2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Particle Swarm Optimization for Integrated Yard Truck Scheduling and Storage Allocation Problem

B. Niu[1,2,3], T. Xie[1]
[1]College of Management
Shenzhen University, Shenzhen, China
[2]Hefei Institute of Intelligent Machine
Chinese Academy of Science, Hefei, China
ben.niu@polyu.edu.hk

Q.Q Duan[1], L.J. Tan[4]
[3]Department of Industrial and Systems Engineering
The Hong Kong Polytechnic University, Hong Kong
[4]Management School
Jinan University Guangzhou, China
mstlj@163.com

***Abstract*—The Integrated Yard Truck Scheduling and Storage Allocation Problem (YTS-SAP) is one of the major optimization problems in container port which minimizes the total delay for all containers. To deal with this NP-hard scheduling problem, standard particle swarm optimization (SPSO) and a local version PSO (LPSO) are developed to obtain the optimal solutions. In addition, a simple and effective 'problem mapping' mechanism is used to convert particle position vector into scheduling solution. To evaluate the performance of the proposed approaches, experiments are conducted on different scale instances to compare the results obtained by GA. The experimental studies show that PSOs outperform GA in terms of computation time and solution quality.**

Keywords—Yard truck scheduling; Storage allocation; Container terminal; Particle Swarm Optimization (PSO)

I. INTRODUCTION

As crucial interfaces between land and sea transportation modes, container terminals have been increasingly important and this trend is expected to continue. Therefore, the manager of container terminal should guarantee that the terminal system can handle the containers cost-effectively to meet the increased volume of container traffic. Operations in the field of container terminal involves Berth Allocation, Quay Crane (QC) Scheduling, Yard Crane (YC) Scheduling, Yard Truck (YT) Scheduling, Container Storage Allocation, etc. As we know, the YT connects the two ends, i.e., the QC and YC, therefore the YT scheduling is an important component of container terminal operations that needs to be addressed. In addition, the allocating strategies of storage space will greatly affect the efficiency on discharging operation. Hence, in this paper, the integrated Yard Truck Scheduling and Storage Allocation Problem (YTS-SAP) is considered.

The YTS-SAP problem has attracted many attentions by researchers in the last decade. Bish et al. [1] firstly considered the two problems as a whole by formulating an assignment model and subsequently solving it via a heuristic algorithm. Bish [2] extended the problem to a more complex situation, which not only consider the yard location assignment and the vehicle dispatching, but also include the scheduling of loading and discharging operations at QC. Bish et al. [3] further solved the problem via new heuristic algorithms which the asymptotic and absolute worst-case performance ratios were identified. Lee et al. [4] developed a mixed integer programming model considered the yard truck scheduling and the storage allocation at the same time to minimize the weighted average of total requests delay and total trucks travel time. A hybrid insertion algorithm was proposed to solve the model. Xue et al. [5] studied the integration of YT scheduling, the QC scheduling and the yard location assignment for the discharging containers. A two-stage heuristic algorithm was developed for solving it.

To the best of our knowledge, litter attention has been made to use PSO-based approach to solve YTS-SAP so far. Therefore, based on our previous study of PSO-based approach for trucking scheduling in container terminals[6], a PSO-based approach is proposed for YTS-SAP task in this paper.

Followed by the introduction is the mathematical model of the YTS-SAP in Section 2, A PSO algorithm with two types of mapping mechanisms are proposed to solve the YTS-SAP in Section 3. Computational experiments compared with GA are reported in Section 4. Finally, conclusions are summarized in Section 5.

II. PROBLEM DESCRIPTION

The YTS-SAP model used in this paper is the same as that used in literature [4] and the notations and mathematical model are given as follows.

Let us assume that a certain number of YTs are dispatched to handle containers, each of which is assigned to one route. Let R be the set of all routes indexed by r, where $|R|=m$. To simplify notation, we define job to be the movement of a container between its pick-up location and drop-off location denoted by i and j. For the loading job, the container is moved from the yard location where it is stored to the QC location where the container will be loaded for shipment. For the unloading job, the origin is the QC location where the container is discharged, and the destination is a yard location which is a decision variable. The set of all containers is denoted by J where $|J|=\mathrm{n}$, including the set J^+ of loading containers and the set J^- of unloading containers. For each job, a soft time window $[a_i, b_i)$ will be given to it. a_i is the starting time of job i,

that is, job i should be processed at or after a_i . The due time of job i , b_i can be violated with a penalty. The processing time t_i of job i may encounter two conditions. If job i is a loading job, t_i is known in advanced since the origin and destination of job i is predetermined. On the other hand, if job i is a unloading job, t_i is determined by which storage location is allocated to job i . Let job i start to be serviced at starting time w_i , which is a decision variable, then the completion time of service is:

$$c_i = w_i + t_i \tag{1}$$

The delay of job i is:

$$d_i = \max\{0, c_i - b_i\} \tag{2}$$

If job j is processed immediately after job i by the same YT, the relation between w_i and w_j is:

$$w_j = \max\{w_i + t_i + s_{ij}, a_j\} \tag{3}$$

where s_{ij} is the travel time (setup time) of empty trip between destination of job i and origin of job j , which is also a decision variable.

For each route, we add two dummy jobs ι_r and κ_r to denote the initial and final location of each YT. Let J' be the set of $J \cup \{\iota_r\}_{r\in R}$ and J'' be the set of $J \cup \{\kappa_r\}_{r\in R}$, denoting the origin of job i , o_i and destination of job i , d_i, respectively. The set of all yard locations for all loading and discharging jobs is denoted by L with indices p and q . We also denotes the travel time of YT along the shortest route between yard location p and q by τ_{pq} . Let ζ_k , where $k \in K$ be the index for the storage location (SL) k , where K is the set of all SLs. The followings is the processing time of job i and the setup time from the drop-off location of job i to the pick-up location of job j :

$$t_i = \begin{cases} \tau_{o_i,d_i} & \text{if Request i is a loading request} \\ \tau_{o_i,\xi_k} & \text{if Request i is a discharging request and allocated to SLk} \end{cases} \tag{4}$$

$$s_{ij} = \begin{cases} \tau_{d_i,o_j} & \text{if Request i is a loading request} \\ \tau_{\xi_k,o_j} & \text{if Request i is a discharging request and allocated to SLk} \end{cases} \tag{5}$$

The followings are additional notations used in model description:

x_{ik} $= 1$,if job i is allocated to SL k .
$= 0$,otherwise.

y_{ij} $= 1$,if job i is connected to job j in the same route.
$= 0$,otherwise.

Mathematical model:

$$\textit{Minimize} \sum_{i\in J} d_i$$

Subject to

$$\sum_{i\in J^-} x_{ik} = 1, \quad \forall k \in K \tag{6}$$

$$\sum_{k\in K} x_{ik} = 1, \quad \forall i \in J^- \tag{7}$$

$$\sum_{j\in J''} y_{ij} = 1, \quad \forall i \in J' \tag{8}$$

$$\sum_{i\in J'} y_{ij} = 1, \quad \forall j \in J'' \tag{9}$$

$$w_i \geq a_i, \quad \forall i \in J' \cup J'' \tag{10}$$

$$d_i \geq w_i + \mathrm{t_i} - \mathrm{b_i}, \quad \forall i \in J' \cup J'' \tag{11}$$

$$w_j + M(1 - y_{ij}) \geq w_i + \mathrm{t_i} + \mathrm{s_{ij}}, \quad \forall i \in J' \text{ and } J \in J'' \tag{12}$$

$$t_i = \tau_{o_i,d_i}, \quad \forall i \in J^+ \tag{13}$$

$$t_i = \sum_{k\in K} \tau_{o_i,\xi_k} x_{ik}, \quad \forall i \in J^- \tag{14}$$

$$s_{ij} = \tau_{d_i,o_i}, \quad \forall i \in J^+ \textit{ and } j \in J \tag{15}$$

$$s_{ij} = \sum_{k\in K} \tau_{\xi_k,o_j} x_{ik}, \quad \forall i \in J^- \textit{ and } j \in J \tag{16}$$

$$x_{ik}, y_{ij} \in \{0,1\}, \quad \forall i \in J', j \in J'' \textit{ and } k \in K \tag{17}$$

$$w_i \in \Re, \quad \forall i \in J' \cup J'' \tag{18}$$

$$t_i \in \Re, \quad \forall i \in J^- \tag{19}$$

$$s_{ij} \in \Re, \quad \forall i \in J^- \textit{ and } j \in J \tag{20}$$

III. METHODOLOGY

A. The background of particle swarm optimization

In 1995, Kennedy et al. [7] designed a population-based evolutionary computation technique named Particle swarm optimization (PSO) by drawing inspiration from the group foraging behavior of animals such as bird flocking, fish schooling. At the beginning of the evolutionary process, the system is initialized with random particles. Each particle flies in the searching space according to the experience of its own and its companions'. Therefore, particles are inclined to search for optima by updating generations. Let $V_i(v_{i1}, v_{i2}, \cdots, v_{in})$ be the *ith* particle's velocity vector, and $X_i(\mathrm{x}_{i1}, \mathrm{x}_{i2}, \cdots, \mathrm{x}_{in})$ be the *ith* particle's position vector, respectively. The two following equations are applied to update each particle's velocity and position:

$$v_{ij} \leftarrow w * v_{ij} + c_1 * rand() * (P_{ij} - x_{ij}) + c_2 * Rand() * (\mathrm{G_j} - x_{ij}) \tag{21}$$

$$x_{ij} \leftarrow x_{ij} + v_{ij} \tag{22}$$

Where j $(j = 1, 2, \cdots, D)$ represents the *jth* dimension of the search space. The inertia weight w was designed by Shi et al. [8], and it is applied to control the magnitude of the

previous velocities. The best previous position of particle $i\,(i=1,2,\cdots N)$ is denoted by $P_i(p_{i1},p_{i2},\cdots,p_{in})$, while the historically best solution of the neighborhood is represented by $G(\mathrm{g}_1,\mathrm{g}_2,\cdots,\mathrm{g}_n)$. In global version of PSO, this neighborhood is the entire swarm, while in local one, this neighborhood is a subset of the particles. c_1 is the self learning factor and c_2 is the social learning factor. $rand()$ and $Rand()$ are uniformly distributed random variables in $[0,1]$.

The scheduling problem in this paper consists of the containers scheduling sequence problem, trucks allocation problem, and storage space allocation problem, each of which is discrete combinatorial optimization problem. Hence, we need to find a suitable mapping method to apply PSO to discrete combinatorial optimization.

B. Mapping mechanism

The $(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)$ dimensional particle $\pi_i = [\pi_{i1},\pi_{i2},\cdots,\pi_{i(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)}]$ is used to map N^+ loading containers and N^- discharging containers with L potential locations transported by M trucks. The first $n^+ + n^-$ dimensions $\pi_i^J = [\pi_{i1},\pi_{i2},\cdots,\pi_{i(n^+ + n^-)}]$ denote the job permutation. The second l dimensions $\pi_i^L = [\pi_{i(n^+ + n^- + 1)},\pi_{i(n^+ + n^- + 2)},\cdots,\pi_{i(n^+ + n^- + l)}]$ related to potential locations denote the index of locations available to the discharging containers which are negative integers from -1 to $-L$. The last m dimensions $\pi_i^T = [\pi_{i(n^+ + n^- + l + 1)},\pi_{i(n^+ + n^- + l + 2)},\cdots,\pi_{i(n^+ + n^- + l + m)}]$ denote the assignment method of the M trucks. The number of jobs assigned to these trucks must be m positive integers whose sum is equivalent to the number N of jobs.

Fig. 1 gives an example of a particle encoding schema , in which for dispatching two trucks $(\mathrm{M}=2)$ will be designated to handle three discharging jobs $(N^-=3)$ and three loading jobs $(N^+=3)$ with three potential locations $(L=3)$ available to the discharging jobs (assume that the first three jobs are discharging jobs). The first truck is assigned to sequentially handle jobs 4, 1, 6, and the second truck is asked to sequentially handle jobs 3,2,and 5. The corresponding discharging job is located in ς_2, ς_1 and ς_3, respectively (see Fig. 2).

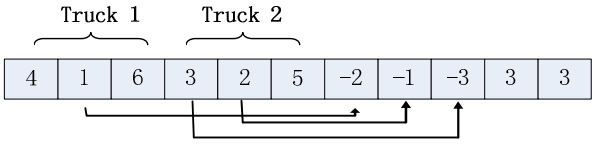

Fig. 1. An example for encoding scheme of YTS-SAP

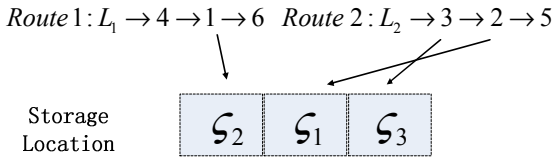

Fig. 2. Decoding of encoding scheme illustrated in Fig. 1

TABLE I. REPRESENTATION OF THE JOB PERMUTATION AND CORRESPONDING POSITION VALUES

Dimension, j	1	2	3	4	5	6
Position value, X_i^J	-0.87	-9.62	6.42	-1.10	2.30	5.83
Job permutation, π_i^J	3	1	6	2	4	5

In order to convert continuous position of particles $X_i = [\mathrm{x}_{i1},\mathrm{x}_{i2},\cdots,\mathrm{x}_{i(n^+ + n^-)}]$ into job sequence $\pi_i^J = [\pi_{i1},\pi_{i2},\cdots,\pi_{i(n^+ + n^-)}]$ in PSO, we introduced Smallest Position Value (SPV) rule to realize the mapping mechanism [9]. Based on the SPV rule, firstly, the particle value of each dimension in ascending order is sorted, and then the smallest floating value is placed as the smallest integer. The next smallest floating value is put by the next smallest integer until that a complete job permutation is constructed. An example of the excitation mode using SPV rule is presented in TABLE I.

In the truck solution representation part, we should guarantee that π_{ij}^T $(j=1,2,\cdots,m)$ is positive integers whose sum is equivalent to the number N of jobs $(\sum_{j=n+1}^{n+m} \mathrm{x}_{\mathrm{ij}} = n)$. The assignment algorithm for this mapping schema is presented as follows. Schematic example of the encoding procedure of the truck solution representation is shown in TABLE II, where 3 trucks are assigned to hand 9 jobs.

Step 1. Replace $X_i^T = [\mathrm{x}_{i(n+l+1)},\mathrm{x}_{i(n+l+2)},\cdots,\mathrm{x}_{i(n+l+m)}]$ by its absolute value $|X_i^T|$;

Step 2. Set $\mathrm{S_i}^{\mathrm{T}} = \sum_{j=n+l+1}^{n+l+m} \mathrm{x}_{\mathrm{ij}}$ to obtain sum of each dimensions of X_i^T;

Step 3. Set $\pi_{ij}^T = x_{ij} / S_i^T * n$, where n is the total number of jobs;

Step 4. Replace π_{ij}^T by its rounded value;

Step 5. If $\pi_{ij}^T = 0$,replace it by 1;

Step 6. If $\sum_{j=1}^{m} \pi_{ij}^T > n \; or < n$, find the set of maximum(minimum) numbers of $\pi_i^T = [\pi_{i1}^T,\pi_{i2}^T,\cdots,\pi_{im}^T]$ and let first number in the set minus (plus) 1;

Step 7.If $\sum_{j=1}^{m} \pi_{ij}^T = n$,then stop; otherwise return to step 6.

TABLE II. ENCODING PROCEDURE ILLUSTRATED IN ASSIGNMENT ALGORITHM($n = 9$)

Value	-1.41	-0.03	-2.92
Step 1	1.41	0.03	2.92
Step 2-3	2.88	0.09	6.03
Step 4	3	0	6
Step 5	3	1	6
Step 6-7	3	1	5

As shown in Fig. 1, the second l dimensions $\pi_i^L = [\pi_{i(n^+ + n^- + 1)}, \pi_{i(n^+ + n^- + 2)}, \cdots, \pi_{i(n^+ + n^- + l)}]$ are also the permutation of the integers from -1 to $-L$ (The minus here is used to differentiate them from the job permutation part). Therefore, the SPV rule is introduced again to formulate the permutation of storage locations. A simple example is provided in TABLE III to illustrate the encoding procedure.

TABLE III. REPRESENTATION OF THE STORAGE PERMUTATION AND CORRESPONDING POSITION VALUES

Dimension, j	1	2	3	4	5	6
Position value, X_i^L	2.89	6.35	3.20	-3.16	-4.20	-3.17
Storage permutation π_i^L	-4	-6	-5	-3	-1	-2

C. Initial population

Each particle's positions and velocities are updated using the following equations:

$$x_{ij}^1 = x_{\min} + (x_{\max} - x_{\min}) * r_1 \quad (23)$$

$$v_{ij}^1 = v_{\min} + (\mathrm{v}_{\max} - v_{\min}) * r_2 \quad (24)$$

where $x_{\min} = 0$, $x_{\max} = 10$, $v_{\min} = -1$, and $v_{\max} = 1$. r_1 and r_2 represent random values uniformly distributed in interval [0, 1].

D. Fitness computation

In YTS-SAP, the delay of job k processed by truck m implied by particle i , $\mathrm{d}_{\mathrm{im}}(k)$ equal to $\max\{0, C_{im}(k) - \mathrm{b}_{\mathrm{im}}(\mathrm{k})\}$,where $C_{im}(k)$ is the completion time and $\mathrm{b}_{\mathrm{im}}(\mathrm{k})$ is the due time of job k . Let $o_m(i)$ be the summation of delay generated by jobs assigned to truck $m\,(m = 1, 2, \cdots, M)$ based on the schedule π_i implied by particle i , the objective of YTS-SAP is to minimize the total delay of jobs, i.e.

$$O(i) = \sum_{m=1}^{M} o_m(i) \quad \mathrm{m} = (1, 2, \cdots, M) \quad (25)$$

E. Computational procedure

The whole computation procedure of the PSO for the YTS-SAP is described as follows:

Step 1: Set the parameter's values;

- Set the population size (denoted by Ps);
- Set the maximum iteration (denoted by $MaxGen$), w_{start} , w_{end} , c_1 , c_2 ;
- Set Dim as the number of dimensions to represent N^+ loading jobs, N^- discharging jobs, M trucks and L storage locations, which equal to $(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)$.

Step 2: Initialization;

- Set iteration $t = 1$;
- Initialize Ps particles stochastically using Equation (23), $\{X_i^1, i = 1, 2, \cdots, Ps\}$,where $X_i^1 = [x_{i1}^1, x_{i2}^1, \cdots, x_{i(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)}^1]$;
- Initialize the initial velocities for particle i stochastically using Equation (24), $\{\mathrm{V}_i^1, i = 1, 2, \cdots, Ps\}$,where $V_i^1 = [\mathrm{v}_{i1}^1, \mathrm{v}_{i2}^1, \cdots, \mathrm{v}_{i(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)}^1]$;
- Convert current particle $Xi^1 (i = 1, 2, \cdots, Ps)$ to the schedule $\pi_i^1 = [\pi_{i1}^1, \pi_{i2}^1, \cdots, \pi_{i(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)}^1]$ according to the SPV rule and Algorithm 1;
- Evaluate schedule π_i^1 of each particle i in the swarm by the objective function $O^1(i)$ for $i = 1, 2, \cdots, Ps$;
- Set the personal best position with a copy of particle itself, i.e. $P_i^1 = X_i^1$, together with its best fitness value, $f_i^{pb} = O^1(i)$ for $i = 1, 2, \cdots, Ps$;
- Find the lowest fitness value in the swarm, i.e. $f_l = \min\{f_i^1\}$ for $i = 1, 2, \cdots, Ps$ with its corresponding positions X_l^1 .Set the global best to $G^1 = X_l^1$, with its fitness value $f_{gb} = f_l$.

Step 3: Update iteration counter;

- $t = t + 1$.

Step 4: Update inertia weight;

- $w^t = w_{start} - (w_{start} - w_{end}) / MaxGen * t$.

Step 5: Update velocity and position

- Use Equation (21) and Equation (22) to update velocity and position, respectively.

Step 6: Find schedule;

- Apply the SPV rule and the Assignment Algorithm to find the schedule $\pi_i^t = [\pi_{i1}^t, \pi_{i2}^t, \cdots, \pi_{i(\mathrm{n}^+ + \mathrm{n}^- + \mathrm{l} + m)}^t]$ for $i = 1, 2, \cdots, Ps$.

Step 7: Update the personal best;

- Use the schedule π_i^t to evaluate each particle again. That is, the personal best is updated as $P_i^t = X_i^t$ and $f_i^{pb} = f_i^t$ if $f_i^t < f_i^{pb}$ for $i = 1, 2, \cdots, Ps$.

Step 8: Update the global best;

- Set $f_l^t = \min\{f_i^{pb}\}$ $(l \in \{i, i = 1, 2, \cdots, Ps\})$ to obtain the minimum value of the personal best, the global best

is updated as $G^t = X_l^t$ and $f^{gb} = f_l^t$ if $f_l^t < f^{gb}$.

Step 9: Stopping criterion.

- If the current iterations exceeds *MaxGen*, stop and output computational results. Otherwise, return to step 3.

IV. COMPUTATIONAL EXPERIMENTS

In the experimental study, standard PSO (SPSO) and local PSO (LPSO) are tested compared with the real-coded genetic algorithm (GA) on YTS-SAP. The neighbor topology of LPSO is von Neumann configuration [10]. Based on the results of test runs, we set the parameters of PSOs as: $w_{start} = 0.9$, $w_{end} = 0.4$, $c_1 = c_2 = 2$. A real-coded genetic algorithm with Gaussian mutation and intermediate crossover is employed in our experiment and the involved parameters are set the same as that used in literature [11].

We coded the three algorithms in matlab7.0 and implemented it on a Intel Core i5, 2.27 GHz, 4 GB PC, under the Windows 7 operation system. The population size and the maximum number of iterations of three algorithms are set as 200 and 80, respectively, for all the experiments. Each algorithm was simulated with 10 runs for every problem.

We test the algorithms on randomly generated problems with their generating way the same as used in literature [4]. TABLE IV compares the SPSO, LPSO and GA for YTS-SAP with six different size instances. In the table, N^+, N^-, M, L are the numbers of loading jobs, discharging jobs, trucks and storage locations, respectively. The convergence rates of the three algorithms for the six problems are listed in Fig. 3.

TABLE IV. COMPARISON WITH DIFFERENT ALGORITHMS FOR SIX TEST INSTANCES

No.	(N^+, N^-, M, L)	Algorithm	Mean	Standard deviation	Aver time(s)
1	(4,4,3,8)	GA	1.98e+002	0.00	4.7
		SPSO	1.98e+002	0.00	1.4
		LPSO	1.98e+002	0.00	1.1
2	(8,7,5,14)	GA	6.18e+002	257	9.0
		SPSO	4.35e+002	167	1.4
		LPSO	4.69e+002	65	1.5
3	(15,15,10,30)	GA	2.78e+003	686	21.4
		SPSO	2.33e+003	733	2.2
		LPSO	2.32e+003	464	2.3
4	(25,25,15,50)	GA	1.05e+004	1707	39.7
		SPSO	1.00e+004	1702	3.6
		LPSO	9.35e+003	1392	3.4
5	(38,37,20,74)	GA	2.50e+004	2740	66.6
		SPSO	2.47e+004	1929	4.9
		LPSO	2.29e+004	1690	4.6
6	(50,50,30,100)	GA	2.98e+004	3996	103
		SPSO	3.08e+004	1345	6.5
		LPSO	3.27e+004	1471	6.2

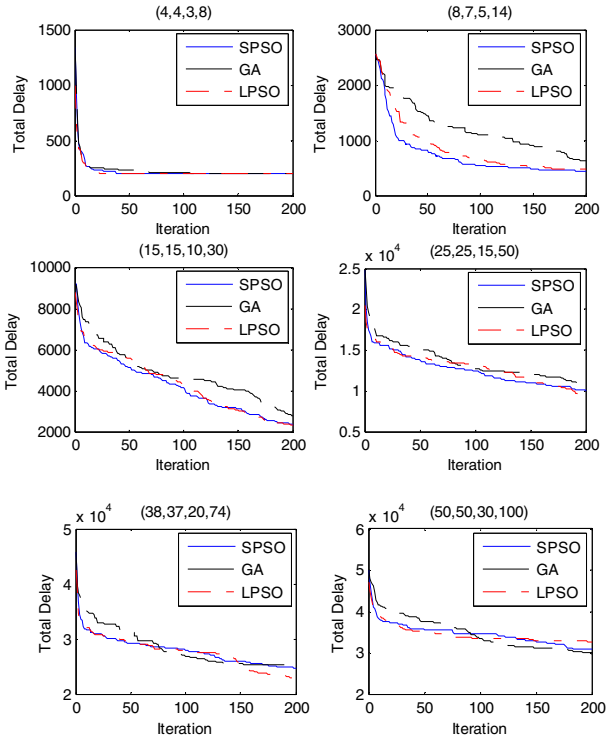

Fig. 3. The average evolution curve comparisons for different algorithms

From TABLE IV, we can see that the LPSO performs best for three out of six problems. SPSO performs best for one out of six problems, while GA only dominates the other algorithms in one problem. In other words, the solution obtained by PSOs is better than that obtained by GA for four out of six problem instances. Furthermore, we can observe that the GA algorithm get the biggest computational time for each different scale instance and the LPSO algorithm get the smallest computational time for most different scale instances except for No.2 and No.3, on which the computational time obtained by SPSO algorithm is smaller. Besides, in the experimental study with almost the same computational time, LPSO generates better solution than the SPSO solution in most instances. Furthermore, SPSO and LPSO converge faster than GA in almost all the cases, which is demonstrated in Fig. 3.

Based on the above experimental results, it can be found that with smaller computational time, PSO-based approaches are able to use its local search ability to explore good solutions, especially for LPSO approach.

V. CONCLUSION

In this article, we describe the global version PSO (SPSO) and local version PSO (LPSO) to tackle the integrated Yard Truck Scheduling and Storage Allocation Problem (YTS-SAP). Most published PSO applications are designed to solve continuous optimization problems, but litter work has been done to solve this kind of discrete optimization problems (YTS-SAP). Hence, we focus on the use of PSO to find the optimal solution of YTS-SAP. In the experimental studies, SPSO (global version of PSO) is proved to have a fast convergence rate, while with a

potential to converge to local minima. To deal with this issue, LPSO (local version of PSO) with neighbor communication strategy is used to solve YTS-SAP. We evaluated the performance of the three algorithms using six different scale instances. The mean objective value curves, together with computational results and time demonstrate the superiority of the PSO-based approaches compared with GA-based approach. In particular, we can highlight the ability of the LPSO of generating excellent average results, as well as small computational time.

ACKNOWLEDGMENT

This work is partially supported by The National Natural Science Foundation of China (Grants nos. 71001072, 71271140, 60905039), The Hong Kong Scholars Program 2012 (Grant no. G-YZ24), China Postdoctoral Science Foundation (Grant nos. 20100480705), Special Financial Grant from the China Postdoctoral Science Foundation (Grant nos. 2012T50584, 2012T50639) and the Natural Science Foundation of Guangdong Province (Grant nos. S2012010008668, S2012040007098, 9451806001002294). The authors also would like to thank The Hong Kong Polytechnic University Research Committee for financial and technical support.

REFERENCES

[1] E. K. Bish, L. Thin-Yin, L. Chung-Lun, J. W. C. Ng, and D. Simchi-Levi, "Analysis of a new vehicle scheduling and location problem," *Naval Research Logistics*, vol. 48, pp. 363-85, 2001.

[2] E. K. Bish, "A multiple-crane-constrained scheduling problem in a container terminal," *European Journal of Operational Research*, vol. 144, pp. 83-107, 2003.

[3] E. Bish, F. Chen, Y. Leong, B. Nelson, J. Ng, and D. Simchi-Levi, "Dispatching vehicles in a mega container terminal," *OR Spectrum*, vol. 27, pp. 491-506, 2005.

[4] D.H. Lee, J. X. Cao, Q. Shi, and J. H. Chen, "A heuristic algorithm for yard truck scheduling and storage allocation problems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, pp. 810-820, 2009.

[5] Z. Xue, C. Zhang, L. Miao, and W.-H. Lin, "An ant colony algorithm for yard truck scheduling and yard location assignment problems with precedence constraints," *Journal of Systems Science and Systems Engineering*, vol. 22, pp. 21-37, 2013.

[6] B. Niu,T. Xie, F.T.S. Chan, L.J. Tan and Z.X. Wang, "Particle swarm optimization for the truck scheduling in container terminals," *accepted by 2014 International Conference on Information Science,Electronics and Electrical Engineering*, April 2014.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE International Conf. Neural Networks*, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4.

[8] S. Yuhui and R. Eberhart, "A modified particle swarm optimizer," in *Proc. The 1998 IEEE International Conf. IEEE World Congress on Computational Intelligence*, 1998, pp. 69-73.

[9] M. Fatih Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem," *International Journal of Production Research*, vol. 44, pp. 4737-4754, 2006.

[10] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. CEC '02. Proceedings of the 2002 Congress. Evolutionary Computation*, 2002, pp. 1671-1676.

[11] S. Sumathi, T. Hamsapriya, and P. Surekha, *Evolutionary Intelligence: An Introduction to Theory and Applications with Matlab*: Springer Publishing Company, Incorporated, 2008.