.ir (R. Tavakkoli-Moghaddam).
problem has several unique characteristics that are different from typical parallel machine problems. For example, when loading and unloading operations are performed at the same ship-bay, the discharging operation must precede the loading operation. When unloading operations are performed in a ship-bay, tasks on a deck may not be performed simultaneously. It causes interference among yard cranes that transfer containers corresponding to two tasks.

The literature can be divided over the various problems in the container terminal as follows: (1) arrival of the ship; (2) unloading and loading of the ship; (3) transport of containers from ship to stack; and (4) stacking of containers. The most important problem in the first problem is the berth allocation problem (BAP). Imai, Nagaiwa, and Tat (1997) introduced a model minimizing the sum of port staying times of ships as well as minimizing dissatisfaction of ships in terms of the berthing order. Imai, Nishimura, and Papadimitriou (2001) introduced both static and dynamic BAPs. They developed a heuristic based on Lagrangian relaxation. Nishimura, Imai, and Papadimitriou (2001) extended the dynamic BAP proposed by Imai et al. (2001) to treat some physical restrictions on berthing ship such as water depth and berth length. They employed a genetic algorithm (GA) to find the approximate solution for the given problem.

Imai, Nishimura, and Papadimitriou (2003) introduced the service priority of the ship in dynamic berth allocation circumstances. They first attempted to utilize the Lagrangian relaxation; however, they found that the relaxed problem is reduced to the quadratic assignment problem (QAP), which is difficult to solve in the polynomial bounded computation time. Consequently, they applied a genetic algorithm for this problem. Imai, Nishimura, Hattori, and Papadimitriou (2007) solved the berth allocation problem at indented berths for mega-containerships by a GA.

In the above-mentioned problem of the second problem, more useful and important subjects are Stowage planning for container and the crane scheduling problem. We present a model for crane scheduling problem in this paper.

Kim and Park (2004) presented a mathematical model for a QC scheduling problem. This study proposed a branch and bound (B&B) method to obtain the optimal solution of the QC scheduling problem and a heuristic search algorithm, called greedy randomized adaptive search procedure (GRASP), to overcome the computational difficulty of the B&B method. Daganzo (1989) first discussed that the limitation in the length of a berth must be considered simultaneously during the crane scheduling. However, more emphasis was placed on schedules of QCs than that of the berth which is the main issue of this study. Regarding the crane-scheduling problem, he suggested an algorithm to determine the number of cranes in order to assign to ship bays of multiple vessels.

Li, Cai, and Lee (1998) considered the berth-scheduling problem to be a scheduling problem for a single processor (i.e., berth) that can simultaneously perform multiple jobs (vessels). They suggested various algorithms based on first-fit-decreasing (FFD) heuristics and tested the algorithms by a simulation study. Bish (2003) considered a container terminal loading/unloading containerers to and from a set of ships, and storing the containers in the terminal yard. The problem is (i) to determine a storage location for each unloaded container, (ii) to dispatch vehicles to containers, and (iii) to schedule the loading and unloading operations on the cranes in such a way that the maximum time to serve a given set of ships is minimized. At last, he proposed a heuristic algorithm based on formulating the problem, referred to a transshipment problem.

Cordeau, Gaudioso, Laporte, and Moccia (2005) considered the quay crane scheduling problem to minimize the vessel completion time and the crane idle times. They proposed a branch-and-cut algorithm to solve this problem. Lee, Wang, and Miao (2008) studied QC scheduling with non-interference constraints and they proposed a genetic algorithm to obtain near-optimal solutions. Jung and Kim (2006) proposed genetic algorithm and simulated annealing methods to schedule loading operations when multiple yard cranes are operating in the same block. The loading scheduling methods considered interferences between adjacent yard cranes. It attempts to minimize the makespan of the yard crane operation. They considered the container handling time, the yard crane travel time, and the waiting time of each yard crane, when evaluating the makespan of the loading operation by yard cranes. Kim and Kim (1999) investigated a routing problem of straddle carriers in port container terminals. They proposed a beam search algorithm to minimize the total travel distance of these carriers in the yard.

Kia, Shayan, and Ghobt (2002) used a simulation model to investigate and evaluate the performance of a container terminal considering its handling equipment and terminal capacity. Kim, Kim, Hwang, and Koc (2004a) studied an operator-scheduling problem of a real-case container terminal in Korea. They used commercial software to present an efficient scheduling of operators of handling devices. Kim, Won, Lim, and Takahashi (2004b) proposed a control framework for automated container terminals. They presented an architectural design of control software and a simulation-based test-bed for testing various control rules of the control software, in which automation of handling equipment and operation in port container terminals was utilized. Kim and Bae (1998) proposed a methodology to convert a current bay layout into a desirable layout by moving the fewest possible number of containers and in the shortest possible travel distance. They presented three mathematical models for the bay matching, the move planning, and the task sequencing problems minimizing the completion time of the re-marshaling operation.

The rest of this paper is organized as follows: the extended quay crane problem (QCP) is formulated in Section 2 and the proposed GA is developed in Section 3. Computational results are reported in Section 4 and finally Section 5 covers the conclusion.

### 2. Mathematical model

Vessels can be processed simultaneously by several QCs. The processing time of a vessel will be longer than scheduled if an insufficient number of QCs is assigned to the vessel. This will delay in the departure of the vessel. However, we cannot assign too many QCs to vessels because of the number of QCs constraint. So, in this paper, we consider the optimal allocation and schedule of a given number of QCs to vessels planned to arrive in the planning horizon. Zhang, Liu, Wan, Murty, and Linn (2003) considered a fixed planning horizon with the rolling-horizon approach (RHA) for a storage space allocation problem (SSAP), as shown in Fig. 1.

In most real cases, there are dynamic situations in ports where vessels always move in and out. We cannot assume that the number of active vessels are fixed because many vessels after unload/load processes leave the port in a given time horizon and other vessels are replaced with them. Therefore, we cannot present a fixed schedule for a planning horizon. This plan assumes that all vessels arrive and leave the port at the beginning and end of the planning horizon, respectively. However, the above situation never happens in real cases. In this case, a
new planning horizon starts while arriving or departing a vessel to/from the port, as shown in Fig. 2. In this figure, planning horizon 2 starts after 1.5 days of planning horizon 1 with arriving a new vessel in the terminal. However, planning horizon 3 starts after two days of planning horizon 2 with arriving/departing another vessel. In other words, the starting of this planning horizon happens when a vessel arrive or leave the port. So, after each rescheduling, a QC assigned to vessel A, may be assigned to vessel B in the new scheduling (after completing all the tasks in vessel A).

Following is an extension of the QCSAP formulated as a mixed-integer programming model. We assign QCs to vessels and determine the sequence of loading/unloading operations performed for each planning horizon.

2.1. Assumptions

1- The docking positions and the workloads of vessels are given as input to the QCSAP.
2- QCs are identical, both in terms of productivity in loading/unloading containers and in terms of moving speed from a bay to other bay.
3- The safety distance between each pair of adjacent QCs depends on the width of a bay. Then, only one QC can work on a bay at a time.
4- Each vessel is divided longitudinally into bays; each bay accommodates a row of container stacks; bays in all vessels are of the same length. Thus, the lengths of vessels are in (whole) number of bays (bay lengths).

2.2. Input parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>number of quay cranes ($K = 1, 2, \ldots, K$)</td>
</tr>
<tr>
<td>$M$</td>
<td>number of vessels ($m = 1, 2 \ldots M$)</td>
</tr>
<tr>
<td>$N_m$</td>
<td>number of tasks on the vessel $m$ ($i, j = 1, 2 \ldots N_m$)</td>
</tr>
<tr>
<td>$r_k$</td>
<td>time required to perform task $i$ on vessel $m$</td>
</tr>
<tr>
<td>$t_{kj}^m$</td>
<td>travel time of the $k$-th QC from location ($l_i^m$) of task $i$ to location ($l_j^m$) of task $j$ on vessel $m$. Additionally, $t_{ij}^m$ represents the travel time from the initial position ($l_k^m$) of the $k$-th QC to location ($l_j^m$) of task $j$ on vessel $m$. In addition, $t_{ij}^m$ represents the travel time from location ($l_i^m$) of task $j$ to the final destination ($l_k^m$) of $k$-th QC on vessel $m$.</td>
</tr>
<tr>
<td>$H_k$</td>
<td>variable cost of using the $k$-th QC</td>
</tr>
<tr>
<td>$S_k$</td>
<td>fixed cost of using the $k$-th QC</td>
</tr>
<tr>
<td>$W_m$</td>
<td>tardiness cost of vessel $m$</td>
</tr>
<tr>
<td>$V_m$</td>
<td>earliness income of vessel $m$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>set of pairs of tasks that cannot be performed simultaneously. When tasks $i$ and $j$ cannot be performed simultaneously ($i, j \in \psi$).</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>set of ordered pairs of tasks between which there is a precedence performed relationship when task $i$ must precede task $j$ ($i, j \in \Phi$).</td>
</tr>
<tr>
<td>$M$</td>
<td>arbitrary positive number</td>
</tr>
</tbody>
</table>

Fig. 2. Rolling of the fixed planning horizon.

Fig. 2. Rolling of the dynamic planning horizon.
2.4. Mathematical model

\[
\begin{align*}
\text{min } Z &= \sum_{k=1}^{K} \left( H_k Q_{k2} + S_{k1} \sum_{m=1}^{M} Y_{km} \right) + \sum_{m=1}^{M} W_m T_m - \sum_{m=1}^{M} V_m E_m \\
\text{s.t. } \quad & E_m - T_m = \lambda^* \sum_{n=1}^{N_m} P_{ij}^m - C_m \quad \forall m \\
& \sum_{j=1}^{N_n} X_{ij}^{km} = Y_{km} \quad \forall m, k \\
& \sum_{j=i}^{N_n} X_{ij}^{kb} = Y_{km} \quad \forall m, k \\
& \sum_{j=0}^{N_m} \sum_{j=1}^{N_m} X_{ij}^{km} = 1 \quad \forall j, j \neq i; \forall m \\
& \sum_{n=0}^{N_n} X_{ij}^{km} - MY_{km} \leq 0 \quad \forall m, k \\
& \sum_{j=1}^{N_m} \sum_{j=1}^{N_m} X_{ij}^{km} = 0 \quad \forall m, k \\
& D_{ij}^m + T_{ij}^m + P_{ij}^m - D_{ij}^m \leq M(1 - X_{ij}^{km}) \quad \forall i, j; j \neq i; \forall m, k \\
& D_{ij}^m + P_{ij}^m \leq D_{ij}^m \quad \forall (i, j) \in \Phi_m, \forall m \\
& Z_{ij}^m \leq P_{ij}^m \leq M(1 - Z_{ij}^m) \quad \forall i, j; j \neq i; \forall m \\
& \sum_{n=0}^{N_n} X_{ij}^{km} - \sum_{n=0}^{N_n} X_{ij}^{km} \leq (M Z_{ij}^m + Z_{ij}^m) \quad \forall i, j; j \neq i; l < j; \forall m, k \\
& D_{ij}^m + T_{ij}^m + P_{ij}^m - D_{ij}^m \leq M(1 - X_{ij}^{km}) \quad \forall j, i \neq j; \forall m, k \\
& \sum_{k=1}^{K} \sum_{i=1}^{N_m} Y_{km} - M(1 - Y_{km}) \leq 0 \quad \forall m, k; k \leq K - 1 \\
& Q_k \leq M(1 - Y_{km}) \quad \forall m, k \\
& X_{ij}^{km}, Z_{ij}^m, Y_{km} \in \{0, 1\} \quad \forall i, j, m, k \\
& Q_k, C_m, D_j \geq 0 \quad \forall j, m, k
\end{align*}
\]

We have corrected Eq. (2).

The objective function (1) minimizes the total cost of the proposed model. Three terms of Eq. (1) measure the revenue of the earliness, the cost of tardiness as well as the fixed and variable costs of the QC assigned to vessels. Constraint (2) computes the total tardiness and earliness. The first term of the right hand side of this constraint is related to our case study representing an approximate due date based on an agreement between the port and vessel. We sum the total operation time according to the number of jobs assigned by the vessel. \( \lambda \) is the agreement coefficient, which is greater than 1. Constraints (3) and (4) select the first and last tasks for each QC on each vessel, respectively. Constraint (5) ensures that each task on each vessel should be completed by exactly one QC. Constraint (6) states if a QC is not assigned to a vessel, the vessel tasks will not be performed by this QC. Constraint (7) shows a flow balance ensuring that tasks are performed in well-defined sequences. Constraint (8) determines the completion time for each task and eliminates sub-tours simultaneously. Constraint (9) denotes that task \( i \) should be completed before task \( j \). Constraint (10) defines \( Z_{ij}^m \) such that \( Z_{ij}^m = 1 \) when the operation for task \( j \) on vessel \( m \) starts after the operation for task \( i \) completed; 0 otherwise. Constraint (11) guarantees that tasks \( i \) and \( j \) cannot be performed simultaneously when \((i, j) \in \Psi\). By Constraint (12), interference among QCs can be avoided. Suppose that tasks \( i \) and \( j \) on vessel \( m \) are performed simultaneously and \( f_{ij}^m < f_{ij}^m \). This means that \( Z_{ij}^m + Z_{ij}^m = 0 \). It is worth noting that both QCs and tasks are ordered in an increasing one of their relative locations in terms of an increasing ship-bay number. Suppose that, for \( k_1 < k_2 \), QC \( k_1 \) performs tasks \( j \) and QC \( k_2 \) performs task \( i \). Then, interference between QCs \( k_1 \) and \( k_2 \) results. However, in the case of \( \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{n=1}^{N_n} (X_{ik}^{km} - X_{jk}^{km}) = 0 \), it cannot be allowed because of constraint (12), and then we have \( Z_{ij}^m + Z_{ij}^m = 0 \). The completion time of each QC is defined by Constraint (13). Constraint (14) restricts the earliest starting time of operations by each QC. Constraint (15) shows that there is no interference in assigning QCs to vessels and Constraint (16) determines the completion time of vessels.

3. The proposed genetic algorithm

Genetic algorithms (GAs) have been used extensively in combinatorial optimization problems, such as sequencing and scheduling problems (Holland, 1975; Goldberg, 1989). GA is a well-known meta-heuristic approach inspired by the natural evolution of the living organisms that works on a population of the solutions simultaneously. It combines the concept of survival of the fittest with structured, yet randomized, information exchange to form robust exploration and exploitation of the solution space. A fitness value derived from the objective value of the schedule is assigned to each single individual, called a chromosome. The exploration process is performed by a genetic operator, namely crossover. The exploitation process is also performed by another genetic operator, namely mutation. The trade-off between these two processes is controlled by the parent selection and offspring acceptance strategies. A single iteration is called a generation. The individuals of the new generation are obtained from the individuals of the previous one by applying reproduction and mutation procedures. Only the more fit individuals are selected.

3.1. Chromosome representation and initial solution procedure

The initial and most important step of the GA implementation is the solution representation or chromosome design. The chromosome representation used in this paper demonstrates each job in the schedule as a gene in the chromosome. In the proposed GA, a chromosome consist of a rectangular matrix with \((V \times (A + 1))\) genes, where \(V\) and \(A\) represent number of vessels and the greatest
sider following steps:

1. Denote $N_i$ for the number of jobs on vessel $i$ and genes with amount zero “0” considering virtual jobs.
2. Set non-zero digits in the first row of the chromosome as the number of assigned quay cranes to vessels, in which the sum of these digits must be less than or equal to the total number of quay cranes. The entire set of jobs on each vessel can be encoded by a vertical single string. A GA chromosome represents a sequence of jobs as depicted in Fig. 3. This figure also shows the structure of a quay crane assignment, in which there are three vessels and six quay cranes.

It is a well-known fact that the structure of the initial population plays an essential role in determining the efficiency of GAs (Goldberg, 1989). However, most GA implementations in the literature employ randomly generated populations for initiation. To generate the random solutions for the initial population, we consider the following steps:

Step 1. Based on the sequence of jobs on each vessel represented by the chromosome, a quay crane schedule can be constructed using the following steps that are the extension of the procedure proposed by Lee et al. (2008) used for each vessel separately. However, in this paper, we know the initial position of each quay crane at the beginning of scheduling.

Step 2. Compare the completion time of the two available quay cranes to finish their assigned jobs, and assign this job to the quay crane with earlier completion time. Then the position and the completion time of the assigned quay crane and job are updated. If there are two quay cranes available, go to Step 2.

Step 3. Compare the distance between this job and these two available quay cranes, and assign this hold to the quay crane with the shorter distance. Then the position and the completion time of the assigned quay crane and job are updated. If their distance is equal, go to Step 4.

Step 4. Assign this job to the quay crane with the smaller number. Then the position and the completion time of the assigned quay crane and job are updated.

Step 5. Steps 1–4 are repeated until all the jobs in the chromosome are assigned.

By assigning the jobs to quay cranes, the precedence and simultaneity constraints must be satisfied as follow:

1) Suppose job $i$ precedes job $j$, following steps must be considered for assigning job $i$ to QCs.

If job $i$ is not assigned to any QCs, swap the position of jobs $i$ and $j$ in the chromosome (i.e., assign job $i$ instead of job $j$).

If job $i$ is assigned to one of the QCs, the start time of job $j$ is determined by the completion time of job $i$, according to Constraint 9.

1) Suppose job $i$ and $j$ cannot proceed simultaneously, for assigning job $j(i)$ to QCs operates as follows:

If job $i(j)$ is not assigned to any QCs, assign job $j(i)$ according to the current algorithm.

If job $i(j)$ is assigned to one of the QCs, the start time of job $j(i)$ is determined by the completion time of job $i(j)$, according to Constraint 10.

The QCs assignment to jobs on each vessel for the above-mentioned example that obtained by the above procedure is shown in Fig. 4. As a result, the fitness function of the chromosomes is defined as given in Eq. (1).

### 3.2. Genetic operators design

In this paper, we use two crossover operators for two parts of the chromosome separately. The first row shows the QCs assignment to vessels and other genes show the sequence of jobs in each vessel. For the first and second parts, we use arithmetic crossover and extended patching crossover, respectively.

#### 3.2.1. Arithmetic crossover

It is important to maintain the feasibility of the newly generated offspring for the problem at hand. Thus, we use the arithmetic crossover (AC) operator to explore the solution space and maintaining the feasibility of the newly generated offspring simulta-

<table>
<thead>
<tr>
<th>Sequence of jobs on each QC</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3$</th>
<th>$Q_4$</th>
<th>$Q_5$</th>
<th>$Q_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_1$</td>
<td>2, 1, 7, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$J_2$</td>
<td></td>
<td>6, 3, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$J_3$</td>
<td></td>
<td></td>
<td>4, 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$J_4$</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$J_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5, 6, 3</td>
<td></td>
</tr>
<tr>
<td>$J_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3, 1, 2, 4, 5</td>
</tr>
</tbody>
</table>

Fig. 4. Structure a quay crane assignment and schedule from a chromosome.
neously. The AC produces a new offspring as a complimentary linear combination of the parents as follows:
\[
\text{Offspring} = \lambda \times \text{Parent 1} + (1 - \lambda) \times \text{Parent 2},
\]
where, \(\lambda\) is a randomly generated number within interval \((0.5,1)\). Thus, each gene value (i.e., allele) in the newly generated offspring is obtained. To increase the affect of the parent with better fitness, we set the parent with better fitness to Parent 1. The AC guarantees that the generated offspring will remain feasible if its parents are feasible. Since, values in genes are integer values, the rounded of gene value (Offspring), must be considered as a true value. Thus, the total QC’s Constraint may be not satisfied and we face an error as \(e = PV\) [gene value(offspring)] - K. To overcome this problem, if \(e > 0\) then heuristically subtract \(e\) value from the gene with maximum value. The performance of the AC with \(\lambda = 0.7\) and the total number of QCs 7 is shown in Fig. 5.

3.2.2. Extended patching crossover

Patching crossover operator is based on the crossover operator used in Cheng and Gen (1997), which they call uniform order-based crossover. This crossover operator generates a template binary string where the number of “1”s and “0”s are controlled. The template binary string is mapped on one of the parents, in which those genes that are positioned in the same locations with the “1”s in the template binary string are directly transported to the child chromosome. The remaining idle gene locations in the child, which is correspond to the locations containing zeros in the template binary string, are filled with the genes in the second parent. In this paper, we proposed an extended patching crossover for part of the chromosome that represents sequence of jobs in vessels, in which we produce a binary matrix \((V \times A)\) and use the operator for each column (vessel) separately. Fig. 6 depicts a typical illustration of this crossover operator. The main steps of the proposed algorithmic structure of this crossover operator are given below.

1. Set the number “1”s in the binary matrix to be generated to “p” and \(p\) is equal to \([|V \times A|/2]\).
2. Randomly generate a binary string with \(p\) “1”s as defined in Step 1.
3. Randomly choose two parents (e.g., Parent 1 and Parent 2) from the population.
4. Copy the genes from Parent 1 corresponding to the locations of the “1”s in the binary matrix to the same positions in the child except the genes with amount of the zero.
5. Cross out the genes from Parent 2 copied from Parent 1 so that the repetition of a gene in the new offspring is avoided. Do not cross out the asterisks because of the optimal assignment number of QCs to each vessel. In this means, the assigned maximum number of QCs to each vessel in offspring do not affect from parents. Thus, the genes with amount of zero are virtual jobs so do not cross them too.
6. Fill out the remaining idle gene locations with the uncrossed genes that remain in Parent 2 by preserving their gene sequence in Parent 2.

3.2.3. Swap mutation operator

The main task of the mutation operator is to maintain the diversity of the population in the successive generations and to exploit
the solution space. Generally, in the population all individuals in the
population are checked bit by bit and the bit values are randomly
reversed according to a pre-specified rate. The mutation operator
consists of swapping any two randomly chosen genes in a chromo-
some (Cheng & Gen, 1997). A modification is incorporated into the
well-known swap mutation operator to strengthen its influence on
the GA. This modification is called “mutation strength” and is sim-
ply the measure of the strength of the mutation operator in terms
of the maximum number of swap moves that are performed. If the
strength of the mutation operator is chosen to be one, then it pe-
forms a single swap move if a given probability \( P(M) \) is satisfied.
For instance, when the strength of the mutation operator is se-
lected to be four, then the mutation operator performs at most four
consecutive swaps on the individual chromosome. These swaps are
applied on totally random locations, and therefore the four swap
moves are independent. However, in this paper the mutation se-
lects all columns of the chromosomes at the current population
randomly in terms of the probability of mutation then chooses
two positions of that column at random then swaps the jobs on
these positions as depicted in Fig. 7.

3.3. Parent selection strategy

Parent selection is important in regulating the bias in the repro-
duction process. The parent selection strategy means that how to
choose the chromosomes in the current population that will create
offspring for the next generation. Generally, it is better that the
best solutions in the current generation have more chance for se-
lected as parents for creating offspring. The most common method
for the selection mechanism is the “roulette wheel” sampling, in
which each chromosome is assigned a slice of a circular roulette
wheel and the size of the slice is proportional to the chromosome’s
fitness. The wheel is spun \( \text{Pop} \times \text{Size} \) times. On each spin, the chro-
mosome under the wheel’s marker is selected to be in the pool
of parents for the next generation.

3.4. Offspring acceptance strategy

We use a semi-greedy strategy to accept the offspring gener-
ated by the genetic operators. In this strategy, an offspring is ac-
ccepted for the new generation if its fitness is less than the
average fitness of its parent(s). This strategy reduces the computa-
tional time of the algorithm and leads to a monotonous conver-
gence toward the optimum solution neighborhood.

3.5. Stoppage rules

We use two criteria as stoppage rules: (1) Maximum number of
elapsed generation \( g \) that is a common criterion; and (2) standard
deviation of the fitness value of chromosomes in the current
generation (Tavakkoli-Moghaddam & Safaei, 2006). This parameter
implies the degree of diversity or similarity in the current popu-
lation in terms of the objective function value. If this criterion
reduces below an arbitrary constant, say \( \varepsilon \), then the algorithm
is stopped. The standard deviation of the fitness value of chromo-
somes in generation \( g \) is calculated as

\[
\sigma_g = \sqrt{\frac{1}{\text{Pop} \times \text{Size}} \sum_{k=1}^{\text{Pop} \times \text{Size}} \left( F_k - \overline{F}_g \right)^2}
\]

where \( F_k \) is the fitness of the \( k \)th chromosome in generation \( g \), \( \overline{F}_g \) is the average fitness of all chromosomes in generation \( g \) that is calculated as

\[
\overline{F}_g = \frac{1}{\text{Pop} \times \text{Size}} \sum_{k=1}^{\text{Pop} \times \text{Size}} F_k
\]

Therefore, if \( g > G_{\text{max}} \) or \( \sigma_g < \varepsilon \) then the algorithm is stopped (Tavakkoli-
Moghaddam & Safaei, 2006).

4. Computational results

In this section, the performance of the proposed model and devel-
oped GA are verified by 30 numerical examples in the different sizes.
Small-sized examples are solved optimally by a branch-and-bound
(B&B) method under the LINGO 8.0 software on a Personal Computer
including two Intel CoreTM2 T5600@1.83 GHz processors and 512
GB RAM. All these examples are also solved by the proposed GA. The
obtained results are then compared with the solutions found by LIN-
GO 8 in terms of the objective function value (OFV) and CPU time.
Each example is solved by the proposed GA about twenty times.
Then, the average OFV and CPU time are reported.

4.1. Random instances with small sizes

Ten random instances in small sizes are considered in this pa-
per. The associated processing times of jobs are generated in a uni-
form distribution of \( U(30,180) \). Based on the preliminary tests in
these computational experiments, the population size, probability
of crossover, probability of mutation, and maximum number of
generations are set 150, 0.25, 0.1, and 100, respectively. As shown
in Table 1, the computational time of the LINGO 8 grows exponen-
tially as the instance size increases, in which the QCSP is known as
NP-hard one. Moreover, it is obvious that the proposed GA can

<table>
<thead>
<tr>
<th>No.</th>
<th>Problem Information</th>
<th>B&amp;B</th>
<th>GA</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. V</td>
<td>No. QC</td>
<td>No. job</td>
<td>CPU Time (s)</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>47</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>216</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>1811</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>4556</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>4925</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>12</td>
<td>8</td>
<td>5308</td>
</tr>
</tbody>
</table>

Average 1.9

![Fig. 7. Illustration of the swap mutation.](image-url)
obtain the near-optimal solution for small-sized instances in a reasonable time (say, 5 s). The average gap between B&B and the proposed GA in terms of the OFV for such small sizes is computed about 2% with a standard deviation 0.67 that is a promising result.

4.2. Random instances with large sizes

In this paper, we also consider ten random instances in large sizes. The associated processing times of jobs are randomly generated in a uniform distribution of U(30,180). According to the preliminary tests in these computational experiments, population size, probability of crossover, probability of mutation, and maximum number of generations of the GA are set 300, 0.25, 0.2, and 1000, respectively. To evaluate the performance of the proposed GA for solving these instances, the average gap between the GA and the best solution obtained by B&B in terms of the OFV is computed about 3.5%, as shown in Table 2.

5. Conclusion

In this paper, we have proposed an efficient genetic algorithm (GA) to solve an extended quay crane scheduling problem (QCP) specified for a container terminal. The extended QCP developed a mixed-integer programming model for the quay crane scheduling and assignment QCs to vessels simultaneously (QCSAP) that is the outstanding advantages in this paper. The extended QCP was solved by an efficient GA for real-sized instances. Because of existing the several equality constraints in the extended model, the solution representation and operators design are two important factors in order to better exploring and exploiting of the feasible space. A number of test problems are solved to verify and validate the extended model and the performance of the proposed GA. The obtained results showed a reasonable gap about 1.9% and 3.5% between the optimal solutions found by the LINGO 8 and the proposed GA in terms of the objective function value. Furthermore, the proposed GA reaches to the near-optimal solution in reasonable time.

Acknowledgement

This study was partially supported by the University of Tehran under the research Grant No. 8106043/1/06. The first author is grateful for this financial support.

Table 2

<table>
<thead>
<tr>
<th>No.</th>
<th>Problem Information</th>
<th>B&amp;B</th>
<th>GA</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. V</td>
<td>No. QC</td>
<td>No. job</td>
<td>CPU Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>12</td>
<td>8</td>
<td>5640</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>8832</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>9205</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>12</td>
<td>10800</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>10800</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>12</td>
<td>12</td>
<td>10800</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>12</td>
<td>12</td>
<td>10800</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>5</td>
<td>16</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>16</td>
<td>–</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>5</td>
<td>16</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References


