Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 714832, 10 pages
http://dx.doi.org/10.1155/2014/714832

Research Article

An Improved Discrete PSO for Tugboat Assignment Problem under a Hybrid Scheduling Rule in Container Terminal

Su Wang,[1] Min Zhu,[1] Ikou Kaku,[2] Guoyue Chen,[2] and Mingya Wang[1]

[1]*Computer Center, East China Normal University, North Zhongshan Road, Shanghai 200062, China*
[2]*Faculty of Systems Science and Technology, Akita Prefectural University, 84-4 Tsuchiya-Ebinokuti, Yulihonjo 015-0055, Japan*

Correspondence should be addressed to Su Wang; swang@cc.ecnu.edu.cn

Received 31 July 2014; Accepted 4 December 2014; Published 28 December 2014

Academic Editor: Valder Steffen Jr.

In container terminal, tugboat plays vital role in safety of ship docking. Tugboat assignment problem under a hybrid scheduling rule (TAP-HSR) is to determine the assignment between multiple tugboats and ships and the scheduling sequence of ships to minimize the turnaround time of ships. A mixed-integer programming model and the scheduling method are described for TAP-HSR problem. Then an improved discrete PSO (IDPSO) algorithm for TAP-HSR problem is proposed to minimize the turnaround time of ships. In particular, some new redefined PSO operators and the discrete updating rules of position and velocity are developed. The experimental results show that the proposed IDPSO can get better solutions than GA and basic discrete PSO.

1. Introduction

Nowadays, more and more people have recognized the importance of world trade via container terminals. Ship docking is one important operation having an influence on the productivity of container terminal which includes two types of operations. One is the berth allocation often described as the problem in which the berthing time and the berthing position are specified to each ship [1]. The other is the tugboat assignment described as a decision problem to find the best assignment of tugboats and scheduling sequence of ships which minimizes the turnaround time of ships subject to the assignment rules. Tugboat is one kind of small ships with large engine used to move ships in and out of berth [2]. With the steadily growing size and rapidly increasing number of entry ships, safe and efficient docking operation of ships is one of the most important issues in container terminals [3]. Because container terminal has finite tugboats with different service abilities, it is very important to develop effective tugboat assignment method to reduce turnaround time of ships and maintain high productivity of container terminal.

When a ship arrives at the container terminal, the ship should be assigned an available berth for unloading and loading containers. However ships cannot dock by themselves to the berths because the port lane is narrow and the port water is shallow. The specified equipment called tugboat is used to help ships dock safely. Due to the strong structure and large power, tugboat can pull large-sized ships. If a ship is so large that one tugboat cannot tug, two or more tugboats can be used together to tow the ship. It is considered that, only with an available berth and proper tugboats, a ship can dock safely and prepare for unloading or loading operation. Otherwise, it has to wait at the anchorage outside the container terminal.

Several studies have been conducted for equipment assignment or scheduling problem in container terminals, such as berth assignment [4, 5], quay crane scheduling [6–8], yard crane scheduling [9, 10], truck scheduling [11], and storage space assignment [12]. However, little research work on tugboat assignment problem can be found. Mori [3] outlined tugboat development situation and tugboat business in Asian countries, especially in Japan. Liu and Wang introduced tugboat allocation simulation [13] and then Liu studied the tugboat scheduling problem and employed a hybrid evolutionary algorithm to solve this problem [14]. However, it only focused on the tugboat scheduling rules without mathematical model. Furthermore, the study did

not cover the optimization of the assignment plan between tugboats and ships.

Tugboat assignment is a typical kind of assignment optimization problem. The existing approaches for solving the assignment problems can be divided into two categories. The traditional methods are mathematical programming approaches using Branch and Bound [15] and Lagrangian Relaxation [16] to get the best solution efficiently, but only for small-sized problem. Besides, metaheuristic algorithms involving genetic algorithms (GA) [17–19], ant colony optimization [20, 21], simulated annealing (SA) [22], Tabu search (TS) [23, 24], and variable neighborhood search (VNS) [25] are fast and efficient algorithms to get approximate solutions with reasonable computational time for solving large scale and more difficult combination optimization problem.

Recently, particle swarm optimization (PSO) is a novel evolutionary algorithm presented by Kennedy and Eberhart [26]. Due to easy implementation and fast convergence, PSO has proved to be an effective and competitive algorithm for a wide variety of optimization problems. However, most of the published studies have concentrated on the continuous optimization problems. Recently, some work has been done to the discrete optimization problem. In order to extend the application to discrete space, Kennedy and Eberhart [27] proposed a discrete binary version of PSO where a particle moved in a state space restricted to zero and one on each dimension. Yin et al. [28] embedded a hill-climbing heuristic in PSO to assign application tasks to processors such that the resource demand of each task was satisfied and the system throughput was increased. Lin et al. [29] implemented a hybrid PSO (HPSO) to solve the biobjective personnel assignment problem. The HPSO algorithm was combined with the random-key (RK) encoding scheme and individual enhancement (IE) scheme. Similar hybrid PSO algorithms are proposed by introducing operations of GAs into PSO systems by Robinson et al. [30]. Lian et al. [31] proposed some new valid PSO operators for permutation job-shop scheduling to minimize makespan. Prescilla and Immanuel Selvakumar [32] applied the modified binary particle swarm optimization algorithm to solve the real-time task assignment in heterogeneous multiprocessor.

To the best of our knowledge, there is no published work for dealing with tugboat assignment using PSO. In this paper, we introduce the tugboat assignment problem under a hybrid scheduling rule (TAP-HSR) and give the mathematical model. Then, an improved discrete PSO algorithm for TAP-HSR problem is proposed to minimize the turnaround time of ships. In particular, a new method for redefining PSO operators is developed. The experimental results show that the proposed method is effective and efficient compared to GA.

The rest of this paper is organized as follows. In the next section, tugboat assignment problem under a hybrid scheduling rule (TAP-HSR) and the mathematical model are described. Section 3 proposes the improved discrete PSO (IDPSO) for TAP-HSR problem. Section 4 discusses the experimental results. Finally, Section 5 provides some conclusions and the future work.

2. Problem Formulation and Scheduling Rules

2.1. Problem Description. The tugboat assignment problem under a hybrid scheduling rule is an essential decision-making problem in container terminal. The tugboat assignment operation has two procedures. One is to find the assignment plan under the assignment rules. The other is to generate the scheduling sequence of ships based on the assignment plan and some scheduling rules.

Container terminal assigns tugboats to ships based on some assignment rules. Firstly, each tugboat can serve at most one ship and each ship can be assigned one or more ships at any time. Secondly, the assigned number of tugboats is determined by ship length. The ship with length equal to or less than 100 meters needs one tugboat at least. Otherwise, the ship with the length more than 100 meters needs two tugboats at least. Finally, it is required to assign tugboat with proper horsepower (ps). Horsepower is a common unit to measure tugboat power. The tugboat with higher horsepower has more service capacity to tug longer ship. The assignment rules between ships and tugboats are shown in Table 1.

Assignment plan only describes the matching relationship of multiple tugboats and multiple ships. It is required to use some scheduling rules to determine a scheduling sequence of ships according to the assignment plan. This paper applies a hybrid scheduling rule combined with first-come-first-served (FCFS) rule and first-fit rule. Based on the hybrid scheduling rule, the ship with earlier arriving time and the available assigned tugboats has the higher priority of docking.

The goal of TAP-HSR is to minimize the turnaround time of ships under the premise that all ships finish docking. With more and more ships arriving, it is impossible to make sure that ships can have available tugboats for docking immediately at arrival. Some ships will wait outside the terminal at the anchorage until available tugboats can work for them. Good assignment can decrease the waiting time of ships so that the container terminal can serve more ships and get more profits.

2.2. Model Formulation. In this paper, TAP-HSR is based on the following assumptions.

(1) The expected arriving time and the length of all ships are known.

(2) The docking operation time of ship depends on the length of ship and is independent of tugboat.

(3) Each ship has an available berth for unloading and loading containers.

(4) Tugboat can work for only one ship at the same time.

(5) The preparation time of tugboat is zero which means one tugboat can start to serve the next ship immediately after it finishes the previous ship.

(6) All tugboats are available when the first ship starts to berth.

(7) The operation of tugboat requires an uninterrupted period of time.

Table 1: Assignment rules between ships and tugboats.

Length of ship (m)	Required horsepower (ps) of tugboats	Required number of tugboats	Docking operation time (min)
0–100	⩾2600	⩾1	40
100–200	⩾5200	⩾2	48
200–250	⩾6400	⩾2	60
250–300	⩾6800	⩾2	75
⩾300	⩾8000	⩾2	85

This paper describes TAP-HSR problem with the following notations.

m: the number of tugboats

n: the number of ships

J_j: the ship j, $j \in \{1, 2, 3, \ldots, n\}$

h_i: the horsepower of tugboat i

p_j: the state of tugboat i ($p_j = 0$ denotes tugboat i is available. Otherwise, $p_j = 1$)

q_j: the state of ship j ($q_j = 1$ denotes ship j finished docking. Otherwise, $q_j = 0$)

o_j: the scheduling order of ship j

l_j: the length of ship j

h_i: the horsepower of tugboat i. $i \in \{1, 2, 3, \ldots, m\}$

t_j: the operation time of ship j

I_j: the arriving time of ship j

S_j: the start docking time of ship j

T_j: the completion of docking time of ship j

H_j: the least required horsepower of ship j

R_j: the least required tugboat number of ship j

x_{ij}: decision variable denotes tugboat i assigned to ship j.

This paper gives the following definitions.

Definition 1 (assignment matrix). X_{mn} is the decision matrix which describes the assignment between ships and tugboats. The decision variable $x_{ij} \in \{0, 1\}$ denotes whether or not tugboat i is assigned to ship j:

$$X_{mn} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}, \quad (1)$$

$$x_{ij} = \begin{cases} 1 & \text{Tugboat } i \text{ is assigned to ship } j \\ 0 & \text{Otherwise.} \end{cases}$$

Definition 2 (conflict ship). If tugboat i is assigned to both ship j and ship k and ship j is scheduled for docking before ship k, ship j is called the conflict ship of ship k. Let C_k denote the conflict ship set of ship k:

$$C_k = \left\{ J_j \mid x_{ij} \cdot x_{ik} = 1,\ o_j < o_k \right\}. \quad (2)$$

$C_k = \varnothing$ represents that all of the assigned tugboats to ship k are available when ship k arrives. In that condition, the ship can dock immediately at arriving time. Otherwise, if the assigned tugboats are working for the conflict ships of ship k, ship k has to wait until all the conflict ships finish docking.

Let T_S denote the start docking time of the first ship and let T_F denote the completion docking time of the last ship. The objective function of TAP-HSR problem is to minimize the turnaround time of ships described as

$$\min \left(T_F - T_S \right), \quad (3)$$

where $T_S = 0$ according to assumption (6) and T_F is the maximum completion docking time of ships defined as

$$T_F = \max_{j=1}^{n} T_j. \quad (4)$$

The objective function is $\min T_F$ represented by the following formula:

$$\min \max_{j=1}^{n} T_j. \quad (5)$$

To find the maximum completion docking time of ships, it is required to calculate the completion docking time of each ship T_j ($j = 1, 2, 3, \ldots, n$) represented as the start docking time of ship S_j plus the operation time of ship t_j:

$$T_j = S_j + t_j. \quad (6)$$

In TAP-HSR problem, let J_j denote ship j. According to Definition 2, the maximum completion time of the conflict ships of J_j is $\max_{J_k \in C_j} T_k$. The start docking time S_j is calculated by

$$S_j = \max \left\{ I_j, \max_{J_k \in C_j} T_k \right\}. \quad (7)$$

Equation (7) states that if ship arrives after the time when all the conflict ships have done the docking operation, the start docking time is the arriving time. Otherwise, the start docking time of this ship is the maximum completion docking time of the conflict ships.

The constraints in TAP-HSR problem are as follows:

$$\sum_{i=1}^{m} x_{ij} \geq R_j, \qquad \sum_{i=1}^{m} x_{ij} h_i \geq H_j. \quad (8)$$

```
While (k < n)
  Set the state of each tugboat p_i = 0
  For j = 1 to n
    If (q_j = 0 and the state of each assigned tugboat p_i = 0)
      o_j = o_flag
      Set the state of each assigned tugboat p_i = 1
      Calculate the conflict ship set c_j
      Calculate the start docking time s_j according to (7)
      Calculate the completion docking time T_j according to (6)
      Set the state of ship j q_j = 1
      k = k + 1
    End if
  Next j
  o_flag = o_flag + 1
End
Calculate the turnaround time of ships according to (4).
```

Pseudocode 1

Constraints in (8) ensure that the total number and horsepower of tugboats assigned to each ship should meet the assignment rules to make sure that the tugboats have enough horsepower to tug the ship.

2.3. The Scheduling Method. The assignment matrix only describes the assignment plan between ships and tugboats. It is required to generate the scheduling sequence of ships based on the assignment matrix and scheduling rules. The scheduling sequence is generated as in the following steps.

Step 1. Order the ships by the arriving time. Initialize $p_i = 0$, $q_j = 0$, the number of finished ships $k = 0$, and the initial scheduling order $o\_flag = 1$.

Step 2. Check the assignment matrix whether it is a feasible solution which satisfies constraints in (8). If it is a feasible solution, go to Step 3.

Step 3. Find the conflict ships of each ship and generate the scheduling sequence based on first-fit rule. The pseudocode of generating scheduling sequence is described in Pseudocode 1.

Step 4. Output the scheduling sequence, the start docking time and completion docking time of each ship, and the turnaround time of ships.

We give one example of scheduling ships based on assignment plan. Table 2 shows the tugboat allocation. Table 3 shows the arriving ships.

One assignment matrix between tugboats and ships is as follows:

$$X_i = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ J_1 & J_2 & J_3 & J_4 & J_5 \end{pmatrix}. \quad (9)$$

Table 2: One example of tugboat allocation.

Tugboat number	Horsepower (ps)	Number
1	2600	1
2	3200	1
3	3400	1
4	4000	1

Table 3: One example of arriving ships.

Ship number	Length (m)	Arriving time (min)	Operation time (min)
1	87	0	40
2	245	5	60
3	286	10	75
4	93	15	40
5	202	20	60

X_i satisfies constraints in (8) so that it is a feasible assignment. According to the scheduling method, the scheduling sequence $O = (J_1, J_3, J_4, J_2, J_5)$ and turnaround time of ships is 135 minutes. The scheduling Gant graph is as in Figure 1.

3. Discrete PSO Algorithm for TAP-HSR Problem

TAP-HSR is in one kind of NP-hard problem and difficult for human schedulers. The goal of our research is to achieve as good as possible results in reasonable computational time.

3.1. Standard PSO Algorithm. Swarm optimization algorithm is a swarm intelligence algorithm proposed by Kennedy and Eberhart [26] as a simulation of the social behavior of bird flocking. PSO is composed of a number of individuals called particles. Particles are initialized with random locations and velocities. Each particle tries to improve the location by

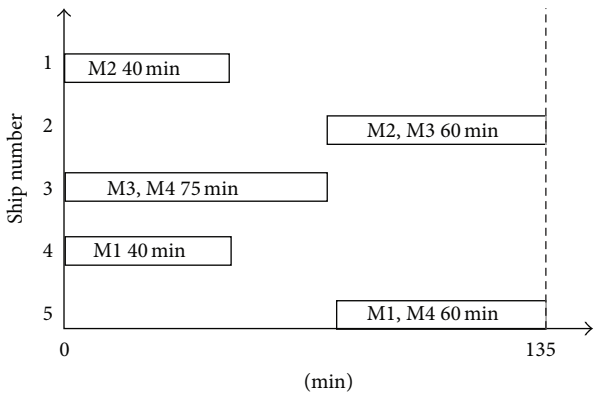

Figure 1: The scheduling Gant graph.

following the best location of itself and the best particle of the whole swarm. The position of ith particle is represented by M-dimensional vector denoted by $X_i = (x_{i1}, x_{i2}, \ldots, x_{iM})$ $(i = 1, 2, \ldots, k)$. The velocity for the ith particle can be described as $V_i = (v_{i1}, v_{i2}, \ldots, v_{iM})$ $(i = 1, 2, \ldots, k)$.

A particle's movement is based on (10):

$$v_i^{t+1} = \omega \times v_i^t + c_1 \times \text{rand}_1(\,) \times \left(p_i^t - x_i^t\right) + c_2 \times \text{rand}_2(\,) \times \left(p_g^t - x_i^t\right), \tag{10}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{11}$$

where ω is called inertia weight [33], c_1 and c_2 are two constant numbers which are often called social or cognitive confidence coefficients, the functions $\text{rand}_1(\,)$ and $\text{rand}_2(\,)$ can generate a random number in $[0, 1]$, x_i^t is the position of ith particle at the iteration t, v_i^t is the velocity of ith particle at the iteration t, p_i^t is the local best position that ith particle has reached at the iteration t, and p_g^t is the global best position that all the particles have reached at the iteration t.

3.2. The Improved Discrete PSO. Because of the continuous position space of particles, standard encoding method of PSO cannot be directly adopted for discrete optimization problems. It is a challenge to employ the algorithm in combinatorial optimization problems, especially for assignment problem. In this paper, we exploit a new redefining PSO operator method and discrete updating rules of position and velocity to get the solutions for TAP-HSR problem.

3.2.1. Discrete Particle Representation. The most important problem in applying PSO to TAP-HSR problem is to find a suitable particle representation method. In assignment problem, each particle represents a candidate assignment matrix with $m \times n$ elements [34]. Each element is an integer value in $\{0, 1\}$. For example, in (12) X_i denotes one assignment that assigns four tugboats to five ships. Tugboat 1 is assigned to ship 1. Tugboats 2 and 3 are assigned to ship 2. Tugboats 1 and 2 are assigned to ship 3. Tugboat 4 is assigned to ship 4. Tugboats 3 and 4 are assigned to ship 5.

One example of particle represented by assignment matrix is as follows:

$$X_i = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{12}$$

3.2.2. Operators Redefinition

Definition 3. The position subtraction operator $X_i \boxminus X_j$: the subtraction of two positions can be defined as XOR (exclusive OR) operation of binary matrix:

$$X_i \boxminus X_j = \left(\neg X_i \wedge X_j\right) \vee \left(X_i \wedge \neg X_j\right). \tag{13}$$

Equation (14) gives one example of the subtraction of positions X_i^t and X_i^{t+1}.

$$X_i^t = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

$$X_i^{t+1} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \tag{14}$$

$$X_i^{t+1} \boxminus X_i^t = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

It can be seen that the subtraction of positions is to find the different elements between two positions [34]. According to (11), the velocity is the subtraction of positions that can be represented by a set of the numerical pairs. $(v_j)_{\text{length}}$ denotes the element number of v_j:

$$v_j = \{(x, y) \mid x \in \{1, 2, \ldots, M\}, y \in \{1, 2, \ldots, N\}\}. \tag{15}$$

In (12), $v_i^{t+1} = X_i^{t+1} \boxminus X_i^t = \{(2, 3), (3, 2), (3, 3), (4, 2)\}$. $(v_i^{t+1})_{\text{length}} = 4$.

Definition 4. The velocity addition operator $v_i \boxplus v_j$: the addition of velocities is considered as a union of the elements in v_i or in v_j:

$$v_i \boxplus v_j = \left\{(x, y) \mid (x, y) \in v_i \text{ or } (x, y) \in v_j\right\}. \tag{16}$$

Definition 5. The velocity subtraction operator $v_i \boxminus v_j$: the subtraction of velocities is considered as the relative complement elements of v_j with respect to v_i. $v_i \boxminus v_j$ is denoted by the set of elements in v_i but not in v_j:

$$v_i \boxminus v_j = \left\{(x, y) \mid (x, y) \in v_i, \ (x, y) \notin v_j\right\}. \tag{17}$$

Table 4: Examples of $c \boxtimes v_2$ and $v_1 \boxplus (c \boxtimes v_2)$.

c	a	b	$c \boxtimes v_2$	$v_1 \boxplus (c \boxtimes v_2)$
$c \in (0, 0.5]$	1	3	$\|v_2\|_1^3 = \{(2,3),(3,2),(3,3)\}$	$v_1 \boxplus \|v_2\|_1^3 = \{(1,5),(2,3),(3,2),(3,3)\}$
	2	3	$\|v_2\|_2^3 = \{(3,2),(3,3)\}$	$v_1 \boxplus \|v_2\|_2^3 = \{(1,5),(2,3),(3,2),(3,3)\}$
	4	4	$\|v_2\|_4^4 = \{(4,2)\}$	$v_1 \boxplus \|v_2\|_4^4 = \{(1,5),(3,2),(2,3),(4,2)\}$
$c \in (0.5, 1)$	1	3	$\boxminus \|v_2\|_1^3 = \{(2,3),(3,2),(3,3)\}$	$v_1 \boxminus \|v_2\|_1^3 = \{(1,5)\}$
	2	3	$\boxminus \|v_2\|_2^3 = \{(3,2),(3,3)\}$	$v_1 \boxminus \|v_2\|_2^3 = \{(1,5),(2,3)\}$
	4	4	$\boxminus \|v_2\|_4^4 = \{(4,2)\}$	$v_1 \boxminus \|v_2\|_4^4 = \{(1,5),(3,2),(2,3)\}$

One example of the addition and subtraction of two velocities is as follows:

$$\begin{aligned} v_i &= \{(2,3),(3,2),(3,3),(4,2)\}, \\ v_j &= \{(1,5),(3,2)\}, \\ v_i \boxplus v_j &= \{(2,3),(3,2),(3,3),(4,2),(1,5)\}, \\ v_i \boxminus v_j &= \{(2,3),(3,3),(4,2)\}. \end{aligned} \tag{18}$$

Definition 6. The multiply operator of coefficient and velocity $c \boxtimes v$: let c be coefficient and let v be velocity. c is a random real number in $(0, 1)$:

$$c \boxtimes v = {}_c\|v_2\|_a^b = \begin{cases} \|v\|_a^b & c \in (0, 0.5] \\ \boxminus \|v\|_a^b & c \in (0.5, 1), \end{cases} \tag{19}$$

where a and b are random integers and $a, b \in [1, (v)_{\text{length}}]$. $\|v\|_a^b$ represents a subelement string of v_2 from the ath element to the bth element in (20). If $a = 1, b = (v)_{\text{length}}$, $\|v\|_1^{(v)_{\text{length}}} = v$.

Examples of $\|v\|_a^b$ are as follows:

$$\begin{aligned} v \quad & (2,3),(3,2),(3,3),(4,2) \\ \|v\|_1^3 \quad & |_1 (2,3),(3,2),(3,3),|_3 (4,2) \\ & (2,3),(3,2),(3,3) \\ \|v\|_2^3 \quad & (2,3),|_2 (3,2),(3,3)|_3 (4,2) \\ & (3,2),(3,3) \\ \|v\|_1^4 \quad & |_1 (2,3),(3,2),(3,3),(4,2)|_4 \\ & (2,3),(3,2),(3,3),(4,2). \end{aligned} \tag{20}$$

Based on (19), $v_1 \boxplus (c \boxtimes v_2)$ can be represented by

$$v_1 \boxplus (c \boxtimes v_2) = \begin{cases} v_1 \boxplus \|v_2\|_a^b & c \in (0, 0.5] \\ v_1 \boxminus \|v_2\|_a^b & c \in (0.5, 1). \end{cases} \tag{21}$$

For illustration, $v_1 = \{(1,5),(3,2),(2,3)\}$ and $v_2 = \{(2,3),(3,2),(3,3),(4,2)\}$. Table 4 shows some examples of $c \boxtimes v_2$ and $v_1 \boxplus (c \boxtimes v_2)$.

From Table 4, it is known that if $c \in (0, 0.5]$, $v_1 \boxplus (c \boxtimes v_2)$ can be added to some new elements. If $c \in (0.5, 1)$, some elements might be deleted from v_2. This method can keep the diversity of the swarm and exploit the global search ability of PSO thoroughly.

Definition 7. The multiply operator of inertia weight and velocity $\omega \boxtimes v$: let $\omega = 1$ and let $\omega \boxtimes v = v$.

Based on Definitions 3–7, the new discrete movements of particles are followed by the following equations:

$$v_i^{t+1} = v_i^t \boxplus {}_{c1}\left\|p_i^t \boxminus x_i^t\right\|_{a1}^{b1} \boxplus {}_{c2}\left\|p_g^t \boxminus x_i^t\right\|_{a2}^{b2} \tag{22}$$

$$x_i^{t+1} = x_i^t \boxplus v_i^{t+1}. \tag{23}$$

3.2.3. The Procedure of IDPSO for TAP-HSR. The pseudocode of the improved discrete PSO is listed in Pseudocode 2.

4. Experimental Results

In this section, we present the computational experiments to evaluate the performance of the improved discrete PSO algorithm in this paper. The algorithms are coded in Java language with the environment of Eclipse and simulated on a PC with a 2.80 GHz Intel CPU and 4 G RAM. In the experiments, three algorithms are implemented for TAP-HSR problems, including GA, the basic discrete PSO (BDPSO), and the improved discrete PSO (IDPSO) in Section 3. The basic discrete PSO is implemented without the multiple operator of coefficient and velocity $c \boxtimes v$. The movements of particles are followed by the following equations:

$$\begin{aligned} v_i^{t+1} &= v_i^t \boxplus \left(p_i^t \boxminus x_i^t\right) \boxplus \left(p_g^t \boxminus x_i^t\right) \\ x_i^{t+1} &= x_i^t \boxplus v_i^{t+1}. \end{aligned} \tag{24}$$

In IDPSO, there are two important parameters: the max iteration $NC_{\max}$ and swarm size K. Through the experiments, it can be seen that when the max iteration is fixed, the optimal solution is improved with the growth of swarm size because more particles have stronger space search capabilities. However, if swarm size is excessive, the optimal solution quality can be decreased so that it is required to increase the max iteration to get a better solution. The reason is that too many particles cause the solution dispersion so that it is difficult to converge to an optimal solution. On the other hand, the program running time will increase with the increasing of the max iteration and swarm size. Therefore, it is important to select the proper max iteration and swarm size according to the problem size shown in Table 5.

Step 1. (Initializing)
Initialize the parameters: the max iteration $NC_{\max}$, the no-improvement iteration $GC_{\max}$, swarm size K, each particle's location and velocity, the local best location p_i and the global best location p_g.
Step 2. (Updating and Fitness Calculation)
Set the iteration number $t = 1$
Repeat
  For each particle, generate random parameter: $c1$, $c2$, $a1$, $a2$, $b1$, $b2$
  Update the velocity of each particle according to (22)
  Update the position of each particle according to (23)
  Check the new position X_i^{t+1} is a feasible assignment according to (8)
  If (X_i^{t+1} is not a feasible assignment) then
    Mutation X_i^{t+1} to a feasible assignment
  End if
  Generate the scheduling sequence
  Calculate the turnaround time as fitness of X_i^{t+1}
  Update p_i, p_g
  Check the improvement of p_g
  If p_g is improved then
    $c\_flag = 0$
  Else
    $c\_flag = c\_flag + 1$
  End if
  $t = t + 1$
  Untill $t > NC_{\max}$ or $c\_flag = GC_{\max}$
Step 3. (Final stage)
Output the best tugboat assignment, the ship scheduling sequence, the start docking time and the completion docking time of eachship, the turnaround time of the best assignment.

Pseudocode 2

Table 5: The max iteration $NC_{\max}$ and swarm size K.

Problem size	$NC_{\max}$	K
Small-sized problem	1000–2500	50–150
Middle-sized problem	2500–3500	150–250

To compare the performance of IDPSO, BDPSO, and GA, the experiments consisted of 12 small-sized problems and 10 middle-sized problems that are conducted. In both IDPSO and BDPSO, we use a swarm of 100 particles for small-sized experiments. The max iteration is 2000 and the no-improvement iteration is 1000. In middle-sized experiment, we use a swarm of 200 particles. The max iteration is 3000 and the no-improvement iteration is 1500. In GA, population size is 100. The max iteration is 3000 and the no-improvement iteration is 1500. The probability of crossover is 0.8. The probability of mutation is 0.01. For the three algorithms, two stopping rules are adopted. One is max-iteration rule and the other is no-improvement rule [34]. The comparative experimental results obtained by the IDPSO, BDPSO, and GA on the different size problems are shown in Tables 6 and 7. Since PSO and GA algorithms are stochastic algorithms, each separate run of the program could result in a different result. We run each algorithm 20 times for each problem case. The values of $f_{b\_\min}$ represent the minimum turnaround time of 20 runs. The values of $f_{\text{avg\_min}}$ are the average turnaround time of the 20 runs. The values of mean time are the computational time of average computational time.

Through this comparison it is observed that the quality of solutions obtained by IDPSO is always better than that obtained using BDPSO and GA in terms of the best turnaround time and the average turnaround time. Furthermore, the improvement of solutions actually becomes greater as the size of the problems increases. The results obtained by GA are bit better than BDPSO in most of the instances. The proposed IDPSO can obtain the optimal or near-optimal solutions in all cases.

As far as computational efficiency is concerned, it is worth mentioning that GA requires less time when compared to BDPSO and IDPSO. On the other hand, IDPSO reports solutions for all problem sizes within reasonable time. Therefore, IDPSO is recommended for use in TAP-HSR problem, as it yields satisfactory results with reasonable computational time. All these experiments showed that the improved discrete PSO in this paper is an effective mechanism for solving the tugboat assignment in container terminals.

5. Conclusions

Although there is a huge amount of literature on discrete PSO, the PSO algorithm for tugboat assignment problem does not have rich literature. In this paper, we present a mathematic model of tugboat assignment problem under a hybrid scheduling rule to minimize the turnaround time of

Table 6: The compared results for small-sized problem.

Problem size (ship × tugboat)	Tugboats 2600	3200	3400	4000	Algorithm	f_{b_min} (min.)	f_{avg_min} (min.)	Mean time (s)
5 × 4	1	1	1	1	IDPSO	135	135	2
					BDPSO	175	183	3
					GA	135	159	2
5 × 7	1	2	2	2	IDPSO	80	80	5
					BDPSO	120	136.75	7
					GA	80	110.75	2
6 × 4	1	1	1	1	IDPSO	175	175	2
					BDPSO	183	213.2	4
					GA	175	181	3
6 × 7	1	2	2	2	IDPSO	108	110.8	7
					BDPSO	123	161.3	9
					GA	108	131.3	3
7 × 6	2	2	1	1	IDPSO	168	182.8	7
					BDPSO	195	238.7	9
					GA	168	194.15	3
7 × 6	1	1	2	2	IDPSO	160	172.35	7
					BDPSO	185	229.95	9
					GA	160	183.65	3
7 × 8	2	2	2	2	IDPSO	125	140.75	10
					BDPSO	200	227.35	14
					GA	135	174.9	3
8 × 6	2	2	1	1	IDPSO	173	197.4	8
					BDPSO	193	251.5	8
					GA	193	217.8	4
8 × 6	1	1	2	2	IDPSO	160	185.6	8
					BDPSO	193	236.85	8
					GA	185	208.3	4
8 × 8	2	2	2	2	IDPSO	135	156.3	12
					BDPSO	248	282.5	25
					GA	160	197.45	4
10 × 10	3	3	2	2	IDPSO	208	237.85	26
					BDPSO	308	363.9	45
					GA	216	279.25	6
10 × 10	2	2	3	3	IDPSO	185	218.65	25
					BDPSO	305	362.2	46
					GA	245	290.7	6

ships. Moreover, the improved discrete PSO is proposed to get solutions with good quality in a reasonable computational time. Some new redefined PSO operators and discrete updating rules of position and velocity are described in detail. The proposed IDPSO algorithm in this paper can be considered as more effective approach for TAP-HSR problem compared with GA.

The proposed IDPSO in this paper for small-sized and middle-sized TAP-HSR is very effective but for large-size it is not approving. It needs to be improved in the future work such as combining heuristic rule with IDPSO. Furthermore, applying IDPSO to other discrete combinatorial optimization problems is also possible in further research.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Table 7: The compared results for middle-sized problem.

Problem size (ship × tugboat)	Tugboats 2600	3200	3400	4000	Algorithm	f_{b_min} (min.)	f_{avg_min} (min.)	Mean time (s)
15 × 10	3	3	2	2	IDPSO	375	412.4	47
					BDPSO	528	621.55	65
					GA	446	509.4	22
15 × 10	2	2	3	3	IDPSO	330	398.35	47
					BDPSO	509	596.7	60
					GA	428	495.45	22
15 × 12	3	3	3	3	IDPSO	343	409.1	70
					BDPSO	523	601	89
					GA	393	515.85	28
20 × 10	2	2	3	3	IDPSO	498	547.2	79
					BDPSO	701	795.25	120
					GA	596	695.4	41
20 × 12	3	3	3	3	IDPSO	475	567.2	119
					BDPSO	613	757.2	234
					GA	588	767.65	54
30 × 12	3	3	3	3	IDPSO	959	1070.7	321
					BDPSO	1257	1427.5	828
					GA	1265	1433.4	115
30 × 15	3	4	4	4	IDPSO	858	970	486
					BDPSO	1144	1303.1	1095
					GA	1026	1243.2	151
50 × 12	3	3	3	3	IDPSO	1832	1957.85	1110
					BDPSO	2276	2424	2670
					GA	2167	2491.35	326
50 × 15	3	4	4	4	IDPSO	1683	1814.8	1678
					BDPSO	2119	2245.65	5408
					GA	2089	2252	419
80 × 15	3	4	4	4	IDPSO	3179	3387.6	5870
					BDPSO	3862	4045.0	13247
					GA	3834	4064.15	973

Acknowledgment

This work is supported by the National High Technology Research and Development Program of China (863 Program no. 2013AA01A211).

References

[1] E. Nishimura, A. Imai, and S. Papadimitriou, "Berth allocation planning in the public berth system by genetic algorithms," *European Journal of Operational Research*, vol. 131, no. 2, pp. 282–292, 2001.

[2] http://www.search.com/reference/TUGb.

[3] T. Mori, "The present situation and the issues on tugboat business in Japan," http://www.h2.dion.ne.jp/~t-mori/ronbun.html.

[4] C. Bierwirth and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals," *European Journal of Operational Research*, vol. 202, no. 3, pp. 615–627, 2010.

[5] M. H. Elwany, I. Ali, and Y. Abouelseoud, "A heuristics-based solution to the continuous berth allocation and crane assignment problem," *Alexandria Engineering Journal*, vol. 52, no. 4, pp. 671–677, 2013.

[6] K. H. Kim and Y.-M. Park, "A crane scheduling method for port container terminals," *European Journal of Operational Research*, vol. 156, no. 3, pp. 752–768, 2004.

[7] Y.-M. Fu, A. Diabat, and I.-T. Tsaim, "A multi-vessel quay crane assignment and scheduling problem: formulation and heuristic solution approach," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6959–6965, 2014.

[8] A. Diabat and E. Theodorou, "An integrated quay crane assignment and scheduling problem," *Computers and Industrial Engineering*, vol. 73, pp. 115–123, 2014.

[9] W. C. Ng, "Crane scheduling in container yards with inter-crane interference," *European Journal of Operational Research*, vol. 164, no. 1, pp. 64–78, 2005.

[10] M. E. H. Petering and K. G. Murty, "Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal," *Computers & Operations Research*, vol. 36, no. 5, pp. 1711–1725, 2009.

[11] V. D. Nguyen and K. H. Kim, "A dispatching method for automated lifting vehicles in automated port container terminals," *Computers & Industrial Engineering*, vol. 56, no. 3, pp. 1002–1020, 2009.

[12] M. Bazzazi, N. Safaei, and N. Javadian, "A genetic algorithm to solve the storage space allocation problem in a container terminal," *Computers & Industrial Engineering*, vol. 56, no. 1, pp. 44–52, 2009.

[13] Z.-X. Liu and S.-M. Wang, "The computer simulation study of port tugboat operation," *Journal of System Simulation*, vol. 16, no. 1, pp. 45–48, 2004.

[14] Z.-X. Liu, "Hybrid evolutionary strategy optimization for port tugboat operation scheduling," in *Proceedings of the 3rd International Symposium on Intelligent Information Technology Application (IITA '09)*, pp. 511–515, November 2009.

[15] W. C. Ng and K. L. Mak, "Yard crane scheduling in port container terminals," *Applied Mathematical Modelling*, vol. 29, no. 3, pp. 263–276, 2005.

[16] A. Imai, E. Nishimura, and S. Papadimitriou, "The dynamic berth allocation problem for a container port," *Transportation Research B: Methodological*, vol. 35, no. 4, pp. 401–417, 2001.

[17] F. Li, L. D. Xu, C. Jin, and H. Wang, "Random assignment method based on genetic algorithms and its application in resource allocation," *Expert Systems with Applications*, vol. 39, no. 15, pp. 12213–12219, 2012.

[18] R. K. Ahuja, J. B. Orlin, and A. Tiwari, "A greedy genetic algorithm for the quadratic assignment problem," *Computers and Operations Research*, vol. 27, no. 10, pp. 917–934, 2000.

[19] U. Tosun, "A new recombination operator for the genetic algorithm solution of the quadratic assignment problem," *Procedia Computer Science*, vol. 32, pp. 29–36, 2014.

[20] N. Ç. Demirel and M. D. Toksarı, "Optimization of the quadratic assignment problem using an ant colony algorithm," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 427–435, 2006.

[21] C. Özkale and A. Fığlalı, "Evaluation of the multiobjective ant colony algorithm performances on biobjective quadratic assignment problems," *Applied Mathematical Modelling*, vol. 37, no. 14-15, pp. 7822–7838, 2013.

[22] Y. Hamam and K. S. Hindi, "Assignment of program modules to processors: a simulated annealing approach," *European Journal of Operational Research*, vol. 122, no. 2, pp. 509–513, 2000.

[23] M. S. Hussin and T. Stützle, "Tabu search versus simulated annealing as a function of the size of quadratic assignment problem instances," *Computers & Operations Research*, vol. 43, pp. 286–291, 2014.

[24] J. Skorin-kapov, "Tabu search applied to quadratic assignment problem," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 33–40, 1990.

[25] Y. Kuo, "Optimizing truck sequencing and truck dock assignment in a cross docking system," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5532–5541, 2013.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[27] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.

[28] P.-Y. Yin, S.-S. Yu, P.-P. Wang, and Y.-T. Wang, "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems," *Computer Standards & Interfaces*, vol. 28, no. 4, pp. 441–450, 2006.

[29] S.-Y. Lin, S.-J. Horng, T.-W. Kao et al., "An efficient bi-objective personnel assignment algorithm based on a hybrid particle swarm optimization model," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7825–7830, 2010.

[30] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna," in *Proceedings of the IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Conference*, pp. 168–175, June 2002.

[31] Z. Lian, B. Jiao, and X. Gu, "A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan," *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 1008–1017, 2006.

[32] K. Prescilla and A. Immanuel Selvakumar, "Modified binary particle swarm optimization algorithm application to real-time task assignment in heterogeneous multiprocessor," *Microprocessors and Microsystems*, vol. 37, no. 6-7, pp. 583–589, 2013.

[33] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE World Congress on Computational Intelligence Conference*, pp. 69–73, May 1998.

[34] Y. Han, J. Tang, I. Kaku, and L. Mu, "Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight," *Computers and Mathematics with Applications*, vol. 57, no. 11-12, pp. 1748–1755, 2009.