Research Journal of Applied Sciences, Engineering and Technology 4(9): 1117-1123, 2012
ISSN: 2040-7467

Submitted: November 17, 2011 Accepted: December 16, 2011 Published: May 01, 2012

Particle Swarm Optimization for Gantry Crane Scheduling with Interference Constraints

[1]Peng Guo, [1]Wenming Cheng and [2]Jian Liang
[1]School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China
[2]School of Mechanical Engineering and Automation, Xihua University, Chengdu 610039, Chain

Abstract: This study investigates the problem of scheduling gantry cranes, which are the main handling equipment in railway container terminals. Since the gantry cranes share two common tracks, some interference constraints are considered for the scheduling problem. The scheduling problem is formulated as an integer programming model. In the view of the high computational complexity, a Particle Swarm Optimization (PSO) algorithm is proposed. The effectiveness of the PSO algorithm is evaluated by comparing its results to traditional genetic algorithm and CPLEX on some random instances. Experimental results show that the PSO algorithm reports better quality solution in a short time on larger problem instances.

Key words: Gantry crane scheduling, interference constraint, particle swarm optimization, railway container terminal

INTRODUCTION

Recently, railway container terminals that serve as the key hubs of hinterland logistics transportation have been built in China. Rising competition between various transportation means motivate their operators to improve their customer service. In terms of terminal service level, it is often measured by the terminal turnaround time, i.e. the necessary amount of time is needed to serve a train by some Gantry Cranes (GCs). In most terminals, the discharging and loading of containers by GCs account for the biggest portion of a train terminal turnaround time.

In a railway container terminal, each railroad track working area is served by a number of gantry cranes. After a container train arrived at the terminal, the operator must consider the scheme of GCs. In this paper, a container train is typically divided longitudinally into different discharging/loading tasks according to the attributes of containers, such as position, size, destination for outbound containers, or origin for inbound containers. Here, the definition of a task is referred 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 corresponding handling operation. A typical task partition is described in Fig. 1. Since GCs are tracks mounted, some interference constraints are involved in the Gantry Crane Scheduling Problem (GCSP), 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, the adjacent GCs must keep a suitable distance for ensuring safety in discharging or loading operations. In practice, only one gantry crane can work on a task at any time. Therefore, the crane interference constraints must be included in the formulate model so as to make schedule feasible and rigorous.

There is almost no study on crane scheduling for railway container terminal, but the Quay Crane Scheduling Problems (QCSP) of port terminals have received great attention in the literature. Daganzo (1989) firstly studied the static and dynamic quay crane scheduling problems. Subsequently, Lim *et al.* (2004) provided an integer programming model with non-crossing constraint for the objective of maximizing the total profit. Kim and Park (2004) defined a task as a discharging or loading for a cluster of adjacent slots on one container vessel, and formulated a mixed integer programming model, which considers non-crossing constraint related to the operation of quay cranes. Moccia *et al.* (2006) revised the Kim and Park formulation that may yield 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. Ng and Mak (2006) proposed a scheduling heuristic to find effective schedules. Owing to lacking a correct treatment of crane safety constraints, Bierwirth and Meisel (2009) presented a revised optimization model for the scheduling of quay cranes. Furthermore, simulated annealing, tabu search and genetic algorithm were applied to solve the scheduling problem (Zhu and Lim, 2006; Sammarra *et al*., 2007; Lee *et al*., 2008).

In this study, the non-cross constraints and the safety constraints are considered simultaneously in the researched problem, and the mixed integer programming

Corresponding Author: Peng Guo, School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China

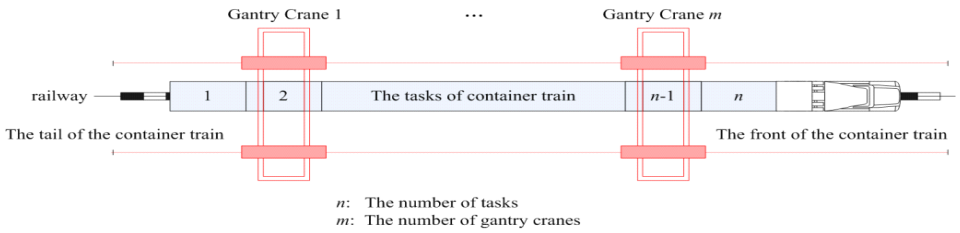

Fig. 1: Typical task partition of a container train

model is addressed. In order to tackle the problem, a Particle Swarm Optimization (PSO) algorithm is employed to yield the near optimal solution of the problem under consideration. Then, the gantry crane schedule can be obtained by using a heuristic procedure. According to the computational experiments, the results show that PSO schemes are effectiveness and efficiency for solving the scheduling problem.

INTEGER PROGRAMMING MODEL

In this section, a mixed integer programming model for the GCSP is formulated. Some following constraints must be considered in the problem: All tasks have different original processing times but the operation rate of cranes is constant. No preemption is allowed among all tasks. That is to say, once a gantry crane starts to process a task, it must complete the task before another task is processed. All gantry cranes move between two adjacent tasks in uniform travel time; Gantry cranes located in the same loading area are operated on the same track and cannot cross each other. In addition, the cranes line up in one track 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 a track from left to right, and tasks 1, 2, 3, …, *n* are in the similar manner. Adjacent GCs have to keep a safety distance at any time.

The following notations will be used to define the problem:

- n The number of tasks
- m The number of gantry cranes
- p_i The processing time of task i (i = 1, 2, …, n)
- s The necessary safety distance between adjacent GCs
- r_k The earliest available time of GC k (k = 1, 2, …, m)
- l_i The location of task i that is expressed by the task number
- l_k^0 The initial position of GC k 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 (i, j = 1, 2, ..., n)
- M A sufficiently large positive number.

The decision variables are defined as follows:

- C_i integer, the completion time of task i (i = 1, 2, …, n)
- $C_{\max}$ integer, the maximum completion time among all tasks
- x_{ik} binary, set to 1 if task i is assigned to crane k; 0, otherwise (i = 1, 2, ..., n and k = 1, 2, …, m)
- y_{ij} binary, set to 1 if task i completes no later than task j starts; 0, otherwise (i, j = 1, 2, ..., n)

The scheduling problem can be stated mathematically as below, followed by a brief explanation:

Minimize:
$C_{\max}$
Subject to:

$$C_{max} \geq C_i \quad i = 1, 2, \ldots, n \tag{1}$$

$$C_i - p_i \geq 0 \quad i = 1, 2, \ldots, n \tag{2}$$

$$\sum_{k=1}^{m} x_{ik} = 1 \quad i = 1, 2, \ldots, n \tag{3}$$

$$t_{ij} = t^0 |l_i - l_j| \; i, j = 1, 2, \ldots, n \tag{4}$$

$$\sum_{k=1}^{m} x_{ik} (r_k + t_{0i}^k) - C_i + p_i \leq M(1 - y_{0i})$$

$$i = 1, 2, \ldots, n \tag{5}$$

$$C_i - (C_j - p_j - t_{ij}) + y_{ij} M \geq 0 \; i, j = 1, 2, \ldots, n \tag{6}$$

$$C_i - (C_j - p_j - t_{ij}) - (1 - y_{ij}) M \leq 0 \; i, j = 1, 2, \ldots, n \tag{7}$$

$$M(y_{ij} + y_{ji}) \geq \sum_{k=1}^{m} k x_{ik} - \sum_{l=1}^{m} l x_{jl} + 1$$

$$i, j = 1, 2, \ldots, n \tag{8}$$

$$M(y_{ij} + y_{ji}) \geq i + s + 1 - j \; i, j = 1, 2, \ldots, n \tag{9}$$

The objective of the scheduling problem is to minimize the maximum completion time of all tasks.

Constraints (1) and (2) define the properties of decision variables C_{max} and C_i. Constraint (3) ensures that every task must be assigned exactly to one crane. Constraint (4) calculates the travel time of a gantry crane between position l_i and l_j. Constraint (5) gives the earliest starting time of loading or discharging operations by every gantry crane. Constraints (6) and (7) define the property of decision variable y_{ij} Constraint (6) indicates that $y_{ij} = 1$ if $C_i \leq C_j - p_j$, which means $y_{ij} = 1$ when task i finishes no later than task j starts; constraint (7) indicates that $y_{ij} = 0$ if $C_i > C_j - p_j$, which means $y_{ij} = 0$ when task i finishes after task j starts. Constraint (8) imposes non-crossing constraint between gantry cranes located in the same track. 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 ascend 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 $k + 1 \leq l$. Constraint (9) guarantees that adjacent cranes have to keep a safety distance at any time when the cranes perform tasks simultaneously. In this paper, $t^0 = 1$ and $s = 1$.

The above model can be solved in a commercial mixed integer programming solver such as CPLEX. However, the problem is NP-complete because the quay crane scheduling problem with the non-cross constraints has proven to be NP-complete by Lee *et al.* (2008). As the number of tasks and cranes increase, the commercial solver will consume a great amount of CPU time to solve the problem optimally. The experimental results also indicate that CPLEX requires long CPU time to solve larger problem instances. Therefore, commercial solvers may not be a viable tool to use in real life. In order to tackle the problem under research more efficiently and effectively, a Particle Swarm Optimization (PSO) algorithm is employed to obtain near optimization solutions in the next section.

PROPOSED PARTICLE SWARMOPTIMIZA TION ALGORITHM

Particle Swarm Optimization (PSO) is a new swarm optimization meta-heuristic based on the social behavior of bird flocking or fish schooling (Kennedy and Eberhart, 1995). It has been widely applied to a large class of combinatorial optimization problems, such as traveling salesman problem (Shi *et al*., 2007) and production scheduling problems (Lian *et al*., 2006; Sha and Hsu, 2006; Niu *et al*., 2010). In a PSO system, a population of random particles (solutions) is initialized and searched for optima according to updating iterations. Each particle flies in the D-dimensional search space of the optimization problem with a velocity which provides information about direction and changing rate. The position of a particle helps to find a feasible solution for the problem under consideration. Its fitness value defines the quality of the solution. Particles are guided by two components: cognitive information based on its own experience and social information based on observation of its neighbors. The details of the proposed PSO algorithm designed for the problem are elaborated as follows:

Table 1: Representation of position and velocity for a particle

Dimension j	1	2	3	4	5	6
Position value	3.72	1.12	2.79	1.84	4.09	3.81
Velocity value	-3.28	4.68	-2.59	1.58	-1.88	-0.45
SPV	4	1	3	2	6	5

Solution representation and decoding: For the problem under consideration, there are n tasks to schedule. Therefore, each particle should have position information for all tasks. The position of particle i in tth iteration is represented as $x_i(t) = (x_{i1}(t),..., x_{ij}(t), ..., x_{in}(t))$, where $x_{ij}(t)$ is the position of the ith with respect to the jth task in tth iteration. Then, the smallest position value (SPV) rule is used to convert the continuous position values of particles to a task sequence. The sequence of the tasks denotes the operating order of the task by the cranes:

$$x_{ij}(0) = x_{min}+(x_{max} - x_{min})*rand(0,1) \quad (10)$$

$$v_{ij}(0) = v_{min}+(v_{max} - v_{min})*rand(0,1) \quad (11)$$

The initial position of each particle is initialized using Eq. (10). In this study, $x_{min} = 0.0$ and $x_{max} = 5.0$. Equation (11) is used to define the initial velocity for each particle, where $v_{min} = -5.0$ and $v_{max} = 5.0$. For example, consider a problem with 6 tasks. The initial position ($x_{ij}(0)$) and velocity ($v_{ij}(0)$) for a particle i is generated according to Eq. (10) and (11), as illustrated in Table 1. Once the position values for each particle are given then the sequence in which tasks will be determined using the SPV rule. This sequence is shown in the last row of Table 1.

Gantry cranes schedule constructs: Once one task sequence is given, a GCs schedule can be constructed by assigning tasks to GCs using the following heuristic. For each sequence of tasks, 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 time of the additional cranes are set infinity, and the others are equal to their individual start time. The two available GCs are chosen based on the current location of each GC. The unassigned task in the task sequence is assigned to the gantry crane that is selected by comparing the earliest available time of the two available GCs, the distance

Heuristic Algorithm: LTDN
Inputs: n, m, x, $ptime$, $crane\_location^0$, r, t^0 // x is a given sequence of all tasks, $ptime_i$ is the processing time of task i, $crane\_location_k^0$ is the initial position of gantry crane k, r_k is the earliest available time of gantry crane k and t^0 is the traveling time of a gantry crane between any two adjacent tasks //
Output: C_{max}// C_{max} is the maximum completion time among all tasks //
Begin
let $crane\_location \leftarrow$(0, $crane\_location_1^0$, …, $crane\_location_m^0$, n+1),
$crane\_r \leftarrow$(infinity, r_1, …, r_m, infinity) //add two dummy GCs 0 and n+1 //
set $crane\_c_k \leftarrow 0$, for $1 \le k \le m + 2$, $task\_time_i \leftarrow 0$, for $1 \le I \le$ n
// initialize the completion time of gantry crane k and the completion time of task i //
set $i \leftarrow 1$
while ($i \le n$) **do**
  for (j=1; $j \le m$+1; j++) **do**
    if $x(i) \ge crane\_location(j)$ and $x(i) \le crane\_location(j+1)$ **then**
      if $crane\_r(j) < crane\_r(j+1)$ **then**
        task $x(i)$ is assigned to crane j
      elseif $crane\_r(j) > crane\_r(j+1)$ **then**
        task $x(i)$ is assigned to crane j+1
      else
        if $|crane\_location(j) - x(i)| < |crane\_location(j+1) - x(i)|$ **then**
          task $x(i)$ is assigned to crane j
        elseif $|crane\_location(j) - x(i)| > |crane\_location(j+1) - x(i)|$ **then**
          task $x(i)$ is assigned to crane j+1
        else
          task $x(i)$ is assigned to crane j
        end if
      end if
      update $crane\_c(k)$, $crane\_location(k)$, $crane\_r(k)$ and $task\_time(x(i))$
      break // terminate the execution of for loop //
    end if
  end for
  $i \leftarrow i + 1$
end while
$C_{max} \leftarrow \max\{ task\_time(x(1)), …, task\_time(x(n)) \}$ // calculate the C_{max} //
End

Fig. 2: Pseudo-code of heuristic LTDN

between this task and these two available GCs, or the number of the GCs. The heuristic is named LTDN (Location, Time, Distance and Number) according to the decision process of the available crane selection. Figure 2 provides a detailed pseudo-code of LTDN.

The gantry crane schedule obtained from the proposed procedure does not violate the non-crossing constraints, but it may violate the safety constraints. 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 (6) and (7), y_{ij} (i, j = 1, 2, n) 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 violates constraints (9), the objective value of its corresponding schedule is set to infinity.

Table 2: Factors and levels considered to generate the instances

	Levels	
Factors	Small size instances	Large size instances
Number of tasks	8, 10, 12, 14, 16, 18	20, 25, 30, 35, 40, 45, 50, 60, 80, 100
Number of cranes	2, 3	3, 4
Processing time distribution	Discrete uniform (20, 150)	Discrete uniform (30,180)
Initial location of cranes	Discrete uniform [1, n]	

Particle updates: each particle (i.e., i) at tth iteration is associated with a position vector, $x_i(t) = (x_{i1}(t),..., x_{ij}(t),..., x_{in}(t))$, and a velocity vector $v_i(t) = (v_{i1}(t),..., v_{ij}(t),...,v_{in}(t))$. Let $pbest_i(t)$ be the personal best solution that particle i has obtained until the tth iteration, and $gbest(t)$ be the global best solution obtained from $pbest_i(t)$ in the whole swarm at the tth iteration. The velocity and position of particle i in the tth iteration are updated by following equations:

$$v_i(t) = \omega \cdot v_i(t-1) + c_1 \cdot r_1 \cdot (pbest_i(t-1) - x_i(t-1)) + c_2 \cdot r_2 \cdot (gbest(t-1) - x_i(t-1)) \quad (12)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (13)$$

where ω is the inertia weight which controls the impact of the previous velocities on the current velocity. c_1 is the cognition learning factor and c_2 is the social learning factor. r_1 and r_2 are random numbers between (0,1).

In summary, the proposed PSO algorithm for GCSP can be illustrated as follows:

- Initialize a swarm of particles with random positions and velocities in the whole scheduling problem space, and evaluate their objective values. Then *gbest* is searched and $pbest_i$ of each particle is ser as its initial position.
- Update the velocity and position of each particle according to Eq. (12) and (13). Evaluate the objective values of all particles and update *gbest* and $pbest_i$ if necessary.
- If a stopping criterion is met, then output *gbest* and its objective value; else go to Step 2.

COMPUTATIONAL EXPERIMENTS AND DISCUSSION

Data sets are randomly generated to test the performance of the proposed PSO approach. Table 2 presents the factors and different levels considered to generate the problem instances. The data sets consist of small size instances and large size instances. The number of tasks considered in the instances ranges from 8 to 100.

Table 3: The experimental results of random instances with small sizes

		CPLEX		GA		PSO	
Instance No.	Size (Tasks× Cranes)	Value	Time	Best/average/ worst (ARD)	Time	Best/average/ worst (ARD)	Time
1	8×2	428	0.73	428/428.1/430 (0.02)	1.24	428/428.9/431 (0.21)	1.53
2	8×3	292	0.13	292/292/292 (0.00)	1.25	292/292/292 (0.00)	1.49
3	10×2	416	1.30	416/416/416 (0.00)	1.50	416/416/416 (0.00)	1.82
4	10×3	299	0.22	299/299.8/301 (0.27)	1.51	299/299/299 (0.00)	1.81
5	12×2	415	217.81	415/416.3/419 (0.31)	1.76	415/416.8/418 (0.43)	2.07
6	12×3	280	0.47	280/280/280 (0.00)	1.76	280/280.5/282 (0.18)	2.07
7	14×2	723	1800.13	725/727/729 (0.55)	2.04	723/724.5/726 (0.21)	2.33
8	14×3	489	60.53	489/492.4/496 (0.70)	2.05	489/489/489 (0.00)	2.34
9	16×2	724	1799.97	728/730.2/732 (0.86)	2.33	724/724.7/730 (0.10)	2.63
10	16×3	480	1799.98	480/482.5/486 (0.52)	2.35	480/481.3/483 (0.27)	2.63
11	18×2	669	1799.97	675/680.1/682 (1.66)	2.60	669/671.5/675 (0.37)	2.88
12	18×3	450	1800.11	453/454.9/457 (1.09)	2.64	450/451.1/454 (0.24)	2.92

For the two kinds of instances, the processing times of the task and the initial location of cranes are generated from discrete uniform distribution with the parameters as shown in the Table 2. From the table, the algorithms are used to solve 6×2+10×2 = 32 problems. To evaluate the proposed algorithm, experiments are conducted to make a comparison with Genetic Algorithm (GA) and CPLEX. For GA, single-point order crossover operation and swap mutation operation are adopted. The crossover rate and the mutation rate are set to 0.8 and 0.2. All algorithms run thirty times on each instance, with the following settings: the population size PS = 100, the maximum iteration max_iter = 200. The average, the best and the worst objective values and are reported in the 30 runs. Furthermore, the average run times required by different approaches to solve the problem instances are also observed among 30 runs. PSO and GA are implemented in Matlab 7.11 software and running on a PC with a Pentium Dual-Core, E5300 2.60GHz processor with 2GB RAM. The preliminary experiments are conducted to fine-tune the PSO parameters (i.e., inertia parameter ω, cognitive and social parameters c_1 and c_2). According to the results, the PSO can yield the best performance with the following values: $\omega = 0.8$, $c_1 = c_2 = 1.5$.

As the number of task increases, CPLEX will consume long run time to solve the GCSP. Even after running for several hours, CPLEX do not converge to an optimum. Consequently, CPLEX is applied to obtain a single integer feasible solution within 1800s for small size instances. For large size instances, the Lower Bound (LB) of the objective value is computed to evaluate the PSO by CPLEX. According to ignoring non-cross constraints and safety constraints, the lower bound of each instance can be obtained. In addition, the mean relative deviation (MRD) is used as an index to evaluate the performance of the PSO algorithm. this index is computed as follows:

$$MRD = (C_{mean} - C_{optimal(low)})/C_{optimal(low)},$$

where C_{mean} is the average objective value of the thirty runs, and $C_{optimal(low)}$ is the optimal value of CPLEX or the lower bound. It can reflect an algorithm availability.

The results of the small size instances are summarized in Table 3. From Table 3, it can be observed that the performance of PSO is slightly better than GA but not significant. When the number of the tasks is less than 10, the computational time of CPLEX is very short. Once the number of the tasks exceeds 10, CPLEX consumes large computational time in solving the GCSP. In addition, since these instances with three cranes produce fewer nodes in the calculation process, CPLEX needs shorter time to solve these problems. However, the proposed algorithm and GA can obtained the optimal or near optimal solution within three seconds. For the large

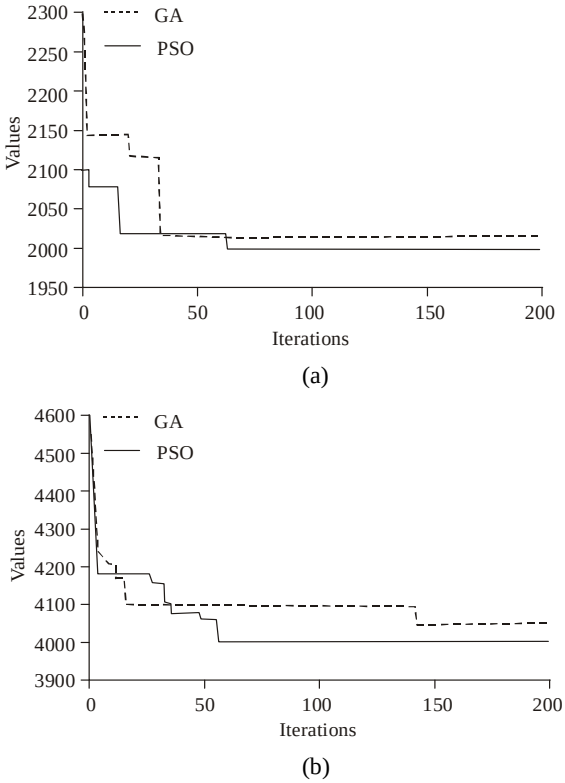

Fig. 3: Evolution curves of the optimal solution for PSO and GA with different instances

Table 4: The experimental results of random instances with large sizes

			GA		PSO	
Instance No.	Size(Tasks × Cranes)	LB	Best/average/ worst (ARD)	Time	Best/average/ worst (ARD)	Time
1	20×3	783	804/811.6/817 (3.65)	2.95	800/809.5/819 (3.38)	3.22
2	20×4	587	617/621.6/624 (5.89)	3.00	607/621.4/638 (5.86)	3.26
3	25×3	924	952/958.5/964 (3.73)	3.81	950/957.6/965 (3.64)	4.03
4	25×4	693	712/719.3/724 (3.80)	3.84	707/718.7/741 (3.71)	4.08
5	30×3	1112	1158/1161.6/1168 (4.46)	4.67	1143/1155.2/1167 (3.88)	4.87
6	30×4	834	883/886.3/889 (6.27)	4.81	870/879.7/899 (5.48)	4.92
7	35×3	11141	174/1181.4/1186 (6.05)	5.59	1159/1169.9/1185 (5.02)	5.81
8	35×4	836	877/887.0/896 (6.10)	5.68	870/881.2/894 (5.41)	5.88
9	40×3	1340	1418/1426.7/1439 (6.47)	6.61	1400/1411.8/1428 (5.36)	6.81
10	40×4	1005	1067/1074.3/1088 (6.90)	6.72	1052/1069.8/1091 (6.45)	6.85
11	45×3	1613	1713/1732.2/1752 (7.39)	7.63	1689/1707.2/1724 (5.84)	7.81
12	45×4	1210	1285/1282.2/1296 (5.97)	7.84	1266/1285.2/1310 (6.21)	7.89
13	50×3	1877	2000/2014.3/2028 (7.31)	8.61	1972/1995.3/2022 (6.30)	8.92
14	50×4	1408	1497/1515.5/1527 (7.63)	8.88	1477/1500.1/1542 (6.54)	9.03
15	60×3	1960	2123/2169.6/2199 (10.69)	11.17	2102/2134.2/2168 (8.89)	11.17
16	60×4	1470	1623/1628.3/1668 (10.77)	11.32	1569/1600.3/1635 (8.89)	11.67
17	80×3	2816	3129/3190.8/3230 (13.31)	16.58	3103/3135.2/3203 (11.34)	17.71
18	80×4	2112	2317/2366.2/2406 (12.04)	17.22	2316/2350.1/2402 (11.27)	18.15
19	100×3	3387	3944/4009.1/4082 (18.37)	23.29	3843/3914.9/3979 (15.59)	24.47
20	100×4	2540	2962/3014.8/3055 (18.69)	23.51	2871/2941.9/3075 (15.82)	25.70

size instances, the results are showed as Table 4. All results obtained by PSO are very close their corresponding lower bounds, so the ARD is relatively small (the maximum deviation among twenty instances is 15.82%, the minimum deviation is 3.38%, and the average value is 7.24%). The ARD is largely failed to get value 0 due to ignoring the travel time between any two adjacent cranes. It is clear from Table 4 that PSO significantly outperforms GA along with the increasing of sizes. In the twenty instances, there are 19 instances that are yielded the better solution by PSO. Figure 3 presents the evolution curves of the optimal solution for PSO and GA with different instances. The PSO can be converged fast to the approximate optimal solution. Generally speaking, for both solution accuracy and efficiency consideration, the proposed PSO algorithm is more suitable than the GA for solving the GCSP

CONCLUSION AND FUTURE STUDY

The study deals with the gantry crane scheduling problems in railway container terminals. Since the gantry cranes are mounted on two common tracks, the non-crossing constraints and safety constraints must be considered in the problem under research. A mixed integer programming model has been proposed for the scheduling problem. It has been shown that finding the optimal solution of the scheduling problem is very time consuming. So a PSO algorithm has been presented to obtained effective schedules. The performance of PSO is evaluated in comparison with CPLEX and well-known GA for a series of randomly generated instances based on the operation data of terminals. The results show that the PSO outperformed both GA and CPLEX in terms of solution quality and run time. PSO requires a smaller number of iteration to obtain the near optimal solution comparing to GA. As the number of tasks considered increased, the proposed PSO algorithm is more suitable than the GA for solving the GCSP. A lot of efforts can be made in the future to improve the current work. For example, external truck, as the main customer of railway container terminal, should be considered in the crane scheduling problem. Owing to the uncertainty of pickup time, the container re-handling operation can be also considered simultaneously.

ACKNOWLEDGMENT

This study is supported by National Science Foundation of China (No.51175442) and the Fundamental Research Funds for the Central Universities (No.2010ZT03).

REFERENCES

Bierwirth, C. and F. Meisel, 2009. A fast heuristic for quay crane scheduling with interference constraints. J. Scheduling, 12(4): 345-360.

Daganzo, C.F., 1989. The crane scheduling problem. Transport. Res. Part B-Meth., 23(3): 159-175.

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimizationed. Proceedings of the IEEE International Conference on Neural Networks, Perth, Aust, pp: 1942-1948.

Kim, K.H. and Y.M. Park, 2004. A crane scheduling method for port container terminals. Eur. J. Oper. Res., 156(3): 752-768.

Lee, D.H., H.Q. Wang, and L. Miao, 2008. Quay crane scheduling with non-interference constraints in port container terminals. Transportat. Res. E- Logi., 44(1): 124-135.

Lian, Z.G., X.S. Gu and B. Jiao, 2006. A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan. Appl. Math. Comp., 175(1): 773-785.

Lim, A., B. Rodrigues, F. Xiao and Y. Zhu, 2004. Crane scheduling with spatial constraints. Nav. Res. Log., 51(3): 386-406.

Moccia, L., J.F. Cordeau, M., Gaudioso and G. Laporte, 2006. A branch and cut algorithm for the quay crane scheduling problem in a container terminal. Nav. Res. Log., 53(1): 45-59.

Ng, W.C. and K.L. Mak, 2006. Quay crane scheduling in container terminals. Eng. Optimiz., 38(6): 723-737.

Niu, Q., T. Zhou and L. Wang, 2010. A hybrid particle swarm optimization for parallel machine total tardiness scheduling. Inter. J. Adv. Manuf. Technol., 49(5-8): 723-739.

Sammarra, M., J.F. Cordeau, G. Laporte and M.F. Manaco, 2007. A tabu search heuristic for the quay crane scheduling problem. J. Scheduling, 10(4-5): 327-336.

Sha, D.Y. and C.Y. Hsu, 2006. A hybrid particle swarm optimization for job shop scheduling problem. Comp. Indus. Eng., 51(4): 791-808.

Shi, X.H., Y.C. Liang, H.P. Lee, C. Lu and Q.X. Wang, 2007. Particle swarm optimization-based algorithms for tsp and generalized tsp. Inf. Proc. Lett., 103(5): 169-176.

Zhu, Y. and A. Lim, 2006. Crane scheduling with non-crossing constraint. J. Oper. Res. Soc., 57: 1464-1471.