Computers & Industrial Engineering 56 (2009) 44–52

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A genetic algorithm to solve the storage space allocation problem in a container terminal

Mohammad Bazzazi [a], Nima Safaei [b,*], Nikbakhsh Javadian [a]

[a] Department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran
[b] Department of Mechanical and Industrial Engineering, University of Toronto, 5 King's College Road, Ont., Canada M5S 3G8

ARTICLE INFO

Article history:
Received 24 May 2007
Received in revised form 12 October 2007
Accepted 27 March 2008
Available online 4 April 2008

Keywords:
Storage space allocation problem
Container terminal
Genetic algorithm

ABSTRACT

In this paper, an efficient genetic algorithm (GA) is presented to solve an extended storage space allocation problem (SSAP) in a container terminal. The SSAP is defined as the temporary allocation of the inbound/outbound containers to the storage blocks at each time period with aim of balancing the workload between blocks in order to minimize the storage/retrieval times of containers. An extended version of a SSAP proposed in the literature is considered in this paper in which the type of container affects on making the decision on the allocation of containers to the blocks. In real-world cases, there are different types (as well as different sizes) of containers consisting of several different goods such as regular, empty and refrigerated containers. The extended SSAP is solved by an efficient GA for real-sized instances. Because of existing the several equality constraints in the extended model, the implementation of the GA in order to quick and facilitate achieve to the feasible solutions is one of the outstanding advantages of this paper. The performance of the extended model and proposed GA is verified by a number of numerical examples.

1. Introduction

The temporary storage of the inbound and outbound containers is one of the most important services at the container terminal that is known as the storage space allocation problem (SSAP). The storage area in the terminal is divided into the several blocks of containers. Each block consists of a number of side by side lanes with each lane including a number of container stacks that are of 4–5 tiers of containers (Zhang, Liu, Wan, Murty, & Linn, 2003).

The fast storage and retrieval of containers at the blocks is essential for the economic performance of container terminals and also shipping companies. These issues affect directly on the traffic of the handling equipment and consequently on the dwell and turnaround time of vessels. The process of the storing (or retrieving) of a container includes the time for adjusting RTGCs, picking up container, moving toward the allocation place and downloading container. Since a container must be allocated to (or picked up from) a certain place at the block, it may be necessary to relocate one or more other containers for accessing to that container. This means a higher operating time and cost for RTGCs. Thus, it can be stated that balancing workload between blocks is critical element of the efficiency of the container terminal and it is important in reducing transportation costs and keeping shipping schedules.

We extend the SSAP proposed in the literature (Zhang et al., 2003) for a container terminal located at the south of Iran namely Shahid Rajaei terminal. In this paper, the SSAP is extended in the case that the type of containers is different such as regular, empty and/or refrigerated containers. The difference between types of containers is often resulted from the difference between goods and items inside them such as foods, chemical substances, liquids, etc. This issue means that similar containers must be (or must not be) allocated to the same block, or a certain type of container must be (or must not be) allocated to a certain block. For instance, refrigerated containers must be allocated to the blocks equipped by the power point or empty containers must not be settled under a full container or the containers with a same size should be allocated to the same block. Thus, the allocation of the containers to blocks may be restricted by above-mentioned limitations.

The SSAP can be formulated as a generalized assignment problem and therefore it belongs to the NP-hard problems category. Thus, a genetic algorithm (GA) is proposed to solve the extended SSAP for the real-sized instances. Because of existence numerous equality constraints in the proposed model, the implementation of the GA in order to facilitate and quick achieve to the feasible solutions is one of the outstanding advantages of this paper.

The rest of this paper is organized as follows: a brief overview of SSAP and related literature is presented in Section 2. The extended

* Corresponding author.
E-mail address: safaei@mie.utoronto.ca (N. Safaei).

0360-8352/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2008.03.012

SSAP is formulated in Section 3. The proposed GA is developed in Section 3. Computational results are reported in Section 4 and finally Section 5 covers the conclusion.

2. Literature review

Various aspects of the operational problems in the container terminal such as resource allocation problem (Imai, Nishimura, & Papadimitriou, 2001), Quay crane scheduling (Lee, Wanga, & Miao, 2008), berth planning optimization (Legato & Mazza, 2001), Human resources management (Legato & Monaco, 2004), stowage and load planning (Imai, Sasaki, Nishimura, & Papadimitriou, 2006) and sequencing delivery and receiving operations for storage space cranes (Kima, Leea, & Hwang, 2003) have been studied until now. A brief review of the operational problems in the container terminal can found in Zhang et al. (2003) and Zhang et al. (2002).

The SSAP has been first formulated by Zhang et al. (2003) for a container terminal in Hong Kong (Zhang et al., 2003). They solved the SSAP using a *rolling-horizon approach*. For each planning horizon, they decomposed the problem into two levels and formulated each level as a mathematical programming model. At the first level, the total number of containers to be placed in each storage block in each time period of the planning horizon is set to balance two types of workloads among blocks. The second level determines the number of containers associated with each vessel that constitutes the total number of containers in each block in each period, in order to minimize the total distance to transport the containers between their storage blocks and the vessel berthing locations. In this paper, only the first level is considered by assuming the different types of containers. The rest of assumptions are the same as Zhang et al. (2003) except to the transit containers do not exist in the considered container terminal, i.e., Shahid Rajaei terminal.

No research has attempted to solve the SSAP by a meta-heuristic approach. However, some operational problems have been solved by novel optimization approaches especially GA. Imai et al. (2006) solved a multi-objective simultaneous stowage and load planning problem by GA (Imai et al., 2006). Imai, Nishimura, Hattori, and Papadimitriou (2007) solved the berth allocation problem at indented berths for mega-containerships by GA (Imai et al., 2007). Imai et al. (2006) used GA to solve the service allocation problem at a container terminal (Cordeau, Gaudioso, Laporte, & Moccia, 2007). Lee et al. (2008) used GA to solve the quay crane scheduling problem.

3. Preliminary definitions and descriptions

The following definitions are frequently used throughput paper. All definitions and explanations are adopted from Zhang et al. (2003).

- *Inbound (IB) container*: a container that is discharged from the vessel and transmitted to the storage space waiting for picking up by customers.
- *Outbound (OB) container*: a container that is bought by customer and transmitted to the storage space waiting for picking up by vessel.
- *Quay Crane* (QC): is used to discharge IBs and transit them from and load OBs and transit them to vessels.
- *Internal truck* (IT): is provided to transport containers between the QCs and the storage blocks.
- *External truck* (XT): is provided to bring OBs from customers into the storage space and pick up IBs from the storage space and deliver them to customers.

Storage space crane (or RTGC[1] in general): are used to handle the containers in storage blocks. They load containers from trucks ITs or XTs and stack them onto blocks, and retrieve containers from blocks and load them onto trucks.

- *Workload of vessel:* the number of IB and OB containers must be loaded on and unloaded from the vessel.

The container flows in a terminal are triggered by the vessel arrival process. Each vessel arrives at a designated time with specified numbers of containers to be discharged and loaded at the terminal. The berth allocation, where the vessel is to be berthed, and the stowage plan, the sequences for discharging and loading the containers of the vessel are determined well before the vessel's arrival. Thus, according to Zhang et al. (2003), containers to be allocated to the blocks can be classified into the following four types according to their status at different handling stages (Zhang et al., 2003):

- Vessel discharge containers: IB containers on vessels before they are unloaded and allocated to the storage space (C1).
- Container yard pickup containers: IB containers already in the storage space waiting for picking up by customers (C2).
- Container yard grounding containers: OB containers before they are brought in and stored in the storage space (C3).
- Vessel loading containers: OB containers already in the storage space waiting for loading to vessels (C4).

These types of containers are measured in terms of the number of containers.(TEU[2]) This unit is used throughout when we discuss the SSAP. Since the arrivals of C1 and the departures of C4 containers are directly triggered by vessel schedules, the time epochs to handle these containers (by RTGCs or QCs) are known in advance. On the other hand, we can only determine from historical data the distributions of the time epochs to handle C3 and C2 containers. There is a free storage period for these containers, usually lasting for several days before the arrivals and after the departures of the corresponding vessels. This makes the time epochs to work on C3 and C2 containers imprecise.

In general, two main objectives of a classical SSAP are:

- to minimize the (average) vessel berthing time, which is a measure of the service of a terminal to ship liners,
- to maximize the (average) throughput of QCs, which is a measure of the productivity of a terminal.

The values of these objectives reflect whether a container terminal is in a healthy operating condition. For example, given the small storage space area and the high throughput of the container terminal, a short berthing time or high QC throughput generally come along with short XT turnaround times.

In the considered container terminal, 85% of the total workload is related to the container types of C1 and C2, i.e., IB or import containers. OB containers (i.e., container types of C3 and C4) are allocated to a single block close to the berth location of the vessels that OB containers must be loaded on them. Therefore, this paper focuses only on container types of C1 and C2. The extended SSAP will decide in which blocks to place the C1 and C2 containers of each vessel by considering the type of container. We assume that the vessels are allocated to the corresponding berths and also QCs are allocated to the corresponding vessels to discharge the containers of each vessel. Thus, our problem is to determine the numbers

[1] Rail Mounted Gantry Crane.
[2] Twenty-foot Equivalent Unit.

of C1 and C2 containers of each vessel stored in each block. The pre-requisite is to determine the (expected) workload requirements (in terms of number of C1 and C2) for each time period of the storage space. For this mean, we need to know the behaviors or patterns of accumulation and dissipation of container types of C1 and C2 related to a given vessel in the storage space at each time period.

The duration of unloading and loading of a vessel can be assumed to be a known quantity for a given vessel (with the prespecified workload). According to the empirically studies, in the considered container terminal, the patterns of accumulation and dissipation are stable over time. Consequently, for any vessel, given the workload associated with it, we can deduce the (expected) workload of the vessel for any time period. By superimposing the workloads of all vessels, we can find the total workload for any time period for the whole terminal. Given the total workload requirement of the whole container terminal, we have a better idea of the demand for RTGCs, QCs, XTs and ITs over time. Clearly, it would be still too complex to consider the interaction of the storage blocks, the travel of ITs, and the availability of RTGCs and QCs all together. To overcome this problem, Zhang et al. (2003) broken down the SSAP into two levels. The first level is aimed to minimize vessel berthing times by balancing the workload of RTGCs and QCs for vessels. With workloads of a vessel dispersing in different blocks, the RTGCs in the blocks serve as parallel servers processing jobs for the vessel, and the deberthing time of the vessel is the maximal processing time of these parallel servers. Balancing the workload of parallel servers generally works well to minimize the completion times of vessels. Similar results on the RTGC deployment problem confirm that balancing workloads of blocks reduces delay in container handling (Zhang, Wan, Liu, & Linn, 2002). The second level determines the number of containers associated with each vessel that constitutes the total number of containers in each block in each period, in order to minimize the total distance to transport the containers between their storage blocks and the vessel berthing locations. As mentioned earlier, we consider only the first level in this paper by assuming the different types of IB containers. The second level is not considered because the OB containers constitute only a little portion of the total workload of the container terminal.

Zhang et al. (2003) considered a fixed planning horizon and run their method with the *rolling-horizon approach* (RHA). According to the RHA, at each planning epoch, we plan for a fixed horizon in immediate future and execute the plan accordingly up to the next planning epoch; then we formulate a new plan based on the latest information; this pattern goes on continually (see Fig. 1). There is a trade-off between the length of the chosen planning horizon, computational burden and power of prediction. A short planning horizon means less computational burden but also less predictive power about the future, while a long planning horizon may be computationally unfeasible and may include too much uncertain information. Zhang et al. (2003) settled on a planning horizon of three days, with each day being divided into six 4-h periods. Whereas, meta-heuristic approaches can process more computational burden than the classical methods within a unit time, we settle on a planning horizon of four days, with each day being divided into six 4-h periods that makes our predication more exact than the three days planning horizon. At (the beginning of) day 1, a storage space allocation plan is formed for the 24 periods in days 1–5. Only the first day of the plan is executed and a new four-day plan is formed at the end of the first day (beginning of the second day) based on the latest information. This goes on for every day.

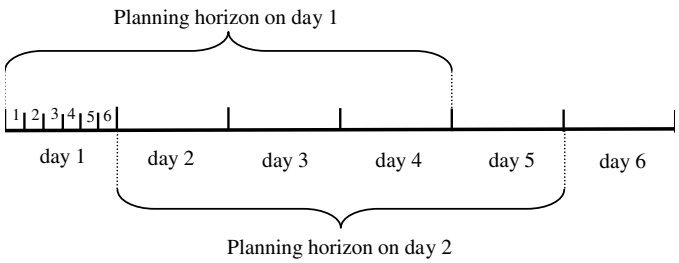

Fig. 1. Rolling of the planning horizon.

The maximum dwell times of IB containers approximately equal the maximal free storage period, which is beyond the planning horizon. Thus, there are containers with unknown departure times at the moment of planning or containers with known departure times beyond the planning horizon. Their workloads do not occur in the planning horizon and consequently such containers are not directly included in the storage allocation model. To account for their possible effect in future, these containers are distributed to blocks in proportion to their available storage capacities at the beginning of the planning horizon so as to balance the block densities. Such an approximation has a marginal effect on the overall performance: The majority of containers of a vessel are accumulated and dissipated in the planning horizon (within four days before or after the vessel's berthing) and, in any case, most containers are allocated under known information, since only the first day of the four-day plan is implemented. By attention to the above-mentioned descriptions, four data types should be available for each IB container as follows:

(1) The time period in which the container must be discharged from vessel and brought to the storage blocks.
(2) The time period in which the container must be removed from space blocks and loaded on the vessel.
(3) The type of each container.
(4) The allowable blocks for each container type.

4. Problem formulation

In this section, the extended SSAP is formulated as a mathematical programming model based on the following assumptions. Basically, we want to determine the numbers of C1 and C2 containers stored in each block for each planning period.

4.1. Assumptions

(1) Only IB containers (i.e., container types of C1 and C2) are considered as the workload that must be relocated between the vessel berthing locations and storage space.
(2) There is enough resource, i.e., RTGC, QC and IT, to handle the workload at the considered container terminal.
(3) Containers are of different types and sizes. Thus, the workload related to each type of container is counted in terms of the number of that container type. The containers of different sizes and types can be mixed in blocks.
(4) The allowable blocks that a container type can be allocated to them are known in advance.

4.2. Input parameters

B	the total number of blocks in the storage space
T	the total number of planning periods in a planning horizon, T=24
R	the number of container types
C_i	the storage capacity of block i, $1 \leqslant i \leqslant B$
$\widetilde{D}_{tkr}$	the expected number of C1 container type r that are discharged from vessels in period t and to be picked up by customers in period $t+k$, $1 \leqslant t \leqslant T$, $0 \leqslant k \leqslant T-t$

β_{itr} the expected number of C1 container type r discharged from vessels in period t, allocated to block i (determined by the proportional method), with an unknown pickup time or a pickup time beyond the planning horizon, $1 \leqslant t \leqslant T$, $1 \leqslant i \leqslant B$

P^0_{itr} the expected total number of initial C2 container type r that arrive at the container terminal in period t and to be loaded onto the vessels in period $t+k$, $1 \leqslant t \leqslant T$, $0 \leqslant k \leqslant T-t$

V_{ir} the initial inventory of container type r in block i, i.e., the number of container type r in block i at the beginning of the planning horizon, $1 \leqslant i \leqslant B$

$S_{ir} = 1$ if container type r can be allocated to block i; otherwise $S_{ir} = 0$

η allowable density for each block

M an arbitrary positive big number

4.3. Decision variables

D_{itkr} the number of C1 container type r with full information stored in block i that are discharged from vessels in period t and to be picked up in period $t+k$, $1 \leqslant i \leqslant B$, $1 \leqslant t \leqslant T$, $0 \leqslant k \leqslant T-t$

D_{itr} the total number of C1 container type r (with full or partial information) stored in block i that are discharged from vessels during period t, $1 \leqslant i \leqslant B$, $1 \leqslant t \leqslant T$

P_{itr} the total number of C2 container type r stored in block i that are picked up by customers in period t, $1 \leqslant i \leqslant B$, $1 \leqslant t \leqslant T$

V_{itr} the number of container type r (consists of C1 and C2 containers) within block i at the end of period t, $1 \leqslant i \leqslant B$, $1 \leqslant t \leqslant T$

4.4. Mathematical model

4.4.1. Objective function

By attention to the above-mentioned notions and the explanations presented in Section 3, the objective function of the model is written as follow:

$$\min Z = \sum_{t=1}^{T}\sum_{r=1}^{R}\left(w_1\left[\max_{i=1}^{B}\{D_{itr}\} - \min_{i=1}^{B}\{D_{itr}\}\right] + w_2\left[\max_{i=1}^{B}\{D_{itr}+P_{itr}\} - \min_{i=1}^{B}\{D_{itr}+P_{itr}\}\right]\right) \quad (1)$$

Eq. (1) balances C1 containers and the total number of containers among blocks for each container type at each period. Generally, the term "max(.) – min(.)" implies the imbalance between workload allocated to blocks. In other word, the term max(.) = min(.), in a goal condition, means the containers are distributed between blocks uniformly. The first term of the objective function focuses on balancing container type C1 while the second term of the objective function focuses on balancing container types C1 and C2 simultaneously. The importance of each term is determined by pre-determined weights. In this Equation, D_{itr} is the expected total number of vessel related containers that need to be handled in block i during period t and $D_{itr} + P_{itr}$ is the expected total number of containers (related to the vessel and customer) to be handled in block $i$during period t. Therefore, the two terms of Eq. (1) measure the imbalances of the discharging containers and of the total number of containers in the blocks in each planning period, respectively. w_1 and w_2, the weights of the two terms of Eq. (1), are adjusted according to the relative importance of the discharging containers within the total number of containers as interpreted by a terminal. Eq. (1) is the same objective function presented by Lee et al. (2008) without considering C3 and C4 containers.

4.4.2. Constraints

(a) Container flow conservation constraints

$$\sum_{i=1}^{B} D_{itkr} = \widetilde{D}_{tkr} \qquad r = 1,2,\ldots,R, \quad K = 1,2,\ldots,T-t, \quad t = 1,2,\ldots,T \quad (2)$$

Constraint (2) ensures that the expected total number of C1 containers type r with full information waiting for the allocation, D_{tkr}, is the sum of these containers assigned to all the blocks.

$$D_{itr} = \beta_{itr} + \sum_{k=0}^{T-t} D_{itkr} \qquad r = 1,2,\ldots,R \quad i = 1,2,\ldots,B, \quad t = 1,2,\ldots,T \quad (3)$$

Constraint (3) ensures that the expected total number of C1 containers type r allocated to block i during period t, D_{itr}, is the sum of the total number of C1 containers with full information, D_{itkr}, and of those containers with unknown departure times at the planning horizon, β_{itr}.

(b) Constraints on C2 container

$$P_{itr} = \sum_{k=0}^{t-1} D_{i(t-k)kr} + P^0_{itr} \qquad i = 1,2,\ldots,B, \quad t = 1,2,\ldots,T, \quad r = 1,2,\ldots,R \quad (4)$$

Constraint (4) indicates that the number of C2 containers type r handled in block i during period t, P_{itr}, consists of two parts. The first part is the containers transferred from the corresponding C2 containers that arrived in the planning horizon. The second part is C2 containers initially stored in block i to be loaded onto the vessels in period t in the current planning horizon.

(c) Block density constraints

$$V_{itr} = V_{i(t-1)r} + \widetilde{D}_{itr} - P_{itr} \qquad i = 1,2,\ldots,B, \quad t = 1,2,\ldots,B, \quad r = 1,2,\ldots,R \quad (5)$$

$$V_{itr} \leqslant \eta C_i \qquad i = 1,2,\ldots,B, \quad t = 1,2,\ldots,B, \quad r = 1,2,\ldots,R \quad (6)$$

Constraint (5) represents the updating of inventory, V_{itr}, from period to period. Constraint (6) ensures that the inventory of each block in each planning period will not exceed the allowable block density. The block density implies this fact that a portion of the block's space is used to transfer containers within the block by RTGCs and ITs.

(d) Container type allocation constraint

$$\widetilde{D}_{itr} \leqslant M \times S_{ir} \qquad i = 1,2,\ldots,B, \quad t = 1,2,\ldots,B, \quad r = 1,2,\ldots,R \quad (7)$$

Constraint (7) ensures that each container type is allocated only to the allowable blocks.

(e) Integer constraint

All decision variables take up non-negative integer values.

The proposed model is non-linear because of using functions *Max*( ) and *Min*( ) in the objective function. It can be converted to a linear model as depicted in Zhang et al. (2003).

5. Genetic algorithm implementation

As mentioned earlier, the main contribution of this paper is the implementation of an evolutionary algorithm, say genetic algo-

rithm (GA), to solve an extended SSAP problem. Our reasons for choosing GA as a solution approach are as follows:

1. We needed a stochastic approach with a strong exploration ability to search the feasible space. As you will see, the most of the model's constraints are as equality form and, therefore, obtaining of the feasible solutions is a hard task. In this case, the probability of reaching infeasible solutions is more than feasible solutions and therefore we need a population-based approach such as GA to better exploration of the solution space.
2. GA is a well-known meta-heuristic that its efficiency is verified for many problems in the literature.
3. The SSAP is not solved by any meta-heuristic until now.

GA are well-known meta-heuristic approach inspired by the natural evolution of the living organisms. GAs work on a population of the solutions simultaneously. They combine the concept of survival of the fittest with structured, yet randomized, information exchange to form robust exploration and exploitation of the solution space. The exploration process is performed by a genetic operator namely *Crossover* and the exploitation process in performed by another genetic operator namely *mutation*. The trade-off between these two processes is controlled by the *parent* selection and *offspring* acceptance strategies. The initial and most important step of the GA implementation is the solution representation or *chromosome* design.

As can be seen in the model proposed in Section 4, most of the model's constraints are as equality, i.e., constraints No. (2)–(5). This issue makes difficult the implementation of the GA and causes GA spends so much time and efforts to access to the feasible solutions especially by increasing the size of the problem. Thus, the chromosome representation and genetic operators design are two important tasks in the GA implementation for quick access to the feasible space and an effective movement toward the optimal solution neighborhood. Following subsections present how implementation of the GA to solve the proposed model.

5.1. Solution representation

By considering the main decision variable of the proposed model, D_{itkr}, we use a four-dimension structure to represent the solution of the extended SSAP. These four dimensions indicate the indexes related to the allocated block, period of discharging, period of picking up and the type of the container. Fig. 2 shows a typical solution representation as a two-dimension structure by assuming $B = 2$, $T = K = 4$ and $R = 2$.

For example, to satisfy constraint (2), equations $\widetilde{D}_{111} = D_{1111} + D_{2111}$, $\widetilde{D}_{112} = D_{1112} + D_{2112}$, …, $\widetilde{D}_{442} = D_{1442} + D_{2442}$ must be observed. Also, to satisfy Constraint (3), equation $\widetilde{D}_{212} = D_{2112} + D_{2122} + D_{2132} + D_{2142}$ must be observed.

5.2. Initial solution generation procedure

To generate the random solutions for the initial population, the procedure is as follow.

(1) Generate randomly D_{itkr} values within interval $[\max\{\widetilde{D}_{tkr}\}, \min\{\widetilde{D}_{tkr}\}]$ where $\sum_{i=1}^{B} S_{ir} D_{itkr} = \widetilde{D}_{tkr} \forall i, t, k, r$. This is simply possible by using the traditional linear combination.
(2) Calculate D_{itr} values by using constraint (3).
(3) Calculate P_{itr} values by using constraint (4).
(4) Calculate V_{itr} values by using constraint (5) and considering constraint (6). If constraint (6) is violated, then, add value $\lambda(V_{itr} - \eta C_i)$ to the objective function as a penalty term in which λ indicates the penalty coefficient.

As a result, the fitness function of the chromosomes is defined as: Eq. (1) plus the penalty term resulted from the violation of constraint (6).

Fig. 3 shows a randomly generated typical example by assuming $B = 4$, $T = K = 4$, $R = 1$. For simplicity, we temporarily ignore the container type in this example, i.e., $R = 1$. In Fig. 3, D_{itk} values are generated randomly in such a way that constraints (2) and (7) are satisfied.

By considering the information provided in Fig. 3, D_{it} values are obtained as follows: For instance, assumeβ_{i1}=0 $\forall i$, thus, we have:

$$D_{11} = \sum_{k=0}^{T-1} D_{11k} + \beta_{11} = (9 + 14 + 10) + 0 = 33 + 0 = 33,$$

$$D_{21} = \sum_{k=0}^{T-1} D_{21k} + \beta_{21} = 32$$

$$D_{31} = \sum_{k=0}^{T-1} D_{31k} + \beta_{31} = 32, \quad D_{41} = \sum_{k=0}^{T-1} D_{41k} + \beta_{41} = 33, \ldots$$

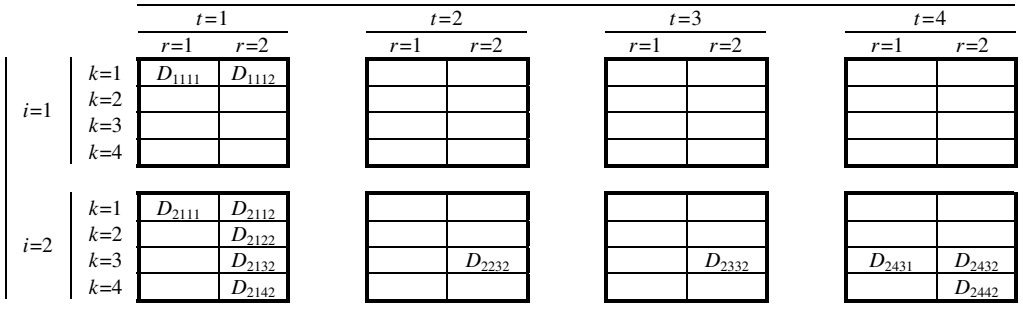

Fig. 2. Solution representation.

	i=1				i=2				i=3				i=4			
D_{11}=33	**9**	**14**	**10**	**0**	**D_{21}=32**	**9**	**14**	**0**	**9**	**D_{31}=33**	**9**	**0**	**18**	**5**	**D_{41}=33**	**9**	**8**	**6**	**10**
D_{12}=35	0	2	1	31	**D_{22}=36**	0	9	20	7	**D_{32}=36**	14	11	11	0	**D_{42}=34**	6	22	6	0
D_{13}=19	16	3	0	0	**D_{23}=20**	19	1	0	0	**D_{33}=21**	0	11	0	10	**D_{43}=20**	0	10	10	0
D_{14}=30	0	30	0	0	**D_{24}=30**	20	10	0	0	**D_{34}=30**	30	0	0	0	**D_{44}=30**	0	10	20	0

Fig. 3. Typical example.

Thus,

$$\max_{i=1}^{4}\{D_{i1}\} = 43, \min_{i=1}^{4}\{D_{i1}\} = 42$$

By assuming $w_1 = 1$ and $w_2 = 0$, the objective function value is obtained as follow:

$$\min Z = \sum_{t=1}^{T}[\max_{i=1}^{4}\{D_{it}\} - \min_{i=1}^{4}\{D_{it}\}] = 1 + 2 + 2 + 0 = 5.$$

5.3. Genetic operators design

5.3.1. 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 simultaneously. The AC produces a new offspring as complimentary linear combination of the parents as follow:

$$\text{Offspring} \equiv \lambda \times \text{Parent1} + (1 - \lambda) \times \text{Parent2},$$

where λ is a randomly generated number within interval (0,1). Thus, D_{itkr} values in the newly generated offspring is obtained as: D_{itkr} (Offspring) = $\lambda \times D_{itkr}$ (Parent1) + $(1 - \lambda) \times D_{itkr}$(Patent2). The AC guarantees the generated offspring will remain feasible if its parents are feasible. Since, D_{itkr} are integer values, the integer part of D_{itkr} (Offspring), i.e., $[D_{itkr}(\text{Offspring})]$, must be considered as a true value. Thus, constraint (2) may be not satisfied and we incur an error as $\varepsilon = \sum_{i=1}^{B}[D_{itkr(\text{offspring})}] - \widetilde{D}_{tkr}$. To overcome this problem, ε value is heuristically added to D_{itkr} with the minimum value, i.e., $\min_i\{D_{itkr}\}\ \forall\ t, k, r$.

5.3.2. Stepping stone mutation

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. In this paper, a mutation operator, called Stepping Stone mutation (SSM), is used that is inspired by the Stepping Stone method for solving the classical transportation problem. The SSM guarantees the generated offspring will remain feasible if its parent is feasible. The procedure of the SSM is as follows:

(1) Select randomly a solution from current population.
(2) For each r, k and t let $D_{ptkr} = \max_{i=1}^{B} D_{itkr}$, $D_{qtkr} = \min_{i=1}^{B} D_{itkr}$ and $\delta_{tkr} = D_{ptkr} - D_{qtkr}$.
(3) Change values D_{ptkr} and D_{qtkr} to $D_{ptkr} \to D_{ptkr} - \left\lfloor \frac{\delta_{tkr}}{2} \right\rfloor$ and $D_{qtkr} \to D_{qtkr} + \left\{\delta_{tkr} - \left\lfloor \frac{\delta_{tkr}}{2} \right\rfloor\right\}$, respectively.

For example, consider the condition of the example presented in Fig. 3 and assume $D_{12} = 35$, $D_{22} = 36$, $D_{32} = 36$ and $D_{42} = 34$. Thus, according to Step 2, $D_{p1} = 36$; $p = 2, 3$ and $D_{q1} = 34$; $q = 4$ and therefore $\delta_{11} = 36 - 34 = 2$. Consequently, values of D_{p1} and D_{q1} are mutated as follows:

$$D_{p1} = 36 - \left\lfloor \frac{2}{2} \right\rfloor = 35, D_{q1} = 34 + \left\{2 - \left\lfloor \frac{2}{2} \right\rfloor\right\} = 35.$$

5.4. Parent selection strategy

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 selected 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 *Pop_Size* times. On each spin, the chromosome under the wheel's marker is selected to be in the pool of parents for the next generation. *Pop_size* is the population size or the number of chromosome at each population that is known in advance.

5.5. Offspring acceptance strategy

We use a semi-greedy strategy to accept the offspring generated by the genetic operators. In this strategy, an offspring is accepted for the new generation if its fitness be less than the average fitness of its parent(s). This strategy reduces the computational time of the algorithm and leads to a monotonous convergence toward the optimum solution neighborhood (see Fig. 6 in Section 6).

5.6. Stoppage rules

We use two criteria as stoppage rules: (1) maximum number of elapsed generation ($G_{\max}$) that is a common criterion and (2) the 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 population in terms of the objective function value. If this criterion reduces below an arbitrary constant, say ε, then the algorithm stopped. The standard deviation of the fitness value of chromosomes in generation g is calculated as $\sigma_g = \left[(1/Pop\_Size)\sum_{k=1}^{Pop\_Size}\left(F_g^k - \overline{F}_g\right)^2\right]^{\frac{1}{2}}$ where F_k^g is the fitness of kth chromosome in generation g. $\overline{F}_g$ is average fitness of all chromosomes in generation g that is calculated as $\overline{F}_g = (1/Pop\_Size)\sum_{k=1}^{Pop\_Size}F_g^k$. Therefore, if $g > G_{max}$ or $\sigma_g \leqslant \varepsilon$ then the algorithm stopped (Tavakkoli-Moghaddam & Safaei, 2006).

6. Computational results

In this section, the performance of the proposed model and developed GA are verified by 22 numerical examples in the different sizes. Small-sized examples are optimally solved 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. However, it is not possible to obtain an optimal solution for the large-sized examples in a reasonable CPU time. All examples are also solved by GA and the obtained results are compared with the optimal solutions in terms of the objective function value (OFV) and CPU time. Each example is solved by GA 20 times and the mean of OFV and CPU time are reported. For the large-sized problems, the run time of LINGO is limited to 3 h. Thus, the best solution obtained after 3 h is reported for the large-sized problems. This limitation is determined based on the quality of the solutions obtained by GA. However, it cannot be found a feasible solution after 3 h for very large examples.

For better understanding and verifying the performance of the proposed model, a typical example in the real environment is considered as follow. Assume that Vessel A is being discharged with start of the previous planning horizon (i.e., previous four days). Another vessel, say, Vessel B arrives the terminal and it is ready to berth. The discharging process for Vessel B will be stared at the beginning of the first period of new planning horizon. Third vessel, say, Vessel C arrives the terminal and will stay in queue to discharge for the second period of new planning horizon. In general, the birthing time in the considered terminal is averagely 4 h (i.e.,

equal to the one period). Without loss of generality, the birthing time is ignored in the proposed model because it can be interpreted as the availability time of the vessel that is the same value for all vessels. The considered terminal has four blocks and three berthing locations which Vessel A berthed in one of locations, say, location 2. Also, two types container, i.e., regular and refrigerated containers, are considered in this terminal that must be allocated to the allowable blocks. The remaining information and data sets are provided in Tables 1–5. Column "**a**" in Tables 1–3 is related to the containers with a known pickup time while Column "**b**" is related to the containers with an unknown pickup time. The data provided in Tables 1–3 are used to compute parameter β_{irt}. For instance, consider the bolded values in Tables 2 and 3, i.e., values 100 and 0 so that 100 + 0 = 100 is equal to the total number of containers type 1 with unknown pickup time loaded on Vessels B and C that must be discharged in current planning horizon. As mentioned in Section 3, to account for the possible effect of the containers with the unknown pickup time, these containers are distributed to blocks in proportion to their available storage capacities at the beginning of the planning horizon so as to balance the block densities. Thus, these 100 containers should be proportionally distributed between the allowable blocks of container type 1, i.e., blocks 2, 3 and 4 (see Table 4). Consequently, values β_{111}, β_{211} and β_{311} are set to 35, 30 and 35, respectively, as shown in Table 5 (i.e., underlined values). The optimum solution resulted from data provided in Tables 1–5 is presented in Table 6. This table shows the values of decision variable D_{itkr}. The corresponding objective function value is obtained as 0.7.

The comparison between optimum and GA results for small and large-sized instances are shown in Tables 7 and 8, respectively. The average of the relative gap between B&B and GA in terms of the OFV for small-sized instances is obtained about 4% with a standard deviation 2.26 that is a promising result. However, as you can see in Fig. 4, the CPU times of B&B and GA are not obviously comparable. The exponential trend of the B&B's CPU time by increasing the size of instances is tangible in Fig. 4.

As shown in Table 8, only in six of eleven instances, B&B could access to the feasible space within 3 h. However, the average of the relative gap between GA and the best solution obtained by B&B in terms of the OFV for six above-mentioned instances is obtained about 5%. In this case, as you can see in Fig. 5, GA's CPU time shows a polynomial behavior when the size of the instance increases. As discussed in Section 5.5, Fig. 6 shows a typical convergence of GA during 20 successive generations related to a single run.

Table 4
The initial inventory of each block for each container type and container type allocation constraint

S_{ir} >			v_{ir}		
i	r		i	r	
	1	2		1	2
1	0	1	1	0	260
2	1	0	2	54	0
3	1	0	3	44	0
4	1	0	4	34	0

7. Conclusion

This paper proposed an efficient genetic algorithm (GA) to solve an extended storage space allocation problem (SSAP) specified for a

Table 1
Data related to the containers loaded on Vessel A with a known or unknown pickup time within

t = 4				t = 3				t = 2				t = 1				k
r = 2		r = 1		r = 2		r = 1		r = 2		r = 1		r = 2		r = 1		
b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	
						20	20			50	50			50	50	1
											50				50	2
											50				50	3
											40				40	4

Table 2
Data related to the containers loaded on Vessel B with a known or unknown pickup time

t = 4				t = 3				t = 2				t = 1				k
r = 2		r = 1		r = 2		r = 1		r = 2		r = 1		r = 2		r = 1		
b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	
			60			50	50			50	50			**100**	50	1
			60				50				50				50	2
			60				50				50				40	3
			60				40				40					4

Table 3
Data related to the containers loaded on Vessel C with a known or unknown pickup time within

t = 4				t = 3				t = 2				t = 1				k
r = 2		r = 1		r = 2		r = 1		r = 2		r = 1		r = 2		r = 1		
b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	
10	60				50	30	30			40	50			**0**		1
	60				50						50					2
	60				50						50					3
	50				30						50					4

Table 5
Data related to parameters P^0_{itr}, $\tilde{D}_{tkr}$ and β_{itr}

	$\tilde{D}_{tkr}$									β_{itr}									P^0_{itr}								
t	1		2		3		4		t	1		2		3		4		t	1		2		3		4	
k	r								k	r								k	r							
	1	2	1	2	1	2	1	2		1	2	1	2	1	2	1	2		1	2	1	2	1	2	1	2
1	50	0	100	0	80	50	60	60	1	0	0	**35**	0	**30**	0	**35**	0	1	0	100	0	0	0	0	0	0
2	50	0	100	0	50	50	60	60	2	0	0	30	0	30	0	30	0	2	0	60	15	0	10	0	15	0
3	40	0	100	0	50	50	60	60	3	0	0	25	0	30	0	25	0	3	0	50	0	0	0	0	0	0
4	0	0	90	0	40	30	60	50	4	0	10	0	0	0	0	0	0	4	0	50	0	0	0	0	0	0

Table 6
Optimum solution of the typical example (D_{itkr} values)

t	k	$i=1$		$i=2$		$i=3$		$i=4$	
		$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$	$r=1$	$r=2$
1	1	0	0	35	0	20	0	15	0
	2	0	0	30	0	20	0	0	0
	3	0	0	0	0	10	0	30	0
	4	0	0	0	0	0	0	0	0
2	1	0	0	20	0	30	0	50	0
	2	0	0	26	0	44	0	30	0
	3	0	0	28	0	40	0	32	0
	4	0	0	56	0	16	0	18	0
3	1	0	50	48	0	19	0	13	0
	2	0	50	0	0	38	0	12	0
	3	0	50	0	0	0	0	50	0
	4	0	30	27	0	13	0	0	0
4	1	0	60	0	0	0	0	60	0
	2	0	60	20	0	20	0	20	0
	3	0	60	0	0	60	0	0	0
	4	0	50	60	0	0	0	0	0

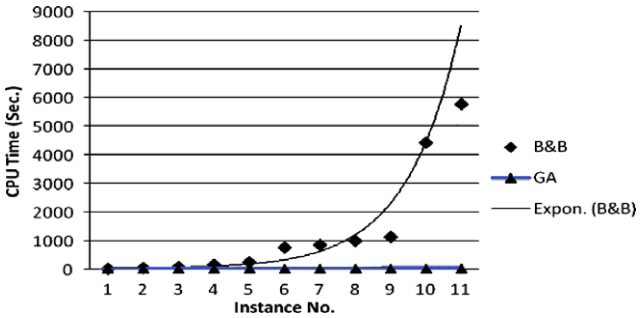

Fig. 4. Comparison between B&B and GA's CPU times.

container terminal. The extended SSAP considers the type of container as well as other advantages included in the problem. In real-world cases, there are different types (as well as different sizes) of containers such as regular, empty or refrigerated containers. The type of container may be affect on the making the decision

Table 7
Comparison between the optimum and GA runs for the small-sized instances

No.	Problem information				B&B		GA		
	B	T	K	R	CPU time (s)	OFV	Mean of CPU time (s)	Mean of OFV	Gap (%)
1	2	2	2	2	2	0	5	0	0
2	3	2	2	2	35	1.2	7.5	1.2	0
3	4	2	2	2	68	2.2	8	2.3	4.54
4	5	2	2	2	150	2.8	9.2	2.9	3.57
5	6	2	2	2	240	3	10	3.2	6.67
6	3	3	3	2	750	4	10.3	4.2	5.00
7	4	3	3	2	840	4.4	12.4	4.6	4.54
8	5	3	3	2	980	4.8	14	5	4.16
9	6	3	3	2	1,120	5.2	16.2	5.5	5.76
10	4	4	4	2	4,400	5.6	18	5.8	3.57
11	5	4	4	2	5,740	6	24	6.4	6.67
Average									4.04

Table 8
Comparison between Optimum and GA runs for large-sized instances

No.	Problem information				B&B		GA		
	B	T	K	R	CPU time (s)	OFV	Mean CPU time(s)	OFV	Gap (%)
1	6	4	4	2	7,650	6.4	30	6.7	4.68
2	7	5	5	2	10,800	7.8*	80	8.2	5.12
3	8	6	6	2	10,800	10*	150	10.4	4.00
4	9	7	7	2	10,800	13.6*	270	14.2	4.41
5	10	8	8	2	10,800	20.6*	310	21.8	5.82
6	11	9	9	2	10,800	25.8*	360	27	4.65
7	12	10	10	2	10,800	–	440	35.4	–
8	13	11	11	2	10,800	–	635	48.6	–
9	14	12	12	2	10,800	–	852	60.2	–
10	15	13	13	2	10,800	–	970	67	–
11	16	14	14	2	10,800	–	1100	80	–
Average									4.78

* Best solution found after 3 h.

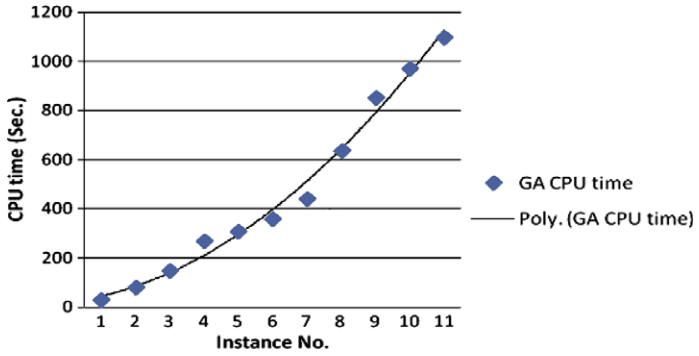

Fig. 5. Polynomial trend of the GA's CPU time for the large-sized problems.

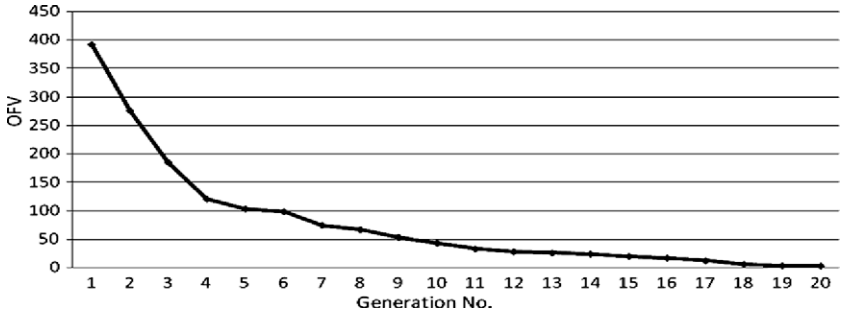

Fig. 6. Typical convergence of GA during 20 successive generations related to a single run.

on the allocation of containers to the storage blocks. For instance, refrigerated containers must be allocated to the blocks equipped by the power point or empty containers must not be settled under a full container or the containers with a same size should be allocated to the same block. Thus, it is necessary to consider the type of container in the SSAP.

The extended SSAP is 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 exploration and exploitation of the feasible space. A typical example is optimally solved to verify the extended model and the performance of the proposed GA is verified by a number of numerical examples. The obtained results showed a relative gap about 5% between GA and optimum solution in terms of the objective function value.

References

Cordeau, J. F., Gaudioso, M., Laporte, G., & Moccia, L. (2007). The service allocation problem at the Gioia Tauro Maritime Terminal. *European Journal of Operational Research, 176*(2), 1167–1184.

Imai, A., Sasaki, K., Nishimura, E., & Papadimitriou, S. (2006). Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research, 171*(2), 373–389.

Imai, A., Nishimura, E., Hattori, M., & Papadimitriou, S. (2007). Berth allocation at indented berths for mega-containerships. *European Journal of Operational Research, 179*(2), 579–593.

Imai, A., Nishimura, E., & Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port. *Transportation Research Part B, 35*(4), 401–417.

Kima, K. H., Leea, K. M., & Hwang, H. (2003). Sequencing delivery and receiving operations for yard cranes in port container terminals. *International Journal of Production Economics, 84*, 283–292.

Lee, D. H., Wanga, H. Q., & Miao, L. (2008). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E: Logistics and Transportation Review, 44*(1), 124–135.

Legato, P., & Mazza, R. M. (2001). Berth planning and resource planning optimization at a container terminal via discrete even simulation. *European Journal Operational Research, 133*, 537–547.

Legato, P., & Monaco, M. F. (2004). Human resources management at a marine container terminal. *European Journal of Operational Research, 156*, 769–781.

Tavakkoli-Moghaddam, R., & Safaei, N. (2006). An evolutionary algorithm for a single-item resource-constrained aggregate production planning problem. In *IEEE Congress on Evolutionary Computation*, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16–21.

Zhang, C., Liu, J., Wan, Y. W., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. *Transportation Research Part B, 37*, 883–903.

Zhang, C., Wan, Y.-w., Liu, J., & Linn, R. (2002). Dynamic crane deployment in container storage yards. *Transportation Research B, 36*(6), 537–555.