International Journal of Computational Intelligence Systems, Vol. 6, No. 2 (March, 2013), 244-260

Gantry Crane Scheduling with Interference Constraints in Railway Container Terminals

Peng Guo*, Wenming Cheng, Zeqiang Zhang, Min Zhang
School of Mechanical Engineering, Southwest Jiaotong University
Chengdu, 610031, P.R. China

Jian Liang
School of Mechanical Engineering & Automation, Xihua University
Chengdu, 610039, P.R. China

Received 8 August 2011

Accepted 25 August 2012

Abstract

Railway container terminals, where gantry cranes are responsible for loading and unloading containers between freight trains and yards, are important hubs of hinterland logistics transportation. Terminal managers confront the challenge in improving the efficiency of their service. As the most expensive equipment in a terminal, the operational performance of gantry cranes is a crucial factor. In this paper, the gantry crane scheduling problem of railway container terminals is investigated. A mixed integer programming model which considers the effect of dwelling position dependent processing times is formulated. In addition, the safety distances, the travel times and the non-crossing requirement of cranes are incorporated in the mathematical model. A novel discrete artificial bee colony algorithm is presented to solve the intractable scheduling problem. Computational experiments are conducted to evaluate the proposed algorithm on some randomly constructed instances based on typical terminal operational data. Experimental results show that the proposed approach can obtain near optimal solutions for the investigated problem in a reasonable computational time.

Keywords: Railway container terminal; Gantry crane scheduling; Interference constraint; Artificial bee colony algorithm.

1. Introduction

With the globalization of trade, container transportation is becoming more and more popular. The freight transport volumes of China have kept an unprecedented increase in the last decades, especially in rail freight. The volume of Chinese railway freight has increased by about 37% from the year 2006 to the year 2011[1]. Recently, railway container terminals that serve as the key substantial hubs of hinterland logistics transportation have successively been built in China. Rising competition from road freight, marine cargo and air cargo, have put pressure on managers of these terminals to improve their competitiveness. In terms of terminal competitiveness, it is often measured by the time necessary to serve trains by gantry cranes (GCs), which are the most important and expensive equipment used in terminals. Compared with other competition indexes, the freight train handling time which is the latest completion time among all operating tasks of the container train is a commonly critical factor. Generally,

*Corresponding author. E-mail address: pengguo318@gmail.com (Peng Guo).

the objective is mainly determined by gantry crane scheduling and train dwelling allocation in practice.

In a railway container terminal, there are a number of railroad track working areas along the railway line and each area is served by a number of gantry cranes. Simultaneously, some trucks are busy transferring containers from railroad track working areas to assistant container yards or vice versa. Fig. 1 illustrates a schematic representation of a railway container terminal. When a container train arrived at the terminal, the operator must decide which railroad track operating area is the most suitable dwelling position of the train. Normally, a desired dwelling position is specified for a train within the vicinity of these container yards in advance. If an actually chosen dwelling position is apart from the desired position, the load of the horizontal transport will increase. Thus, the processing times of some operating tasks of the train are extended. Sometimes, if the distance between the desired position and the actual position is long enough, the effect on the processing time of these discharging/loading tasks is very obvious. In that case, it cannot be ignored. The above descriptive situation is called the effect of the dwelling position dependent processing times for the container terminal.

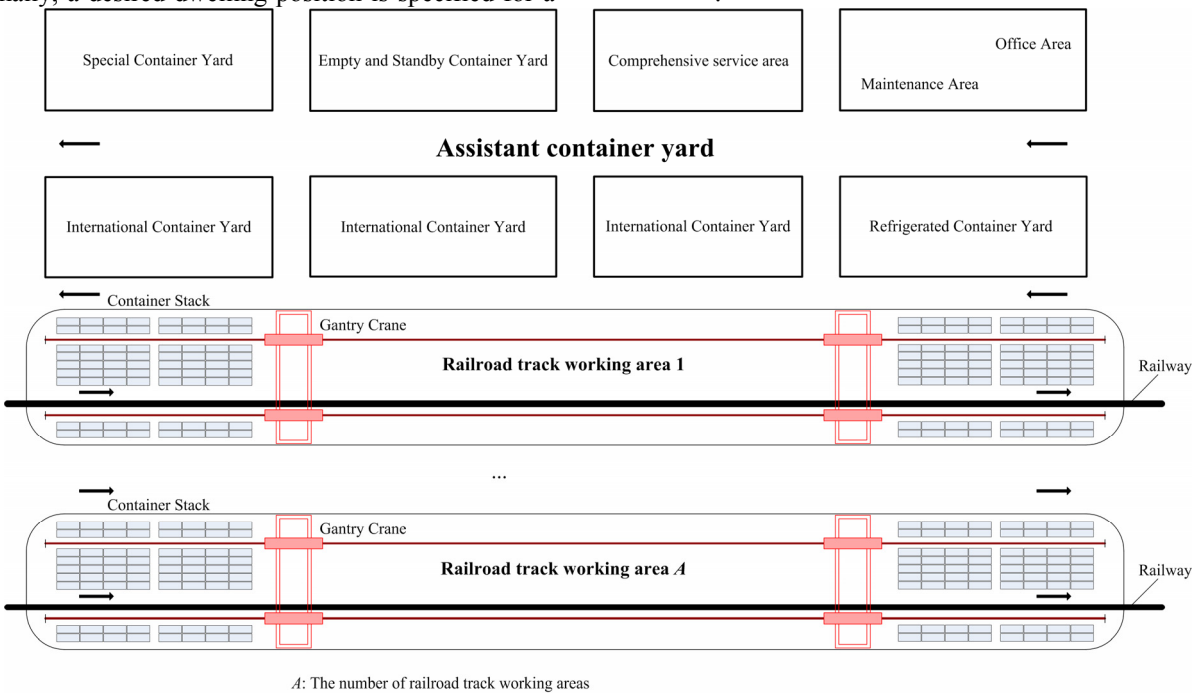

Fig. 1. Schematic Representation of a railway container terminal

After the railroad track working area has been determined, the gantry crane scheduling problem (GCSP) which is similar to the quay crane scheduling problem (QCSP) in seaport terminals arises. In order to make mathematical modeling more convenient, a container train is typically divided longitudinally into different discharging/loading tasks when they share the same attributes: position, size, destination for outbound containers, or origin for inbound containers. Here, the definition of a task refers to the discharging/loading operation of the container groups belonging to the same operating location. GCs that are mounted on two uniform tracks provide the related handling operation in a railroad track working area. Fig. 2 depicts a typical task partition in a container train. Because GCs are tracks mounted, some crane interference constraints are involved in the GCSP. Two types of constraints are largely considered, such as non-crossing constraints and safety constraints. For the non-crossing constraints, all GCs cannot cross each other on the same tracks. For the safety constraints, adjacent GCs must keep a suitable distance at any time. In practice, only one gantry crane can work on a task at any time. In general, the above interference constraints

must be included in the mathematical model so as to make the schedule feasible and rigorous.

Although the described GCSP is similar to the QCSP in port terminals, there are some unique characteristics of GCSP that are different from QCSP. The GCs are equipped with a cantilever on both sides and are arranged alongside the tracks for a parallel processing of container transferring. Once the working area of a freight train is determined, all GCs in this area are served for the train. Moreover, the crane assignment problem may not be considered in railway container terminal. But the choosing of the dwelling position is very important for charging and discharging of trains. All cranes located a given working area only move on two tracks of the area and can not perform handling tasks of other areas. Generally, the number of tracks and GCs directly determine the processing capacity of a railroad track working area.

In this paper, the effect of dwelling position dependent processing times is integrated into the GCSP with the interference constraints, and the corresponding mathematical model is formulated. If the non-crossing requirement and safety constraint are included in QCSP, the problem is NP-complete[2]. Obviously, the proposed problem is also NP-complete hence there exists no polynomial time algorithm for the exact solution of the GCSP. Consequently, heuristic or meta-heuristic algorithms are needed to obtain near optimal solutions. Since there is no detailed work that introduces the use of the artificial bee colony (ABC) algorithm to solve the crane scheduling problem, a novel discrete ABC algorithm is developed for dealing with the integrated GCSP. In the novel algorithm, an analogous operational structure which enables to maintain all major characteristics of ABC is introduced into the classical search equation of ABC algorithm. In addition, a well-designed decoding procedure is employed to transform a solution to a gantry crane schedule. .

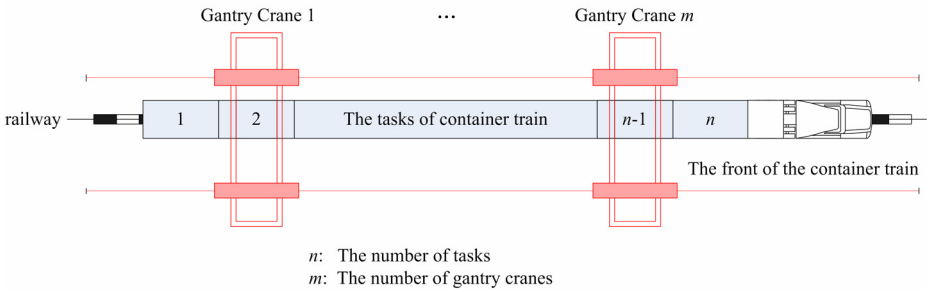

Fig. 2. Typical task partition of a container train

The remainder of this paper is organized as follows: Section 2 surveys the existing literature on the railway container terminal and the quay crane scheduling problem of port container terminal. Section 3 formulates a mathematical model for the considered GCSP. Afterward, a novel discrete ABC algorithm is proposed in Section 4, and the computational experiments in Section 5 investigates the performance of the proposed approach. Some conclusions and future work are given in Section 6.

2. Literature review

There are many different decision problems involved in the operation management of railway container terminals[3], such as the service slots of trains scheduling, railroad track working area assigning, the position of the containers on trains deciding, gantry crane scheduling, and so on. Compared with port container terminals, the researches of the hinterland railway container terminal are relatively less. In order to avoid the complexity of computation, simulation was used for analyzing the performance of new schemes in hinterland railway container terminal[4, 5]. However simulation did not provide any good schedules, mathematical programming is necessary. Boysen et al.[3] considered the freight train scheduling problem of the modern railway container yard, provided a mathematical program and described two different solution procedures. In addition, Kozan[6] designed a network model to improve the efficiency of container transfer operations in multimodal terminals. Corry and Kozan[7] investigated a dynamic assignment problem of load planning for intermodal trains. Jeong and Kim[8] studied an integrated scheduling of rail crane

operations and truck deliveries between a port terminal and a rail terminal. Although the above studies address the different scheduling problem of port-rail terminal, the gantry crane scheduling has never gotten enough attention and research.

There is almost no study on GCSP for railway container terminal, but the gantry crane scheduling problem is similar to the quay crane scheduling problem (QCSP) in port terminals. The QCSP have received great attention in the literature. The literature review of this paper therefore focuses mainly on quay crane scheduling problem with interference constraints. A latest survey of berth allocation and quay crane scheduling problems in port container terminals was presented by Meisel and Bierwirth[9].

Since Daganzo[10] studied the static and dynamic quay crane scheduling problems, the QCSP has drawn a worldwide attention as port container terminal developed rapidly. Daganzo assumed that container vessel can be divided into holds, and only one quay crane can work on a hold at a time. Quay cranes can travel freely and quickly from one hold to another. A branch and bound solution method was designed by Peterkofsky and Daganzo[11] for solving large instances of the static crane scheduling problem. Nevertheless, the two papers did not involve any interference constraints between quay cranes in practical operations. Lim et al.[12] considered that containers from a given area on a vessel were a job, and every job has a profit value when it is serviced by only one crane at any time. They augmented Peterkofsky and Daganzo's study, and provided an integer programming model with non-crossing constraint, neighborhood constraint and job-separation constraint for the objective of maximizing the total profit. Dynamic programming algorithms, a probabilistic tabu search, and a squeaky wheel optimization heuristic were developed to solve the scheduling problem. But a profit value associated with a crane-to-job match is difficult to determine in practice, and hence their research cannot easily be applied in port container terminal. Kim and Park[13] defined a task as a discharging or loading for a cluster of adjacent slots on one container vessel. They formulated a mixed integer programming model, which considers non-crossing constraint related to the operation of quay cranes, and designed a branch and bound method and a greedy randomized adaptive search procedure (GRASP) to obtain the optimal solution.

The above two papers pointed out two main respects of the scheduling problem: QCSP with container groups and QCSP with complete bays. For the QCSP with container groups, Moccia et al.[14] revised the Kim and Park formulation that yielded some solutions where interference between quay cranes is violated, and developed a branch and cut algorithm incorporating several families of valid inequalities adopted from solution methods for the precedence relationships of vehicle routing problem. Sammarraet al.[2] provided a tabu search algorithm based on a local technique for the scheduling problem. Their algorithm provides a good balance between solution quality and computation time, and outperforms the GRASP and the branch and cut algorithm. Afterwards, Ng et al.[15] proposed a scheduling heuristic to find effective schedules for the scheduling problem. Owing to lack of a correct treatment of crane safety constraints, Bierwirth and Meisel[16] presented a revised optimization model for the scheduling of quay cranes and proposed a branch and bound algorithm.

For the QCSP with complete bays, Zhu and Lim[17] studied the crane scheduling problem with non-crossing constraints, and showed that the problem is NP-complete. A simulated annealing algorithm that employed a new graph-search-based neighborhood search was devised to tackle large-sized instances. Besides, Lim et al.[18] developed a different improved simulated annealing algorithm for the m-parallel crane scheduling model with non-crossing constraint. Genetic algorithm was also proposed to obtain near optimal solutions for quay crane scheduling with non-interference constraints by Lee et al.[19]. The handling priority of each ship bay was considered in the quay crane scheduling problem with non-crossing constraint[20, 21]. Furthermore, a unifying rich QCSP model that comprehensively incorporates a variety of practical problem aspects was provided and was solved by a branch and bound method[22].

The aforementioned researches have focused on models with spatial constraints, but recent studies are paying more attention to the integration of the berth allocation problem (BAP) and the quay crane assignment problem (QCAP). The integration of the BAP and the QCAP was originally investigated by Park and Kim[23]. Next, a series of in-depth studies were made by Meisel and Bierwirth[24], Zhang et al.[25], and Raa et al.[26]. They mainly incorporated the berthing position dependent handling times, the coverage ranges of quay cranes,

vessel priorities, preferred berthing locations and handling time considerations in the integrated scheduling model by Park and Kim[23]. Correspondingly, there is less study on the integrated berth allocation and quay crane scheduling problem. Liang et al.[27] introduced a formulation for the simultaneous berth and quay crane scheduling problem, and applied genetic algorithm with heuristic to find an approximate solution. But they did not consider real-life operation constraints of quay cranes. Moreover, Lee and Wang[28] considered the relationship between berth allocation and quay crane scheduling with non-crossing constraints, and proposed a mixed integer programming model including two parts for the integrated scheduling problem. However, they failed to integrate the effect of berthing position dependent processing times into the QCSP with interference constraints.

From this review, it can be said that there are some researches focusing on QCSP. However, to the best of our knowledge, gantry crane scheduling problem with the effect of dwelling position dependent processing times has not been considered for railway container terminals in the existing literature. There are no effective approaches to obtain the optimal solution for the problem under consideration. Therefore, a novel discrete ABC algorithm (DABC) is proposed to solve the problem.

3. Mathematical formulation

In order to minimize the freight container train handling time, a mixed integer programming model for the GCSP is formulated in this section. The following assumptions and constraints are imposed on the GCSP:

- All tasks have different original processing times but the operation rate of cranes is identical;
- No preemption is allowed among all tasks. That is to say, once a gantry crane starts to process a task, it must complete it before another task is processed;
- All gantry cranes move between two adjacent tasks at uniform travel time;
- Gantry cranes located in the same loading area are operated on the same tracks and cannot cross each other. In addition, the cranes line up in tracks and tasks are in trains that stand along the railroad track. These cranes and tasks are labeled according to their relative spatial positions. This means that the cranes 1, 2, 3, …, m are arranged on two parallel tracks from left to right, and tasks 1, 2, 3, …, n are in the similar manner. Fig. 2 shows the details of the layout.
- Adjacent GCs have to keep a safety margin at any time.

The following notations are used to define the problem:

- n The number of tasks;
- m The number of gantry cranes;
- d^0 The desired dwelling position of one single container train;
- d The determined dwelling position of a container train;
- p_i^0 The original processing time of task i ($1 \le i \le n$);
- p_i The actual processing time of task i ($1 \le i \le n$);
- s The necessary safety margin between adjacent GCs;
- r_k The earliest available time of GC k ($1 \le k \le m$);
- l_i The location of task i that is expressed by the task number;
- l_k^0 The initial position of GC k that is expressed by the task number;
- t^0 The travel time of a gantry crane between any two adjacent tasks;
- t_{ij} The travel time of a gantry crane from position l_i to position l_j ($1 \le i, j \le n$). t_{0i}^k represents the travel time from the starting position l_k^0 of GC k to location l_i of task i;
- M a sufficiently large positive number.

The decision variables are defined as follows:

- C_i integer, the completion time of task i ($1 \le i \le n$);
- $C_{\max}$ integer, container train handling time, that is, the maximum completion time among all tasks;
- x_{ik} binary, set to 1 if task i is assigned to crane k; 0, otherwise ($1 \le i \le n$, $1 \le k \le m$);
- y_{ij} binary, set to 1 if task i completes no later than task j starts; 0, otherwise ($1 \le i, j \le n$).

The mathematical formulation is presented below, followed by a brief explanation.

Minimize: $C_{\max}$ (1)

subject to:

$$C_{\max} \ge C_i, \qquad \forall 1 \le i \le n \tag{2}$$

$$p_i = p_i^0 \cdot (1 + \beta \left| d - d^0 \right|), \qquad \forall 1 \le i \le n \tag{3}$$

$$\sum_{k=1}^{m} x_{ik} = 1, \qquad \forall 1 \le i \le n \tag{4}$$

$$t_{ij} = t^0 \cdot \left| l_i - l_j \right|, \qquad \forall 1 \le i, j \le n \tag{5}$$

$$\sum_{k=1}^{m} x_{ik} \cdot (r_k + t_{0i}^k) + p_i \le C_i, \qquad \forall 1 \le i \le n \tag{6}$$

$$C_i - (C_j - p_j - t_{ij}) + y_{ij} \cdot M \ge 0, \qquad \forall 1 \le i, j \le n \tag{7}$$

$$C_i - (C_j - p_j - t_{ij}) + (1 - y_{ij}) \cdot M \le 0, \quad \forall 1 \le i, j \le n \tag{8}$$

$$M \cdot (y_{ij} + y_{ji}) \ge \sum_{k=1}^{m} k \cdot x_{ik} - \sum_{l=1}^{m} l \cdot x_{jl} + 1, \quad \forall 1 \le i, j \le n \tag{9}$$

$$M \cdot (y_{ij} + y_{ji}) \ge i + s + 1 - j, \quad \forall 1 \le i, j \le n \tag{10}$$

$$x_{ik}, y_{ij} \in \{0, 1\}, \quad \forall 1 \le i, j \le n; 1 \le k \le m \tag{11}$$

In the above formulation, objective function (1) minimizes the train handling time, which is determined in Constraints (2) by the maximal completion time among all tasks. Constraints (3) redefine the processing time of task i that considers the effect of the dwelling position dependent processing times. Constraints (4) ensure that every task must be assigned exactly to one crane. Constraints (5) calculate the travel time of a gantry crane between position l_i and l_j. Constraints (6) make sure that a task is not started earlier than the earliest available time of the assigned crane plus the time needed by the crane moving from its initial position to the location of the task. Constraints (7) and (8) define the property of decision variable y_{ij}: Constraints (7) indicate that y_{ij}=0 if $C_i \ge C_j - p_j - t_{ij}$, which means $y_{ij} = 0$ when task i finishes after task j starts; Constraints (8) indicate that y_{ij}=1 if $C_i \le C_j - p_j - t_{ij}$, which means $y_{ij} = 1$ when task i finishes no later than task j starts. Constraints (9) impose non-crossing constraint between gantry cranes located in the same tracks. Suppose that task i and task j are performed simultaneously and $i < j$, which means $y_{ij}+y_{ji} =0$. As both gantry cranes and tasks are arranged in ascending order from the front to the tail of the container train. Thus, if task i is performed by crane k and the task j is performed by crane l, then $k+1 \le l$, cf. the paper of Lee et al.[19]. Constraints (10) guarantee that adjacent cranes have to keep a safety distance at any time when the cranes perform tasks simultaneously. Assume that tasks i and j are processed simultaneously by two cranes, then $y_{ij}+y_{ji} =0$. The necessary difference between tasks i and j must be no less than the required safety margin s. Since the tasks are labeled according to their relative spatial positions, the inequation $i+s+1 \le j$ must be met.

The above descried GCSP is NP-complete. Because if the proposed problem is restricted such that the effect of berthing position dependent processing times is ignored and the travel time of a gantry crane between any two tasks is not considered, the resulting restricted GCSP is identified with the QCSP with safety distance and non-crossing constraint to a certain extent. The quay crane scheduling problem has been proven to be NP-complete by Sammarra et al.[2]. Therefore, the proposed GCSP is NP-complete, and the next section employs a novel discrete artificial bee colony algorithm to obtain near optimal solutions.

4. The discrete artificial bee colony algorithm

The artificial bee colony (ABC) algorithm introduced by Karaboga[29-32] is a new swarm optimization meta-heuristic for continuous function optimization based on the intelligent foraging behavior of honey bee swarm. In the ABC algorithm, the colony of artificial bees consists of three groups of bees namely employed bees, onlooker bees and scout bees searching for food. The first half of the colony consists of the employed bees, and the second half includes the onlooker bees. The position of a food source represents a possible solution of the problem under consideration and the nectar amount of a food source corresponds to the quality of the associated solution (fitness value). At the initialization stages, a randomly distributed population of solutions is filled with SN number of generated D-dimensional real value vectors, where SN denotes the colony size and D denotes the number of optimization variables. After initialization, the population of the solutions is subjected to repeated cycles of the search process of the employed bees, onlooker bees, and scout bees. An employed bee carries out a local search on the position in its memory to find a new food source (solution) and evaluates the nectar amount (fitness value) of the new food source. If the nectar amount of the new food source is higher than that of the old one, then the bee memorizes the new position and forgets the previous one. After the employed bees complete their search process, they share the nectar information of the food source and their position information with the onlooker bee in the dance area. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source depending on a probability related to its fitness value, which is calculated by Eq. (12).

$$prob_i = fitness_i / \sum_{i=1}^{SN} fitness_i, \tag{12}$$

Where $fitness_i$ is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i. Subsequently, an onlooker bee also produces a candidate solution and applies the greedy

selection mechanism to select a better one as the new food source between old and new food sources. In ABC algorithm, the employed bee or onlooker bee uses Eq. (13) to produce a candidate solution from the old one.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{13}$$

where j is a random integer in the range [1, D] and $k \in (1, 2, \ldots, SN)$ is a randomly chosen index that is different from i. ϕ_{ij} is a uniformly distributed real random number in the range [−1, 1]. It controls the production of a neighbor food source position around x_{ij} and the modification represents the comparison of the neighbor food position visually by the bee. If a food source cannot be further improved through the predetermined number of trails "*limit*", then the food source is assumed to be abandoned and replaced with a new food source by a scout. If the abandoned source is $x_{ij}, j \in (1, 2, \ldots, D)$ then the scout discovers a new food source x_{ij} using Eq. (14).

$$x_{ij} = x_j^{\min} + rand(0,1) \cdot (x_j^{\max} - x_j^{\min}), \tag{14}$$

where $x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bounds for the dimension j respectively.

The above standard ABC algorithm was initially developed for continuous optimization problems and cannot be suitable for the discrete scheduling problem. Although, some researchers have attempted to improve the standard ABC algorithm for solving some typical combinational optimization problems[33-36], these improved algorithms do not reserve the classical search characteristics. Therefore, in this paper, the search equation of generating a new solution, that is similar to a discrete particle swarm optimization algorithm for scheduling optimization problem[37, 38], is redefined based on integral number encoding scheme in this paper. In addition, the information of the found best food source is embedded in the local search procedure for accelerating the convergence rate of the proposed algorithm. The details of the proposed algorithm designed for the GCSP are elaborated as follows:

4.1. *Representation of food sources and initialization of the population*

In order to solve the GCSP by the DABC algorithm, the first step is to represent a solution of a problem as a food source. Generally, the most known encoding scheme adapted for scheduling problem is a permutation of all tasks as a food source. The order of the tasks in the permutation denotes the handling order of the tasks by the GCs. Fig. 3 shows the encoding scheme of a sequence of 8 tasks.

It is common to randomly produce the initial population in swarm intelligent optimization algorithms. Nevertheless, this method does not obtain a good population. In this paper, an initialization procedure based on sorting order is proposed. Firstly, a population of 2×*SN* food sources is formed, where each food source is a sequence of n tasks generated randomly. Then, all food sources are evaluated and sorted according to the ascending order of their objective function value .The first *SN* food sources are selected as the initial population.

Position of Food Source: Task Number 1-8

Processing Time	10	60	10	20	40	40	30	25
Food Source	1	6	2	3	4	7	5	8
Operation Sequence	1	2	3	4	5	6	7	8

Fig. 3. Illustration of the food source representation

4.2. *Construction of crane schedule and evaluation of fitness value*

Once a sequence of all tasks is given, a GC schedule can be obtained by assigning tasks to GCs using the decoding procedure in Fig. 4. For each sequence of tasks represented by the food source, two GCs are added at the dummy location 0 and the other dummy location n+1 for ensuring the operation position of every task located between any two GCs, respectively. The earliest available times of the additional cranes are set to infinity, and the others are equal to their individual starting times. The two available GCs are chosen based on the current location of each GC. The unassigned task in the food source is assigned to the gantry crane, which is selected based on the comparison of the earliest available time of the two available GCs, the distance between this task and these two available GCs, or the number of the GCs.

The gantry crane schedule obtained from the proposed decoding scheme does not violate the non-crossing constraints (9), but it may violate the safety constraints (10). Therefore, every gantry crane schedule must be checked whether it satisfies the safety distance as follows. Based on the gantry crane schedule obtained

from the proposed procedure, the completion time of all tasks is determined by mathematical calculation. According to constraints (7) and (8), y_{ij} ($\forall 1 \le i, j \le n, i \ne j$) can be obtained and then the gantry crane schedule will be checked whether it ensures a given safety distance between two adjacent cranes. If it satisfies constraints (10), the fitness value of its corresponding solution is set to the reciprocal of its objective value, as shown in Eq.(15); otherwise, the fitness value of its corresponding solution is set to zero.

$$fitness = 1/C_{\max} \tag{15}$$

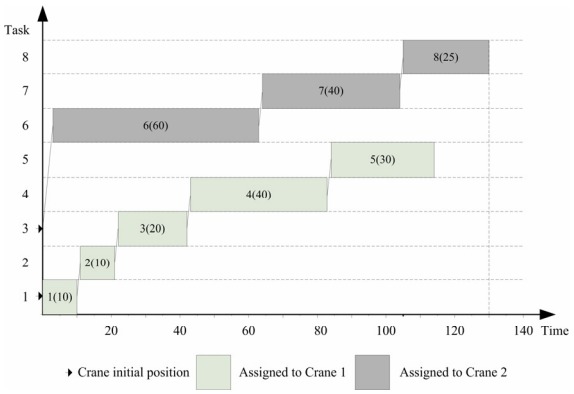

Fig. 5. The feasible schedule based on the proposed decoding procedure

Procedure: transform a sequence of tasks to a gantry crane schedule
Inputs: n, m, x_0, $ptime_i$, $crane\_location_k^0$, r_k, t^0 // x_0 is a given sequence of tasks, $ptime_i$ is the actual processing time of task i ($1 \le i \le n$), $crane\_location_k^0$ is the initial position of gantry crane k, r_k is the earliest available time of gantry crane k ($1 \le k \le m$) and t^0 is the travel time of a gantry crane between any two adjacent tasks //
Output: $C_{\max}$ // $C_{\max}$ is the container train handling time determined by the maximum completion time among all tasks in the sequence x_0 //
begin
let $crane\_location \leftarrow (0, crane\_location_1^0, crane\_location_2^0, \ldots, crane\_location_m^0, n+1)$ //add two GCs, their positions locate at the dummy location 0 and the other dummy location n+1, respectively //
let $crane\_r \leftarrow (\text{infinity}, r_1, r_2, \ldots, r_m, \text{infinity})$ // the earliest available times of two dummy GCs are set to infinity //
set $crane\_c_k = 0$, for $1 \le k \le m+2$ // initialize the completion time $crane\_c_k$ of gantry crane k //
set $task\_time_i = 0$, for $1 \le i \le n$ // initialize the completion time $task\_time_i$ of task i //
$i \leftarrow 1$
while ($i \le n$) **do**
for (j=1; $j \le m$+1; j++) **do**
if $x_0(i) \ge crane\_location(j)$ and $x_0(i) \le crane\_location(j+1)$
if $crane\_r(j) < crane\_r(j+1)$
task $x_0(i)$ is assigned to crane j
elseif $crane\_r(j) > crane\_r(j+1)$
task $x_0(i)$ is assigned to crane j+1
else
if $|crane\_location(j) - x_0(i)| < |crane\_location(j+1) - x_0(i)|$
task $x_0(i)$ is assigned to crane j
elseif $|crane\_location(j) - x_0(i)| > |crane\_location(j+1) - x_0(i)|$
task $x_0(i)$ is assigned to crane j+1
else
task $x_0(i)$ is assigned to crane j
end if
end if
assume that task $x_0(i)$ is assigned to crane k // $k = j$ or $j+1$ //
$min\_num = \min\{l(k), x_0(i)\}$
$max\_num = \max\{l(k), x_0(i)\}$
$crane\_c(k) = \max\{task\_time(min\_num), \ldots, task\_time(max\_num)\} + ptime(i) + t^0 \times |crane\_location(j) - x_0(i)|$ // update the completion time of crane k //
$crane\_location(k) = x_0(i)$ // update the position of crane k //
$crane\_r(k) = crane\_c(k)$ // update the available time of crane k //
$task\_time(x_0(i)) = crane\_c(k)$ // update the completion time of task $x_0(i)$ //
break // terminate the execution of **for** loop //
end if
end for
$i = i + 1$
end while
$C_{\max} = \max\{task\_time(x_0(1)), task\_time(x_0(2)), \ldots, task\_time(x_0(n))\}$ // calculate the container train handling time //
end

Fig. 4. The proposed decoding procedure

Furthermore, an illustrative example is provided according to the given task sequence in Fig. 3. There are two cranes and 8 tasks. The initial positions of gantry crane 1 and 2 are on task 1 and task 3, respectively. The initial earliest available times of all gantry cranes are 0. A feasible schedule is constructed based on the above procedure. As shown in Fig. 5, the maximum completion time among all tasks calculated by the procedure is 130.

4.3. *Employed bee phase*

The employed bees generate new candidate solutions in the neighborhood of their current positions according to the local search process. However, the convergence rate of the standard ABC algorithm is poor. Inspired by particle swarm optimization (PSO)[39], the found best solution is incorporated into the local search equation for accelerating the convergence of ABC algorithm. The process of producing a new solution is modified by the following equations for a given solution x_i:

if *rand*(0, 1) < 0.5

$$v_i = x_i + (\phi(c_1) \times (x_i - x_k)), \quad (16.a)$$

else if *rand*(0, 1) > 0.5

$$v_i = x_i + (\phi(1 - c_1) \times (x_i - x_g)), \quad (16.b)$$

where $\phi(c_1)$ and $\phi(1-c_1)$ are 1-by-n arrays consists of 0 or 1 elements, that are produced by a Bernoulli distribution in which the mathematic expectation of getting 1 is c_1 and $1-c_1$, respectively. x_k is the other solution selected randomly from the population that is different from x_i, and x_g is the global best solution found so far. The redefinitions of the operators, used in the Eq. (16) are depicted as follows.

4.3.1. *The subtract operator* (−)

The subtract operator is identified with a 2-point order crossover operator of genetic algorithm. According to the crossover, the partial information of the neighboring solution or the found best solution is reserved. For crossover operator, a substring is selected from the second solution randomly, and a new chain is produced by copying the substring into its corresponding positions. These tasks which are not already in the substring from the first solution are selected, and placed into the unfixed positions of the new chain from left to right. The new chain is then the result of subtract operator. Fig. 6 that presents an example of producing a new chain illustrates the subtract operator.

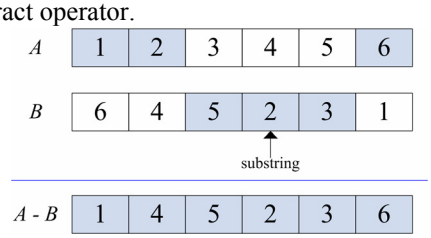

Fig. 6. Illustration of the redefined subtract operator (−)

4.3.2 *The multiply operator* (×)

By this operator, the exploration ability of DABC algorithm can be improved. The random binary vector is generated using Bernoulli distribution, and is multiplied by a task sequence produced by the redefined subtract operator. Carrying out the multiplication process, the solution space is explored by the neighboring solution or the found best solution. The multiply operator is equivalent to Hadamard product. The Hadamard product of two 1-by-n arrays A and B is defined by $(A \cdot B)_i = a_i b_i$. The redefined operation rule is illustrated in Fig. 7.

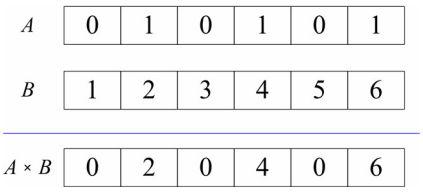

Fig. 7. Illustration of the redefined multiply operator (×)

4.3.3 *The plus operator* (+)

The plus operator is also similar to the crossover operator of genetic algorithm, but it only interchanges the nonzero elements of the second vector with these corresponding elements of the first vector. Fig. 8 illustrates an example of the plus operator. Using this operator, the new solution can absorb the good segment from these chains of the multiply operator at a certain probability.

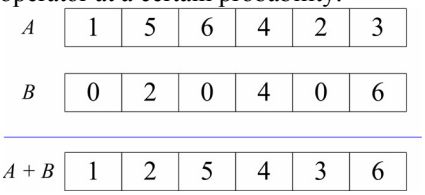

Fig. 8. Illustration of the redefined plus operator (+)

From these operators, it can be seen that the modified equations reserve all major search

characteristics of the ABC algorithm shown in solving the continuous optimization problems. After finishing local search processes, the employed bee obtains a new solution. Subsequently, the new solution will be evaluated and compared with the old one. The suitable solution will be retained in the population according to the following selection procedure.

if $fitness(v_i) \geq fitness(x_i)$ or $rand(0, 1) < c \times (1 - iter)/iter_{\max}$
  solution v_i replaces the old solution x_i
else
  solution x_i is retained in the population
end

where c is the given value that ranges from 0 to 1. *iter* represents the current iteration times, and $iter_{\max}$ is the maximum iteration times.

4.4. *Onlooker bee phase*

In the standard ABC algorithm, each onlooker bee selects a solution based on its probability value associated with the food source. However, the selection approach consumes more computational time to obtain these promising solutions. A tournament selection with size of three is proposed in the algorithm. In the tournament selection strategy, three food sources are picked randomly from the population, and then the solution with highest fitness value will be chosen by the onlooker bee. Afterwards, each onlooker bee also implements the same local search operation with the employed bee for updating the food source. The suitable solution between the old one and the new one will be kept in the population by using the same selection procedure mentioned in the employed bee phase.

4.5. *Scout bee phase*

If a particular solution cannot be improved through the predetermined number of trails, a scout bee regenerates a food source randomly in the predefined solution space. Using the random search process can increase the population diversity, but this will lower the search efficiency. After the employed bee phase and the onlooker bee phase, the current best solution of the whole population has found. The best solution often takes better information of food source than others during the optimization process. Therefore, in the DABC algorithm, the scout bee firstly generates a solution randomly, and then carries out the local search operation with using the found best solution, as shown in Eq. (17).

$$x_{new} = x_{rand} + (\phi(c_2) \times (x_{rand} - x_g)), \quad (17)$$

where $\phi(c_2)$ is 1-by-n arrays consist of 0 or 1 elements, that are also generated by a Bernoulli distribution where the mathematic expectation of getting 1 is c_2. x_{rand} is a randomly generated solution.

4.6. *Parameter tuning of the algorithm*

The appropriate Parameters are critical to the DABC performance. DABC has three key parameters: c_1, c_2 and c. In this paper, the parameter selection method of Ruiz and Stutzle[40] which consists of design of experiments (DOE) and multi-factor analysis of variance (ANOVA) is used to tune the parameters of the proposed algorithm. First of all, the specified levels of all parameters were listed, as follows:

- c_1: seven levels (0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9)
- c_2: four levels (0.1, 0.2, 0.3 and 0.4)
- c: four levels (0.6, 0.7, 0.8 and 0.9)

Table 1. ANOVA table for the experiment on tuning the parameters of DABC

Source	Sum of Squares	*Df*	Mean Square	*F*-Ratio	*p*-Value
Main effects					
c_1	3.379	6	0.563	25.318	0.000
c_2	1.430	3	0.477	21.430	0.000
c	0.011	3	0.004	0.161	0.922
Interactions					
$c_1 * c_2$	0.940	18	0.052	2.347	0.008
$c_1 * c$	0.688	18	0.038	1.720	0.064
$c_2 * c$	0.420	9	0.047	2.097	0.046
Error	1.201	54	0.022		
Total	315.409	112			
Corrected Total	8.069	111			

The above list yields a total of 7×4×4=112 different combinations when one carried out for a full factorial experimental design. The algorithm is tested with a set of randomly generated problem instances. More specifically, 9 different combinations of n and m, with $n \in \{20, 30, 40\}$ and $m \in \{2, 3, 4\}$, and the original processing times are distributed uniformly in the interval (30, 180). For every combination of n and m, the initial locations of cranes are varied in [1, n]. The 9 instances are tackled by 112 different tests with a limited iterations time fixed to $6 \times n \times m$. The response variable is calculated by the following equation:

Relative Percentage Deviation

$$(\text{RPD}) = 100 \times (Alg_{sol} - Best_{sol}) / Best_{sol}, \quad (18)$$

where Alg_{sol} is the objective function value obtained by a combination of factors for a given instance and $Best_{sol}$ is the best solution yielded by all combinations of factors for the same instance.

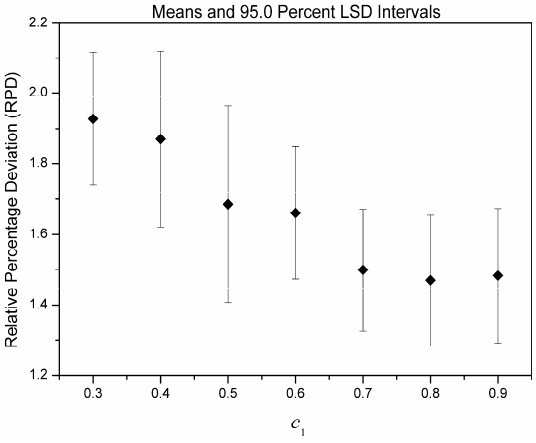

Fig. 9. Means plot and LSD intervals for parameter c_1

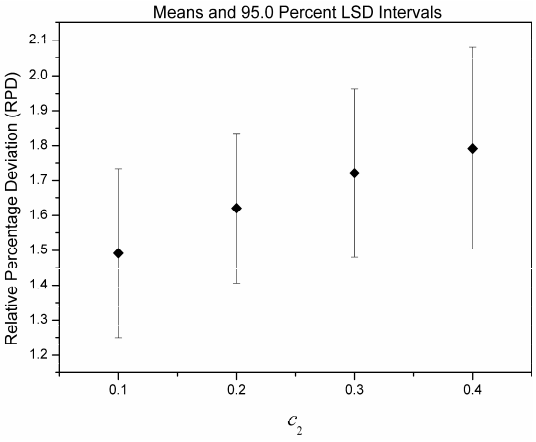

Fig. 10. Means plot and LSD intervals for parameter c_2

The experiment was analyzed by means of ANOVA. In order to apply ANOVA, there is the need to check the three main hypotheses of ANOVA. The ANOVA results are shown in Table 1. The analysis indicates that parameters c_1 and c_2 are very significant. However, the different levels of c do not yield statistically significant in the response variable RPD. This suggests that the proposed DABC algorithm is rather robust with respect to the reserve strategy of new solutions in the employed bee phase. But, the interaction between c_2 and c is statistically significant due to its low p-value. This indicates that the combination of the two parameters is still critical for the performance of DABC. As the only parameter of scout bee phase, c_2 directly influences the population diversity, especially in the evolutionary latter stages. The larger the factor c, the better the population diversity. Maintaining the population diversity is very important to the performance of the algorithm. Therefore, c should not be ignored. Thus, all three parameters are further analyzed by multi-compare method. The means plot with Least Significant Difference (LSD) intervals at 95% confidence level for the three parameters are shown in Fig. 9-11. From the Fig. 9, it seems that a setting of c_1=0.8 gives a best RPD, although it is not statistically significantly different from 0.7 or 0.9 at a 95% confidence level. As shown in Fig. 10, a setting of c_2=0.1 provide better results among all levels. Though different levels of c does not yield statistically significant different in the result, a setting of 0.7 gives a better RPD than the other three levels. The means plot for the parameter c shown in Fig. 11.

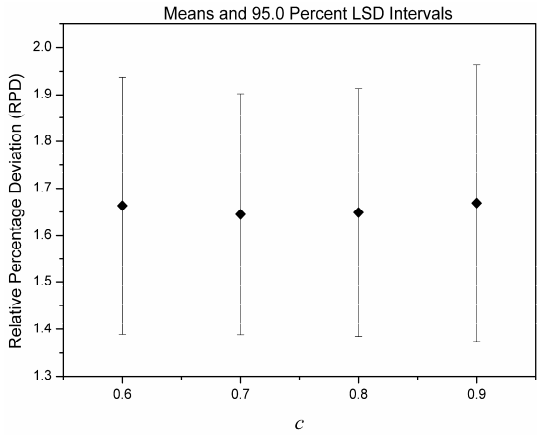

Fig. 11. Means plot and LSD intervals for parameter c

As a result from the experimental analysis, the best parameters are set as follows: c_1= 0.8, c_2= 0.1 and c= 0.7.

5. Computational experiments

In this section, the performance of the developed DABC is evaluated by three different sets of problems. The generation of these instances is described in the next subsection. Afterward, computational experiments are conducted to make a comparison with CPLEX. Owing to the existence of some similarity between GCSP and QCSP, genetic algorithm (GA) proposed by Lee et al.[19] is also employed as a comparison. After reporting the results, the comprehensive statistical analyses are carried out to test the significance of the reported results.

In order to assess the performance of solutions delivered by DABC, attempts are made to solve the small-sized instances using CPLEX 12.4 software which solves mixed integer programming problems based on the branch and cut algorithm. Since the problem under study is NP-complete, it is impossible to obtain optimal solutions by some polynomial time algorithms. For the medium-sized and large-sized instances, the solutions given by CPLEX with a time limit of 10 minutes are compared with the results from GA and DABC. To evaluate the quality of DABC solutions, a lower bound is calculated using CPLEX by ignoring the interference constraints and the travel time. For a particular instance, let $C_{\max}(Alg)$ denote the train handling time for the solution obtained by a given algorithm. Then the effectiveness of the corresponding solution approach can be measured by the following equation:

$$GAP=100\%\times(C_{\max}(Alg)-LB)/LB \quad (19)$$

Clearly, lower values of GAP are preferable.

CPLEX is deterministic and only one run is necessary. However, the GA and DABC are stochastic and some replicates are needed to run for better evaluate the results. The two algorithms are run for five times on each instance, with the following setting: the maximum iteration times $iter_{\max}=6\times n\times m$, the population size $PS=3\times n$. For GA, the crossover rate P_c and the mutation rate P_m are set to 0.25 and 0.2. For DABC, the predetermined number of no improvements *limit* is set to $2\times n$. The listed objective value and computational time are the mean of the five reported results.

The algorithms are implemented in MATLAB. All tests are completed on a Personal Computer including a Pentium Dual-Core 2.6 GHz Processor and 2GB RAM.

5.1. *Benchmark of instances*

Three sets of instances are generated randomly, representing freight trains of small, medium and large size, respectively. For the small-sized instances, the original processing times of all tasks are randomly obtained from an integer uniform distribution on $U(20, 150)$. For the medium-sized and the large-sized instances, their original processing times are randomly picked from a uniform distribution of $U(30, 180)$. For the three sets of instances, the initial locations of gantry cranes are varied in $[1, n]$. The dwelling position dependent factor β is set to 0.2. The safety margin s is set to 1 and the travel time of crane t^0 is set to one time unit. Then, all combinations of $n=\{6, 7, 8, 9, 10, 11, 12\}$ and $m=\{2, 3\}$ as the small-sized problems are tested. For medium-sized problems, the number of tasks is chosen from {15, 16, 17, 18, 19, 20} and the number of cranes is chosen from {2, 3, 4}. For large-sized problems, the combinations of $n=\{30, 40, 50, 60, 70\}$ and $m=\{3, 4, 5\}$ are considered. The data set consists of 47 instances as outlined in Table 2.

Table 2. Parameters for Instance Generation

Set	Number of instances	n	m	Original processing times of tasks	Initial locations of cranes
Small-sized instances	14	6, 7, 8, 9, 10, 11, 12	2, 3	$U(20, 150)$	$[1, n]$
Medium-sized instances	18	15, 16, 17, 18, 19, 20	2, 3, 4	$U(30, 180)$	$[1, n]$
Large-sized instances	15	30, 40, 50, 60, 70	3, 4, 5	$U(30, 180)$	$[1, n]$

5.2. *Experimental results*

Table 3 reports computational results achieved for the small-sized instances. These instances are solved optimally by CPLEX software in a reasonable time. When the number of the tasks is less than 10, CPLEX consumes very short time in solving the GCSP. Once the number of the tasks exceeds 10, the computational times of CPLEX increase rapidly as the instance becomes larger. Compared with the instances with two cranes, CPLEX needs shorter time to solve the instances with three cranes. It is largely because these instances with three cranes produce fewer nodes in the calculation process. Moreover, the proposed DABC algorithm can

obtain the optimal solution in a very short time. None of the instances requires more than 2 seconds to be solved. However, only eight out of the 14 instances are solved to optimality by GA. Nevertheless, the runtime of GA is less than DABC.

Considering the results of medium-sized problems, Table 4 summarizes the LB, the objective value, the average computational time and GAP for each solution method. From Table 4, it is found that CPLEX hardly terminates within the runtime limit for medium-sized instances. As expected, the average runtime grows significantly when turning to instances of medium size. The runtime limit is completely exhausted for most instances in the set leading to an average runtime of 535.85s. Interestingly, three instances with four cranes among 18 instances are solved optimally by CPLEX. This confirms that the instances with more cranes are relatively easy to solve by CPLEX. In addition, DABC and GA give some competitive solutions. The average GAP observed for the solutions of DABC is merely 3.51% that is just 0.3% weaker than the result achieved by CPLEX. Especially the DABC gives a best known solution for seven out of the 18 medium-sized instances. However, the average GAP given by the GA is as many as 5.08%. DABC strikingly outperforms GA. Since the travel times between any two cranes are ignored in *LB*, The GAP given by all three methods fails to get value 0. The average runtime of DABC which is longer than GA is 5.51s. Furthermore, the performance of DABC is affected by the number of cranes, as shown in Fig. 12. DABC delivers solutions of very good quality for instances with m=2 cranes. For m=4, noticeable gaps of up to 11.02% are observed. It is chiefly because when the number of cranes is more, the interference constraints between cranes are easier to be met during the searching process of the DABC algorithm. The algorithm needs more runtime to avoid falling into local optimal solution. Fortunately, the relationship between the algorithm and the number of cranes is no longer significant with the number of tasks increasing.

Table 3. Results of random instances with small sizes

Instance no.	*Size(n×m)*	*CPLEX*		*GA*		*DABC*	
		Value	*CPU(s)*	*Value*	*CPU(s)*	*Value*	*CPU(s)*
1	6×2	**257**	0.05	259	0.06	**257**	0.23
2	6×3	**240**	0.06	**240**	0.08	**240**	0.31
3	7×2	**332**	0.05	**332**	0.08	**332**	0.31
4	7×3	**245**	0.03	**245**	0.13	**245**	0.46
5	8×2	**360**	0.47	**360**	0.12	**360**	0.43
6	8×3	**287**	0.11	**287**	0.18	**287**	0.65
7	9×2	**420**	2.63	421	0.17	**420**	0.58
8	9×3	**269**	0.14	277	0.25	**269**	0.86
9	10×2	**427**	5.08	428	0.23	**427**	0.76
10	10×3	**323**	0.14	**323**	0.35	**323**	1.14
11	11×2	**532**	8.86	534	0.30	**532**	0.98
12	11×3	**326**	1.20	**326**	0.46	**326**	1.46
13	12×2	**645**	240.05	647	0.38	**645**	1.22
14	12×3	**370**	2.16	**370**	0.59	**370**	1.87
Average		359.50	18.65	360.64	0.24	359.50	0.80

Table 5 shows computational results of large-sized instances. Owing to the intractability of the GCSP, CPLEX usually cannot give an optimal solution in a reasonable time for large-sized instances. In particular, CPLEX fails to generate any solution within 10 minutes for experiments 10 to 15. As observed in Table 5, DABC clearly outperforms GA and CPLEX within the limited runtime. There is a remaining average GAP of 4.70%

which is only slightly worse solution quality compared to the medium-sized instances. But the runtime of the proposed algorithm is more than the runtime of GA. The average runtime of DABC is 232.29s and it is acceptable in a practical application.

Table 4. Results of random instances with medium sizes

Instance no.	*Size*($n\times m$)	*LB*	*CPLEX*			*GA*			*DABC*		
			Value	*GAP*(%)	*CPU*(*s*)	*Value*	*GAP*(%)	*CPU*(*s*)	*Value*	*GAP*(%)	*CPU*(*s*)
1	15×2	711	723	1.69	600.00	732	2.95	0.75	724	1.83	2.33
2	15×3	518	538	3.86	600.00	544	5.02	1.14	538	3.86	3.46
3	15×4	477	516*	8.18	222.34	536	12.37	1.53	516	8.18	4.65
4	16×2	972	992	2.06	600.00	994	2.26	0.91	988	1.65	2.71
5	16×3	535	551	2.99	600.00	552	3.18	1.39	552	3.18	4.13
6	16×4	381	406*	6.56	30.73	426	11.81	1.86	423	11.02	5.56
7	17×2	807	819	1.49	600.00	832	3.10	1.09	819	1.49	3.21
8	17×3	536	549	2.43	600.00	565	5.41	1.67	552	2.99	4.88
9	17×4	451	469*	3.99	392.31	483	7.10	2.25	473	4.88	6.58
10	18×2	1017	1038	2.06	600.00	1046	2.85	1.30	1034	1.67	3.84
11	18×3	571	585	2.45	600.00	601	5.25	1.99	590	3.33	5.82
12	18×4	477	488	2.31	600.00	507	6.29	2.68	499	4.61	7.79
13	19×2	978	1002	2.45	600.00	1003	2.56	1.53	991	1.33	4.44
14	19×3	665	685	3.01	600.00	691	3.91	2.33	681	2.41	6.74
15	19×4	524	542	3.44	600.00	552	5.34	3.17	544	3.82	9.17
16	20×2	1016	1046	2.95	600.00	1054	3.74	1.79	1036	1.97	5.20
17	20×3	699	716	2.43	600.00	725	3.72	2.75	715	2.29	7.93
18	20×4	530	548	3.40	600.00	554	4.53	3.72	544	2.64	10.67
Average		659.17	678.50	3.21	535.85	688.72	5.08	1.88	678.83	3.51	5.51

* the optimal solution given by CPLEX

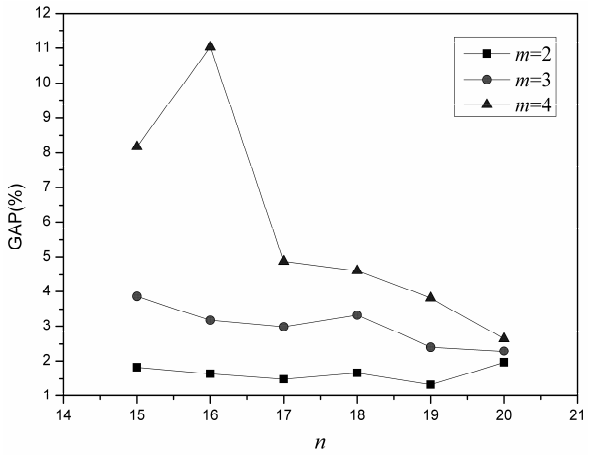

Fig. 12. Comparison of GAP between different numbers of cranes

According to the computational experiments with small, medium and large sizes, the proposed DABC has been well tested to be a competitive algorithm for solving the practical gantry crane scheduling problem in railway container terminals.

The experimental results have demonstrated some significant features of DABC when it is applied to GCSP, as follows.

(1) The performance of DABC algorithm is affected by the number of cranes, as shown in Fig. 12. The effect of the number of cranes on the solution quality achieved by DABC is no longer apparent with the increasing of the number of tasks.

(2) Since the fitness value of the solution that violates the safety constrain is set to zero, the excellent searching

performance of DABC method may be limited. Compared with GA, the proposed method still gives the better solutions for the problem under study.

(3) The neighborhood search method is used to generate the new solution in the employed bee phase and the onlooker bee phase. So the computational time of DABC is longer than GA when the two algorithms have the same population size and the same iteration times.

Table 5. Results of random instances with large sizes

Instance no.	*Size(n×m)*	*LB*	*CPLEX*			*GA*			*DABC*		
			Value	*GAP*(%)	*CPU(s)*	*Value*	*GAP*(%)	*CPU(s)*	*Value*	*GAP*(%)	*CPU(s)*
1	30×3	970	1042	7.42	600.00	1015	4.64	9.76	991	2.16	26.75
2	30×4	775	819	5.68	600.00	809	4.39	13.24	801	3.35	35.88
3	30×5	654	718	9.79	600.00	701	7.19	16.89	672	2.75	45.34
4	40×3	1350	1463	8.37	600.00	1429	5.85	24.86	1394	3.26	64.43
5	40×4	1042	1144	9.79	600.00	1095	5.09	33.64	1078	3.45	87.23
6	40×5	876	952	8.68	600.00	922	5.25	42.86	916	4.57	110.38
7	50×3	1801	2217	23.10	600.00	1935	7.44	52.09	1874	4.05	131.45
8	50×4	1300	1608	23.69	600.00	1384	6.46	71.12	1358	4.46	175.09
9	50×5	1018	1201	17.98	600.00	1092	7.27	88.98	1069	5.01	223.68
10	60×3	2016	N/A*	N/A	600.00	2189	8.58	96.08	2142	6.25	239.88
11	60×4	1435	N/A	N/A	600.00	1554	8.29	130.01	1496	4.25	322.71
12	60×5	1268	N/A	N/A	600.00	1353	6.70	164.38	1340	5.68	407.37
13	70×3	2404	N/A	N/A	600.00	2645	10.02	162.97	2577	7.20	399.12
14	70×4	1590	N/A	N/A	600.00	1738	9.31	219.29	1712	7.67	536.10
15	70×5	1449	N/A	N/A	600.00	1564	7.94	277.26	1542	6.42	676.67
Average		1329.87	1240.44	12.72	600.00	1428.33	6.96	93.56	1397.47	4.70	232.14

*CPLEX fails to generate any feasible solutions within 10 minutes

6. Conclusion

The paper contributes to recent research in a railway container terminal operations schedule by providing a mathematical model for the gantry crane scheduling problem (GCSP) based on the quay crane scheduling problem in seaport container terminals. The effect of dwelling position dependent processing times is incorporated into the GCSP. In addition, the non-crossing requirement and the safety margin as interference constraints have been modeled from practical aspects of cranes in operations. The completion time of tasks calculated in the model contains the travel time of crane from one position to another one. Owing to the studied problem being NP-complete, a novel discrete artificial bee colony algorithm (DABC) has been presented to obtain the near optimal solution. According to redefining the classical plus, subtract and multiply operators, the algorithm reserves the analogous search characteristic of the classical ABC equations. Moreover, a well-designed decoding scheme has been employed to obtain the GC schedule. Finally, computational comparisons have been performed to evaluate the performance of the proposed algorithm. The results show that the proposed DABC algorithm obtains near optimal solutions within reasonable runtime. The standard solver CPLEX is competitive if the instances are of small size, whereas the DABC algorithm is capable of delivering fairly good solutions even for the large-sized instances. Moreover, the solutions given by the DABC algorithm is significantly better than the solutions delivered by GA for the investigated instances. Therefore, the proposed DABC algorithm is a competitive approach to solve the GCSP in the container terminal.

Furthermore, related scheduling problems, such as truck scheduling and storage allocation that are highly interdependent with gantry crane scheduling, need to appropriately integrate into the GCSP, which is another challenging topic for further research.

Acknowledgements

This research is partially supported by the National Natural Science Foundation of China (No.51175442), the Youth Foundation for Humanities and Social Sciences of Ministry of Education of China (No. 12YJCZH296), the Ph.D. Programs Foundation of Ministry of Education of China (No.200806131014), the Fundamental Research Funds for the Central Universities (No.SWJTU09CX022; 2010ZT03) and the Scientific Research Program of Education Bureau of Sichuan Province, China (No. 12ZB322). The authors thank the two anonymous referees for their valuable comments. Finally, we are indebted to Professor Shengfeng Qin for his useful suggestions.

References

1. Ministry of Railway, China. Railway Statistical Bulletin. Retrieved May 1, 2012, http://www.china-mor.gov.cn/zwzc/tjxx/. (2012) (in Chinese).
2. M. Sammarra, J.-F. Cordeau, G. Laporte and M.F. Monaco, A tabu search heuristic for the quay crane scheduling problem, *Journal of Scheduling*. 10 (4-5) (2007) 327-336.
3. N. Boysen, F. Jarhn and E. Pesch, Scheduling freight trains in Rail-Rail transshipment yards, *Transportation Science*. 45 (2) (2011) 199-211.
4. T. Benna and M. Gronalt, Generic Simulation for rail-road container Terminals. in *Proceedings of Winter Simulation Conference*. (Miami, FL, United states, 2008), pp. 2656-2660.
5. P. Guo, W. Cheng and M. Zhang, Simulation on the operation influence by external factors for railway container logistic center. in *Proceeding of International Conference of Logistics Engineering and Management*. (Chengdu, China, 2010), pp. 2915-2922.
6. E. Kozan, Optimising container transfers at multimodal terminals, *Mathematical and Computer Modelling*. 31 (10-12) (2000) 235-243.
7. P. Corry and E. Kozan, An assignment model for dynamic load planning of intermodal trains, *Computers & Operations Research*. 33 (1) (2006) 1-17.
8. B.J. Jeong and K.H. Kim, Scheduling operations of a rail crane and container deliveries between rail and port terminals, *Engineering Optimization*. 43 (6) (2011) 597-613.
9. C. Bierwirth and F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, *European Journal of Operational Research*. 202 (3) (2010) 615-627.
10. C.F. Daganzo, The crane scheduling problem, *Transportation Research Part B: Methodological*. 23 (3) (1989) 159-175.
11. R.I. Peterkofsky and C.F. Daganzo, A branch and bound solution method for the crane scheduling problem, *Transportation Research Part B: Methodological*. 24 (3) (1990) 159-172.
12. A. Lim, B. Rodrigues, F. Xiao and Y. Zhu, Crane scheduling with spatial constraints, *Naval Research Logistics*. 51 (3) (2004) 386-406.
13. K.H. Kim and Y.-M. Park, A crane scheduling method for port container terminals, *European Journal of Operational Research*. 156 (3) (2004) 752-768.
14. 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*. 53 (1) (2006) 45-59.
15. W.C. Ng and K.L. Mak, Quay crane scheduling in container terminals, *Engineering Optimization*. 38 (6) (2006) 723-737.
16. C. Bierwirth and F. Meisel, A fast heuristic for quay crane scheduling with interference constraints, *Journal of Scheduling*. 12 (4) (2009) 345-360.
17. Y. Zhu and A. Lim, Crane scheduling with non-crossing constraint, *Journal of the Operational Research Society*. 57 (12) (2006) 1464-1471.
18. A. Lim, B. Rodrigues and Z. Xu, A m-parallel crane scheduling problem with a non-crossing constraint, *Naval Research Logistics*. 54 (2) (2007) 115-127.
19. D.-H. Lee, H.Q. Wang and L. Miao, Quay crane scheduling with non-interference constraints in port container terminals, *Transportation Research Part E: Logistics and Transportation Review*. 44 (1) (2008) 124-135.
20. D.-H. Lee, H.Q. Wang and L. Miao, Quay crane scheduling with handling priority in port container terminals, *Engineering Optimization*. 40 (2) (2008) 179-189.
21. D.-H. Lee and H.Q. Wang, An approximation algorithm for quay crane scheduling with handling priority in port

container terminals, *Engineering Optimization*. 42 (12) (2010) 1151-1161.

22. P. Legato, R. Trunfio and F. Meisel, Modeling and solving rich quay crane scheduling problems, *Computers & Operations Research*. 39 (9) (2012) 2063-2078.
23. Y.M. Park and K.H. Kim, A scheduling method for berth and quay cranes, *OR Spectrum*. 25 (1) (2003) 1-23.
24. F. Meisel and C. Bierwirth, Heuristics for the integration of crane productivity in the berth allocation problem, *Transportation Research Part E: Logistics and Transportation Review*. 45 (1) (2009) 196-209.
25. C. Zhang, L. Zheng, Z. Zhang, L. Shi and A.J. Armstrong, The allocation of berths and quay cranes by using a sub-gradient optimization technique, *Computers and Industrial Engineering*. 58 (1) (2010) 40-50.
26. B. Raa, W. Dullaert and R.V. Schaeren, An enriched model for the integrated berth allocation and quay crane assignment problem, *Expert Systems with Applications*. 38 (11) (2011) 14136-14147.
27. C. Liang, Y. Huang and Y. Yang, A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning, *Computers and Industrial Engineering*. 56 (3) (2009) 1021-1028.
28. D.-H. Lee and H. Qiu Wang, Integrated discrete berth allocation and quay crane scheduling in port container terminals, *Engineering Optimization*. 42 (8) (2010) 747-761.
29. D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, in: Technical Report TR06, Computer Engineering Department, Erciyes University, Kayseri, Turkey, 2005.
30. D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*. 39 (3) (2007) 459-471.
31. D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*. 8 (1) (2008) 687-697.
32. D. Karaboga and B. Akay, A comparative study of artificial bee colony algorithm, *Applied Mathematics and Computation*. 214 (1) (2009) 108-132.
33. J.-Q. Li, Q.-K. Pan and K.-Z. Gao, Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems, *International Journal of Advanced Manufacturing Technology*. 55 (9-12) (2011) 1159-1169.
34. Q.-K. Pan, M. Fatih Tasgetiren, P.N. Suganthan and T.J. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, *Information Sciences*. 181 (12) (2011) 2455-2468.
35. W.Y. Szeto, Y. Wu and S.C. Ho, An artificial bee colony algorithm for the capacitated vehicle routing problem, *European Journal of Operational Research*. 215 (1) (2011) 126-135.
36. M.F. Tasgetiren, Q.-K. Pan, P.N. Suganthan and A.H.L. Chen, A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops, *Information Sciences*. 181 (16) (2011) 3459-3475.
37. Q. Niu, B. Jiao and X. Gu, Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time, *Applied Mathematics and Computation*. 205 (1) (2008) 148-158.
38. A.H. Kashan and B. Karimi, A discrete particle swarm optimization algorithm for scheduling parallel machines, *Computers and Industrial Engineering*. 56 (1) (2009) 216-223.
39. J. Kennedy and R. Eberhart, Particle swarm optimization. in *Proceeding of IEEE International Conference on Neural Networks*. (Perth, Australia, 1995), pp. 1942-1948.
40. R. Ruiz and T. Stutzle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, *European Journal of Operational Research*. 177 (3) (2007) 2033-2049.