Advanced Engineering Informatics 29 (2015) 59–75

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Yard crane scheduling in a container terminal for the trade-off between efficiency and energy consumption

Junliang He *, Youfang Huang, Wei Yan

Engineering Research Center of Container Supply Chain Technology, Ministry of Education, Shanghai Maritime University, Shanghai 201306, PR China

ARTICLE INFO

Article history:
Received 18 March 2014
Received in revised form 8 September 2014
Accepted 10 September 2014
Available online 29 September 2014

Keywords:
Yard crane scheduling
Energy consumption
Vehicle routing problem
Mixed integer programming
Simulation optimization
Hybrid algorithm

ABSTRACT

Green transportation has recently been the focus of the transportation industry to sustain the development of global economy. Container terminals are key nodes in the global transportation network and energy-saving is a main goal for them. Yard crane (YC), as one type of handling equipment, plays an important role in the service efficiency and energy-saving of container terminals. However, traditional methods of YC scheduling solely aim to improve the efficiency of container terminals and do not refer to energy-saving. Therefore, it is imperative to seek an appropriate approach for YC scheduling that considers the trade-off between efficiency and energy consumption. In this paper, the YC scheduling problem is firstly converted into a vehicle routing problem with soft time windows (VRPSTW). This problem is formulated as a mixed integer programming (MIP) model, whose two objectives minimize the total completion delay of all task groups and the total energy consumption of all YCs. Subsequently, an integrated simulation optimization method is developed for solving the problem, where the simulation is designed for evaluating solutions and the optimization algorithm is designed for exploring the solution space. The optimization algorithm integrates the genetic algorithm (GA) and the particle swarm optimization (PSO) algorithm, where the GA is used for global search and the PSO is used for local search. Finally, computational experiments are conducted to validate the performance of the proposed method.

1. Introduction

The rapidly increasing concentration of pollutants in the environment has become a world-wide concern. Climate change due to carbon dioxide emissions from transportation is considered to be a significant environmental threat [1–3]. Not only do we have climate problems but we are also dealing with a resource depletion issue, especially energy resources, like oil, gas and coal. As a result, green transportation has recently been the focus of the transportation industry to sustain the development of global economy. Container terminals are key nodes in the global transportation network. Because of the considerable number of pieces of large handling equipment, container terminals play a significant role in energy consumption and pollutant emission in the entire container transportation network. Therefore, the operators of container terminals are facing the pressure on energy-saving and emission reduction. Accompanied with the world economic crisis, the competition among container terminals is getting fiercer and fiercer. Container terminals have to improve their service efficiency to attract more customers in the fierce competition environment. At the same time, the improvement of service efficiency cannot be implemented at the expense of the environment. Thus, the goal of container terminals at the operational level is to seek the optimal trade-off between energy-saving and service efficiency improvement. Since the energy consumption and service efficiency of container terminals are mainly contributed by the handling equipment, the scheduling of the handling equipment is critical.

The total energy consumption cost of the handling equipment of a container terminal is between 15% and 25% of the total cost of operations [4]. Yard crane (YC), as the most frequently-used handling equipment in a container yard, accounts for approximately 25–35% of the total energy consumption cost [5]. For example, the energy consumption of a container terminal in Tianjin port in 2013 was around 47,000,000 kW h for handling 2,320,000 twenty-foot equivalent units (TEUs), in which around 14,100,000 kW h was consumed by YCs.

YCs mainly perform four types of storage and retrieval operations, i.e., (1) storing import containers from vessels, denoted as SI; (2) retrieving import containers for carriers, denoted as RI; (3) storing export containers from carriers, denoted as SE; and (4) retrieving export containers for vessels, denoted as RE [6]. YC

* Corresponding author at: 1550 Haigang Avenue, Lingang New Port City, School of Logistics Engineering, Shanghai Maritime University, Shanghai 201306, PR China. Tel.: +86 21 3828 2674; fax: +86 21 3828 2673.
E-mail address: soldierlianglian@gmail.com (J. He).

http://dx.doi.org/10.1016/j.aei.2014.09.003
1474-0346/

scheduling directly impacts the completion time of these operations and the total energy consumption of all YCs. As the importance of YC scheduling on service efficiency has been noticed for a long time, a number of studies have developed new operational approaches that can solely improve the efficiency of container terminals [7]. However, traditional methods of container terminal operations generally do not take into consideration energy-saving at the operational level. Thereby, this paper seeks an appropriate approach for the YC scheduling problem by considering the optimal trade-off between service efficiency and energy consumption. This paper is organized as follows. Section 2 reviews relevant literature. Section 3 describes our problem and converts it into a vehicle routing problem with soft time windows (VRPSTW). In Section 4, a mixed integer programming (MIP) model is formulated. A simulation optimization method is proposed in Section 5, and numerical experiments are presented in Section 6. Conclusions and future research are given in the last section.

2. Literature review

To date, there have been numerous studies on scheduling various handling equipment in container terminals, such as quay crane scheduling, YC scheduling and internal truck scheduling [8,9]. Most of these studies are devoted to promote the handling efficiency, whereas only a small number of works address the energy-saving of container terminals at the operational level. In this section, a brief review of studies highly related to YC scheduling and energy-saving of container terminals is provided.

For energy-saving of container terminals, most studies are focused on the macro level, such as green management practices, green port policies, and the impact and evaluation of carbon dioxide emissions [10–14]. In the scarce literature on energy-saving at the operational level, Chang et al. [15] optimized the combined problem of berth allocation and quay crane assignment by developing a rolling-horizon model based on objective programming. In their work, quay crane's energy consumption was considered. However, this work is indirectly biased towards energy consumption of quay cranes, which considered energy consumption could be reduced based on the constraint of vessel departing on time and the less the energy consumption of a quay crane per unit move. Du et al. [16] proposed an enriched berth allocation model considering the fuel consumption of vessels sailing to a focal port. The model calculated the vessel emission in the sailing periods using the widely-used emission factors, and analyzed the emission in the mooring periods through a post-optimization phase based on the waiting time of the vessels. Esmemr et al. [17] proposed a simulation model to discuss environmental damage caused by handling equipment in cargo handling operations of a Turkish port, and to obtain the optimal number of equipment to deploy to minimize the environment damage. Golias et al. [18] presented a berth-scheduling problem to minimize vessels' total departure delay and thereby indirectly reduce the total fuel consumption and emissions produced by the vessels while in transit to their next ports of call. Vessel arrival times were considered as a decision variable and were optimized to achieve the objective of departure delay minimization. The optimal vessel speeds were hence calculated and provided to ocean carriers. He et al. [19] presented an integer programming model for sharing internal trucks among multiple container terminals, where the minimization of transportation energy consumption was considered as one of two objectives. Alvarez et al. [20] proposed a simulation model to evaluate the potential benefits of berthing policies and shipping contracts, where marine fuel consumptions of the vessels were considered as a function of their sailing speeds. It can be seen that the above-mentioned studies on energy-saving mainly consider the cooperation between terminals and shipping lines to optimize vessel fuel consumption, rather than focus on the trade-off between energy consumption and operation efficiency at a container terminal.

Specifically for YC scheduling, most research uses operations research and/or simulation. Kim and Kim [21] developed a MIP model for a single YC scheduling problem, and an optimal routing algorithm for a transfer crane was proposed. Similarly, a GA was used for solving the same problem [22]. Moreover, Narasimhan and Palekar [23] developed an integer programming model for the transtainer routing problem, and designed a heuristic algorithm to solve the matching problem on a line at the root node of the branch-and-bound tree. Ng and Mak [24] addressed the problem of scheduling a yard crane that performs a given set of loading/unloading jobs with different ready times. The objective was to minimize the sum of job waiting times. A branch-and-bound algorithm was proposed to solve the scheduling problem optimally, embedding efficient and effective algorithms for finding lower and upper bounds. Zhang et al. [25] presented a MIP model for dynamic YC deployment to find the times and routes of crane movements among blocks, subject to the minimization of the total delay of the workloads. The model was solved by Lagrangean relaxation. To minimize the total unfinished workload at the end of each time period, Cheung et al. [26] formulated a mixed-integer linear program for the yard crane scheduling problem, and proposed a successive piecewise-linear approximation method for solving the problem. Ng [27] addressed the problem of scheduling multiple yard cranes with inter-crane interference in a yard zone to minimize the sum of truck waiting times in yard. Chen et al. [28] presented an integrated model to schedule yard cranes. The problem was formulated as a Hybrid Flow Shop Scheduling problem with Precedence and Blocking constraints. Cao et al. [29] focused on providing an efficient operation strategy for the double-rail-mounted gantry crane systems to load outbound containers. A MIP model was developed to formulate the problem, and a greedy heuristic algorithm, a simulated annealing (SA) algorithm and a combined scheduling heuristic were used to solve the proposed problem. Li et al. [30] developed an efficient model for YC scheduling that takes into account realistic operational constraints such as inter-crane interference, fixed YC separation distances and simultaneous container storage/retrievals. The model could be solved efficiently using heuristics and a rolling-horizon algorithm, yielding near optimal solutions in seconds. Petering and Murty [31] presented a simulation model to evaluate the blocks' length in a container yard and different YC scheduling strategies among blocks in the same zone. The experiments showed that the simulation model can be suitably customized to real, pure-transshipment ports. He et al. [32] developed an objective programming model for a YC scheduling problem based on a static rolling-horizon approach. In their work, a hybrid algorithm, consisting of a parallel GA and heuristic rules, was applied. Simulation was employed for evaluation. Chang et al. [33] proposed a dynamic rolling-horizon decision-making strategy for a YC scheduling problem, which can obtain more near optimal solutions than those generated by the corresponding static rolling-horizon approach.

Overall, YC scheduling is a well-researched domain. There are plenty of papers in this area. However, to the best of our knowledge, few scholars have considered energy-saving in their YC scheduling models. Our study thus fills in the gap in the literature on the conventional YC scheduling problem.

3. Problem description

3.1. Construction of task groups

In YC scheduling, a task means one move of a YC for loading/unloading containers, which may be one or two 20 feet containers,

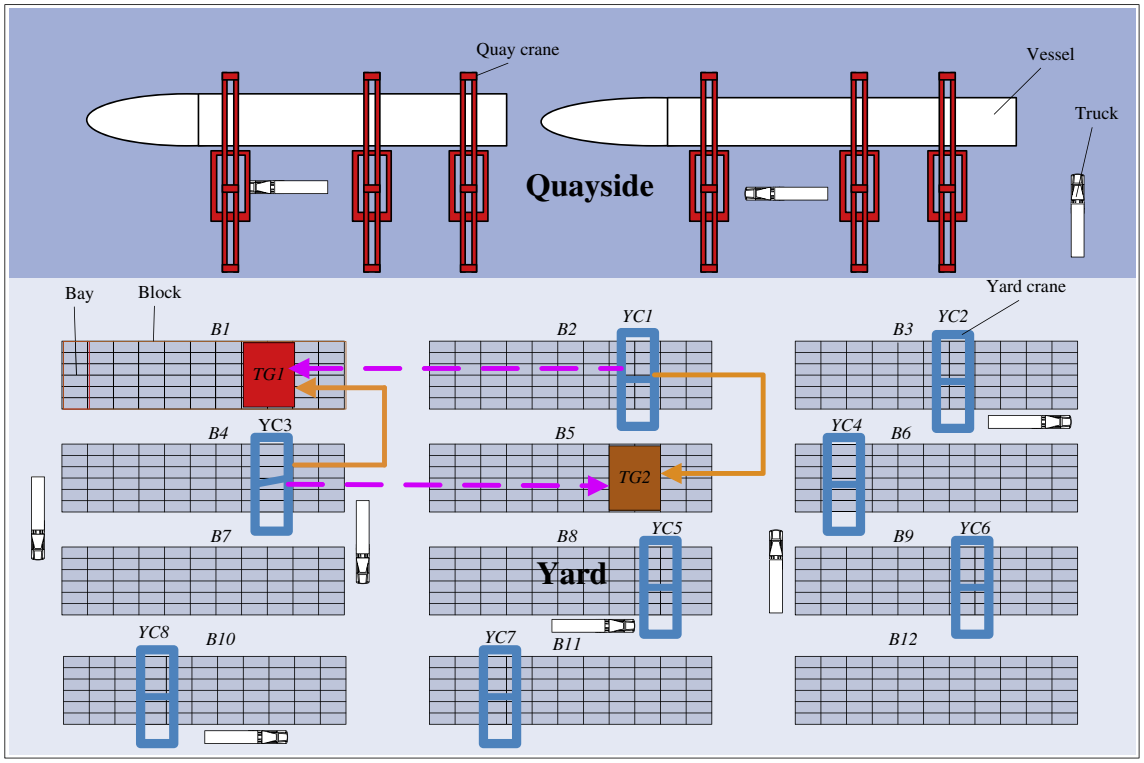


Fig. 1. General layout of a container terminal.

or one 40 feet container. To facilitate YC scheduling operations and reduce computational complexity, we do not deal with each task separately, but group the tasks based on the following rules.

(1) All tasks of one task group should be at the same block.
(2) All tasks of one yard bay should be grouped together.
(3) For SI or RE operations, all tasks of one task group should be from or for the same vessel.
(4) For SE or RI operations, all tasks of one task group should be from or for the same carrier.
(5) The volume of a task group should not be more than the work capacity of one YC from current time to the planned end time. If the volume of one block for one operation is more than the capacity of one YC, these tasks should be divided into n task groups, where n = [the volume/the capacity of one YC].
(6) An uncompleted task group from the previous period should be still treated as a single task group, and its arriving time is set at the beginning time of the current period. For a task group with the planned completion time within the previous period, its planned completion time in the current period can be set at the beginning time of the current period. For a task group with the planned completion time beyond the previous period, its planned completion time in the current period remains unchanged.
(7) In the YC scheduling problem, only tasks of a single period are considered. The span of a period is set as 4 h based on the approach proposed by Zhang et al. [25].

3.2. YC scheduling description

Up to present, most container terminals use rubber tired gantry cranes (RTGCs) or rail mounted gantry cranes (RMGCs) as YCs. RMGCs move on rails, which are fixed to each block. The construction allows more in-between storage lanes, offering higher storage capacity. RTGCs move on rubber tired wheels spanning over a block space. Since RTGC can be transferred from block to block flexibly, it is preferable in yard operation [34]. Therefore, this paper only studies the scheduling of RTGCs.

Fig. 1 shows the general layout of a container terminal using RTGCs as YCs, where the container yard is divided into many blocks and owns many YCs. The main function of YCs is to transfer containers between trucks and the stacks of the blocks. Since YCs are very expensive, container terminals generally cannot keep at least one YC in each block. Therefore, YCs are usually deployed in a way such that one YC serves more than one block by moving between blocks. Since container terminals generally settle on a fixed planning period, YC scheduling should be determined for all tasks within the next period by the operators at the end of the current period. YC scheduling is a very complex decision problem, which includes two interrelated decisions: (1) the handling sequence for task groups by a YC; and (2) start handling times of all task groups handled by YCs. There are two objectives for YC scheduling: (1) minimizing the total completion delay of all task groups; and (2) minimizing the total energy consumption for all task groups by YCs.

As vessels have their planned arriving and departure times, terminals must seek to ensure that they can depart punctually. Therefore, SI and RE operations should be completed punctually. In addition, since the appointed arriving times of SE and RI operations also exist in most terminals, and the cycle time of external trucks in terminals also should not be too long, operators of terminals should also minimize the completion delay of SE and RI operations.

Table 1
Transportation distances among Task groups 1 and 2, and YCs 1 and 2 (unit: m).

Task group	YC	
	YC1	*YC2*
TG1	480	120
TG2	120	480

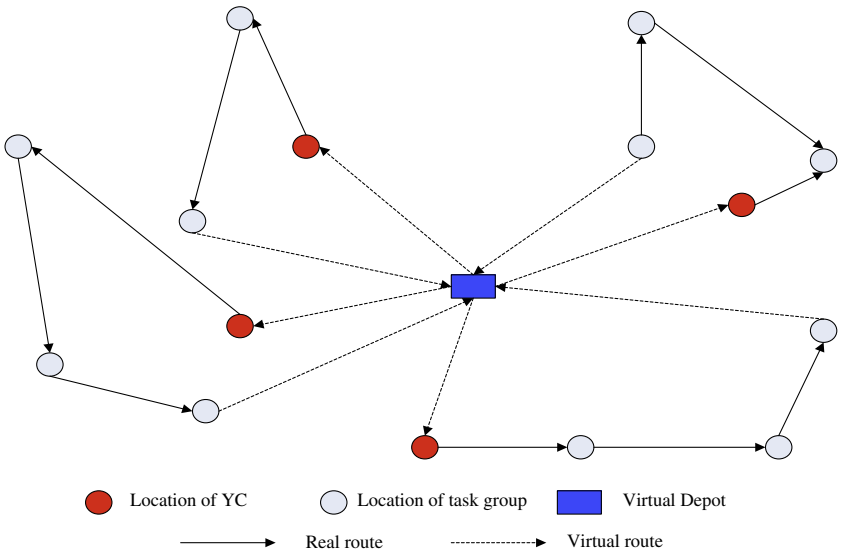


Fig. 2. A VRPSTW model for YC scheduling.

In a word, minimizing the total completion delay of all task groups is one of the most important objectives for container terminals.

In recent years, with the development of green port concept, energy-saving has become another objective of container terminals. Nevertheless, traditional methods for YC scheduling generally do not refer to the energy-saving objective. If the two objectives are positively correlated, traditional methods may be able to achieve the energy-saving objective as well. However, there are many scenarios that the two objectives are negatively correlated. Fig. 1 illustrates an example of YC scheduling with the two negatively correlated objectives. In this example, Task groups 1 and 2 respectively are at Blocks 1 and 2. The transportation distances among all task groups and YCs are listed in Table 1. The parameters, including moving speed, turning time, moving power and non-working power of YCs, are set at 2 m/s, 2.5 min, 148 kW/h and 80 kW/h, respectively. There are two YC schedules: (1) YCs 1 and 2 respectively are deployed to Task groups 1 and 2; and (2) YCs 1 and 2 respectively are deployed to Task groups 2 and 1. The total transferring time, the total moving energy consumption and the total non-working energy consumption of *Schedule* 1 are 8 min, 19.732 kW h and 10.667 kW h, respectively. These indices of *Schedule* 2 are 12 min, 4.933 kW h and 16 kW h, respectively. A careful examination reveals that, owing to the 90-degree-turn of YCs in *Schedule* 2, the total transferring time and the total non-working energy consumption are greater than those of *Schedule* 1. However, the total moving energy consumption of *Schedule* 2 is smaller. Therefore, container terminals should select *Schedule* 1 if the time-saving objective is considered alone, or select *Schedule* 2 if the energy-saving objective is considered alone. Nevertheless, container terminals prefer seeking the optimal trade-off between the two objectives to considering only one objective. Hence, this paper aims to propose a YC scheduling strategy for balancing efficiency and energy consumption.

3.3. YC scheduling as VRPSTW

To make the two interrelated decisions for the YC scheduling problem, we model it as a VRPSTW. This approach is a novel idea that makes the YC scheduling problem easier to formulate. To the best of our knowledge, few scholars have specially addressed the problem as a VRPSTW [35]. In the VRPSTW description, the task groups and initial locations of YCs are seen as customers and the YCs are seen as vehicles. We can set a virtual depot as the vehicles' starting locations. Each vehicle starts its tour at the virtual depot, visits a number of customers, and ends its tour at the virtual depot. The VRPSTW is defined on a graph $G = (V,A)$, where the set of vertices $V = N \cup \{0\}$ contains all vertices for all customers as well as vertex 0 which marks the virtual depot. The set of arcs A is a subset of $V \times V$. Each vertex except depot must be visited by only one vehicle. Time windows can be imposed on every vertex except the virtual depot. Since all YCs must travel from the virtual depot to their initial locations at the beginning time, the time windows of all YCs' initial location vertices can be set at $(0,0)$. The traveling times between the virtual depot and initial location vertices of all YCs should also be set at zero. For a task group, the time window indicates its arriving time and planned completion time. The start time of a YC handling a task group must not be smaller than the lower bound of the task group's time window. However, the YC's end time for the task group can be greater than the planned completion time. In such cases, penalty costs are incurred to account for the delay. Thus, the time window of each customer is soft. For the vertices of initial locations of all YCs, since there is no handling volume, the service times should be set at zero. For the vertices of all task groups, the service times are related to the handling volume. Fig. 2 shows a complete graph representing a VRPSTW model for YC scheduling.

4. Model formulation

4.1. Notation

Parameters	
T	The set of all task groups at the beginning of the time horizon (indexed by i)
Y	The set of YCs that can be scheduled at the beginning of the time horizon (indexed by j)
U	The set of all customers, where $U = T \cup Y \cup 0$ (indexed by k)
V_k	The handling volume of Customer k (unit: move). The handling volume of each YC's initial location and the virtual depot is zero
S_k	The service time of Customer k by a YC

Parameters	
a_k	The lower bound of the time window on Customer k. For each task group, the lower bound should be its arriving time. For each YC's initial location, the lower bound should be zero
dp_k	The upper bound of the time window on Customer k. For each task group, the upper bound should be its planned completion time. For each YC's initial location, the upper bound should be zero
$d_{kk'}$	The moving distance from the center position of Customer k to the center position of Customer k', where $d_{0k} = 0$ and $d_{k0} = 0$ (unit: m)
$t_{kk'}$	The moving time of a YC from the center position of Customer k to the center position of Customer k', where $t_{0k} = 0$ and $t_{k0} = 0$ (unit: h)
TS	The moving speed of a YC (unit: m/h)
TT	The turn time of a YC (unit: h)
p	The handling efficiency of a YC (unit: h/move)
μ	The working energy consumption of a YC per move (unit: kW h/move)
ξ	The non-working energy consumption of a YC per unit time (unit: kW)
τ	The moving energy consumption of a YC per unit distance (unit: kW h/m)
Cp_k	The unit penalty cost for delayed completion time of Customer k (unit: \$/h). For each YC's initial location, the penalty cost should be zero
CE	The unit energy cost (unit: \$/kW h)
M	A large positive number
δ^j_{0k}	A binary parameter satisfying $\delta^j_{0k} = 1$, if arc $(0,k)$ must be traversed by YC j; and $\delta^j_{0k} = 0$, otherwise. This parameter can ensure that each YC must travel from the virtual depot to its initial location

The moving time of a YC from the center position of Customer k to the center position of Customer k' can be defined as:

$$\forall k,k' \in U, k,k' \neq 0, tt_{kk'} = \begin{cases} d_{kk'}/TS, \text{ if YC does not have to make a 90-degree} \\ \quad \text{turn twice between Customers } k \text{ and } k'; \\ d_{kk'}/TS + 2TT, \text{ otherwise} \end{cases} \tag{1}$$

The moving time of a YC from the virtual depot to each customer should be set at zero.

The service time of Customer k by a YC can be defined by the following equation:

$$S_k = \begin{cases} 0, k = 0 \text{ or } k \in Y; \\ V_k \cdot P, k \in T \end{cases} \tag{2}$$

Decision variables	
$\varphi^j_{kk'}$	$\varphi^j_{kk'} = 1$, if arc (k,k′) is traversed by YC j; $\varphi^j_{kk'} = 0$, otherwise
st_k	The start service time of a YC for Customer k

4.2. Yard crane's energy consumption model

The energy consumption of a task group handled by a YC consists of three parts: (1) working energy consumption; (2) non-working energy consumption; and (3) moving energy consumption of a YC from its previous task group to its current task group.

To calculate the working energy consumption of a task group, we assume the moving energy consumption of a YC between different tasks in a task group is zero. Therefore, the working energy consumption of a task group is a constant that is determined by its volume, and can be calculated as

$$E^w_i = \mu \cdot V_i \tag{3}$$

where E^w_i is the working energy consumption of task group i.

The non-working energy consumption of a task group mainly refers to the air conditioning energy consumption, the lighting energy consumption and other auxiliary equipment's energy consumption. The non-working energy consumption of a task group by a YC is mainly determined by the working time and the moving time from the previous task group to the current task group. The moving time from the previous task group to the current task group is determined by the decision variable $\varphi^j_{kk'}$.

The non-working energy consumption can be calculated as

$$E^{nw}_i = \varphi^j_{ii'} \cdot t_{ii'} \cdot \xi + \frac{V_i}{p} \cdot \xi \tag{4}$$

where E^{nw}_i is the non-working energy consumption of task group i.

The moving energy consumption of a YC from its previous task group to its current task group is mainly determined by the moving distance. The moving distance is determined by the decision variable $\varphi^j_{kk'}$.

The moving energy consumption can be calculated as

$$E^{mw}_i = \varphi^j_{ii'} \cdot d_{ii'} \cdot \tau \tag{5}$$

where E^{mw}_i is the moving energy consumption of Task group i.

4.3. Mathematical model

The YC scheduling as a VRPSTW can formulated as the following MIP model.

$$[\text{P1}]\ \text{Objective 1}: \ \min f_1 = \sum_{k \in U} \max(st_k + S_k - dp_k, 0) \tag{6}$$

$$\text{Objective 2}: \ \min f_2 = \sum_{j \in Y} \sum_{k \in U} \sum_{k \in U, k \neq k'} \varphi^j_{kk'} \cdot (t_{kk'} \cdot \xi + d_{kk'} \cdot \tau) + \sum_{k \in U} (S_k \cdot \xi + V_k \cdot \mu) \tag{7}$$

$$\text{S.T.} \ \sum_{j \in Y} \sum_{k \in U, k \neq k'} \varphi^j_{kk'} = 1, \forall k' \in U, k' \neq 0 \tag{8}$$

$$\sum_{j \in Y} \sum_{k' \in U, k' \neq k} \varphi^j_{kk'} = 1, \forall k \in U, k \neq 0 \tag{9}$$

$$\sum_{j \in Y} \varphi^j_{0k} \leqslant 1, \forall k \in U, k \neq 0 \tag{10}$$

$$\sum_{k \in U, k \neq 0} \varphi^j_{0k} = 1, \forall j \in Y \tag{11}$$

$$\sum_{j \in Y} \varphi^j_{k0} \leqslant 1, \forall k \in U, k \neq 0 \tag{12}$$

$$\sum_{k\in U,k\neq 0}\varphi^{j}_{k0}=1,\forall j\in Y \tag{13}$$

$$\varphi^{j}_{kk'}+\varphi^{j}_{k'k}\leqslant 1,\forall k,k'\in U,k\neq k',\ j\in Y,k,k'\neq 0 \tag{14}$$

$$\sum_{k\in U,k\neq k'}\varphi^{j}_{kk'}=\sum_{k''\in U,k''\neq k'}\varphi^{j}_{k'k''},\forall k'\in U,k'\neq 0,\ j\in Y \tag{15}$$

$$st_k\geqslant a_k,\forall k\in U \tag{16}$$

$$st_k+(S_k+t_{kk'})\leqslant st_{k'}+M\cdot(1-\varphi^{j}_{kk'}),\forall k,k'\in U,k\neq k',\ j\in Y \tag{17}$$

$$\varphi^{j}_{0k}=\delta^{j}_{0k},\forall k\in U,k\neq 0,\ j\in Y \tag{18}$$

$$st_k=0,\forall k\in Y \text{ or } k=0 \tag{19}$$

$$\varphi^{j}_{kk'}\in(0,1),\forall k,k'\in U,k\neq k',\ j\in Y \tag{20}$$

$$st_k\geqslant 0,\forall k\in U \tag{21}$$

Eq. (6) minimizes the total completion delay of all task groups, which is the most frequently-used objective in YC scheduling. This equation also indicates the completion delay of a task group is zero when it is completed ahead of its planned completion time. Eq. (7) minimizes the total energy consumption of all task groups by YCs, which is another major concern for port operators due to environment pollution and limited energy resources. In Eq. (7), $\varphi^{j}_{kk'}\cdot t_{kk'}\cdot\xi$ and $\varphi^{j}_{kk'}\cdot d_{kk'}\cdot\tau$ respectively denote the non-working energy consumption and the moving energy consumption in the moving process of YC j from Customer k to Customer k', and $S_k\cdot\xi$ and $V_k\cdot\mu$ respectively denote the non-working energy consumption and working energy consumption in the handling process of Customer k. To unify the two objectives, we reformulate them in terms of cost. In detail, the service level is reflected by multiplying the completion delay of a task group by a unit penalty cost, and the energy cost is equal to energy consumption multiplied by the unit energy cost. Therefore, the two objectives can be unified into one cost function:

$$\min f=\sum_{k\in U}\max(st_k+S_k-dp_k,0)\cdot Cp_k+\left(\sum_{j\in Y}\sum_{k\in U}\sum_{k\in U,k\neq k'}\varphi^{j}_{kk'}\cdot(t_{kk'}\cdot\xi+d_{kk'}\cdot\tau)+\sum_{k\in U}(S_k\cdot\xi+V_k\cdot\mu)\right)\cdot CE \tag{22}$$

Constraint (8) ensures that there is exactly one preceding task group that is assigned to the same YC as Task group k'. Constraint (9) ensures that there is exactly one succeeding task group that is assigned to the same YC as Task k'. Constraints (8) and (9) ensure that each task group is served by exactly one YC. Constraint (10) ensures that for each customer, there is at most one inflow arc from depot. Constraint (11) ensures that each YC must travel from the virtual depot to exactly one initial location. Constraints (10) and (11) ensure that all YCs must travel from the virtual depot to different initial locations. Constraint (12) ensures that for each task group, there is at most one outflow arc to virtual depot. Constraint (13) ensures that each YC must return to the virtual depot from exactly one task group. Constraints (12) and (13) ensure that all YCs must return to the virtual depot from different task groups. Constraint (14) ensures that the routes between any two vertices are unidirectional. For example, considering two task groups A and B, there is only one YC's route for the two task groups, either A → B or B → A. Route continuity is ensured by Constraint (15), i.e. if a YC visits a task group, it must exit from that task group. Constraint (16) defines the lower bounds of the service time windows for all customers, which ensure that the start service time for each task group by a YC must not be earlier than the task group's arriving time. Constraint (17) defines the relationship between decision variables $\varphi^{j}_{kk'}$ and st_k: $st_k+(S_k+t_{kk'})\leqslant st_{k'}$ when YC j travels from Task group k to Task group k'. Constraint (17) also ensures the consistency of start time and service sequence on each YC. In *conjunction with* Constraint (14), Constraint (17) eliminates the impact of sub-circuits. Constraint (18) ensures that each YC must travel from the virtual depot to its initial location. Constraint (19) ensures that each YC must travel to its initial location at the beginning time of the period. Constraint (20) defines the binary variables, and Constraint (21) defines the integer variables.

Since Eq. (22) contains maximum value forms, model [P1] can be linearized by substituting $\max(st_k+S_k-dp_k,0)$ with auxiliary variable f_k. The rewritten model is as follows.

$$[\text{P2}]\quad \min f=\sum_{k\in U}f_k\cdot Cp_k+\left(\sum_{j\in Y}\sum_{k\in U}\sum_{k\in U,k\neq k'}\varphi^{j}_{kk'}\cdot(t_{kk'}\cdot\xi+d_{kk'}\cdot\tau)+\sum_{k\in U}(S_k\cdot\xi+V_k\cdot\mu)\right)\cdot CE \tag{23}$$

S.T. constraints (8)–(21) and

$$f_k\geqslant st_k+S_k-dp_k,\forall k\in U \tag{24}$$

$$f_k\geqslant 0,\forall k\in U \tag{25}$$

Constraints (24) and (25) indicate that if $st_k+S_k-dp_k>0$, variable f_k must be greater than or equal to $st_k+S_k-dp_k$; otherwise, variable f_k must be greater than or equal to zero.

5. Solution method

5.1. Framework of solution method

As aforementioned, model [P2] is a VRPSTW. The VRP is well-known to be NP-hard [36]. Therefore, the proposed YC scheduling model is also NP-hard. Due to the computational intractability, it is extremely difficult to obtain an exact optimal solution for a large size YC scheduling problem in reasonable computational time using traditional approaches such as branch-and-bound algorithms and commercial mathematical programming solvers such as ILOG CPLEX. Therefore, a simulation optimization method is proposed for solving the YC scheduling problem. In the solution method, the simulation is designed to evaluate and handle constraints, and the optimization algorithm is designed for global search. The optimization algorithm integrates a GA and a particle swarm optimization (PSO) algorithm. Since a pure GA is very difficult to explore the solution space effectively, a combination of a global search optimization method with a local search optimization method usually improves the performance of the algorithm [37]. In this paper, the GA is used for global search and PSO is used for local search and adjusting the GA's search direction to identify potential better solutions in each generation. At the end of each generation in the GA, some chromosomes are selected to be the initial particles of PSO for local search. In the PSO algorithm, a simulation module provides the same function as the GA does. The GA adjusts the search direction and creates a new set of chromosomes according to the results obtained by the simulation. The above-mentioned procedure is not stopped until a termination criterion is satisfied. Fig. 3 shows the framework for the proposed solution method.

5.2. Genetic algorithm

5.2.1. Solution representation

In the proposed GA, the YC scheduling problem is represented as a two-vector chromosome by the sequence encoding method, which denotes customers and YCs in the two vectors. Both vectors of the chromosome are integer. Each gene in the first vector represents a customer, and the sequence of the genes is the order of visiting these customers. The number of genes in the first vector is the same as the number of the task groups. The number of genes in the second vector is the same as that in the first vector. Each gene in the second vector represents the YC that serves the corresponding customer in the first vector. Fig. 4 gives an example of a possible solution with 12 customers served by 3 YCs as follows:

Route *YC*1: 0 → 5 → 1 → 7 → 8 → 4 → 0.
Route *YC*2: 0 → 3 → 12 → 9 → 0.
Route *YC*3: 0 → 10 → 11 → 6 → 2 → 0.

Customer	5	1	7	8	4	3	12	9	10	11	6	2
YC	1	1	1	1	1	2	2	2	3	3	3	3

Fig. 4. Structure of a solution representation.

5.2.2. Initialization

A modified heuristic based on the Push-Forward Insertion Heuristic (PFIH) [38] is used to generate the initial solutions for the GA. Since each YC must travel from the virtual depot to its initial location, the number of routes must be equal to the number of YCs and the first customer of each route is also fixed according to the parameter δ^j_{0k} (see Section 4.1). Therefore, in the modified PFIH, the first customer of each route is fixed. In order to maintain the diversity of the population, the second customer of each route is randomly selected from the unrouted customers. We use the first type of insertion heuristic proposed by Solomon [38] to calculate the best feasible insertion place and the best insertion customer.

Let $(j_0 \to j_1 \to \ldots j_{p-1} \to j_p \ldots \to j_m)$ be the current route, with $j_0 = j_m = 0$. The best feasible insertion place is calculated as

$$C_1(k(u), u, k'(u)) = \min(C_1(j_{p-1}, u, j_p)), p = 1, \ldots, m \tag{26}$$

where

$$C_1(k, u, k') = \alpha_1 C_{11}(k, u, k') + \alpha_2 C_{12}(k, u, k'), \alpha_1 + \alpha_2 = 1, \alpha_1, \alpha_2 \geqslant 0 \tag{27}$$

$$C_{11}(k, u, k') = d_{ku} + d_{uk'} - \mu d_{kk'}, \mu \geqslant 0 \tag{28}$$

$$C_{12}(k, u, k') = st_{k'u} - st_{k'} \tag{29}$$

In the Eq. (29), $st_{k'u}$ is the new start time for visiting at Customer k'.

The best insertion customer is calculated as

$$C_2(k(u^*), u^*, k'(u^*)) = \text{Optimum}[C_2(k(u), u, k'(u))] \tag{30}$$

where

$$C_2(k, u, k') = \lambda d_{0u} - C_1(k, u, k'), \lambda \geqslant 0 \tag{31}$$

The procedure of initialization is represented by the following pseudo code.

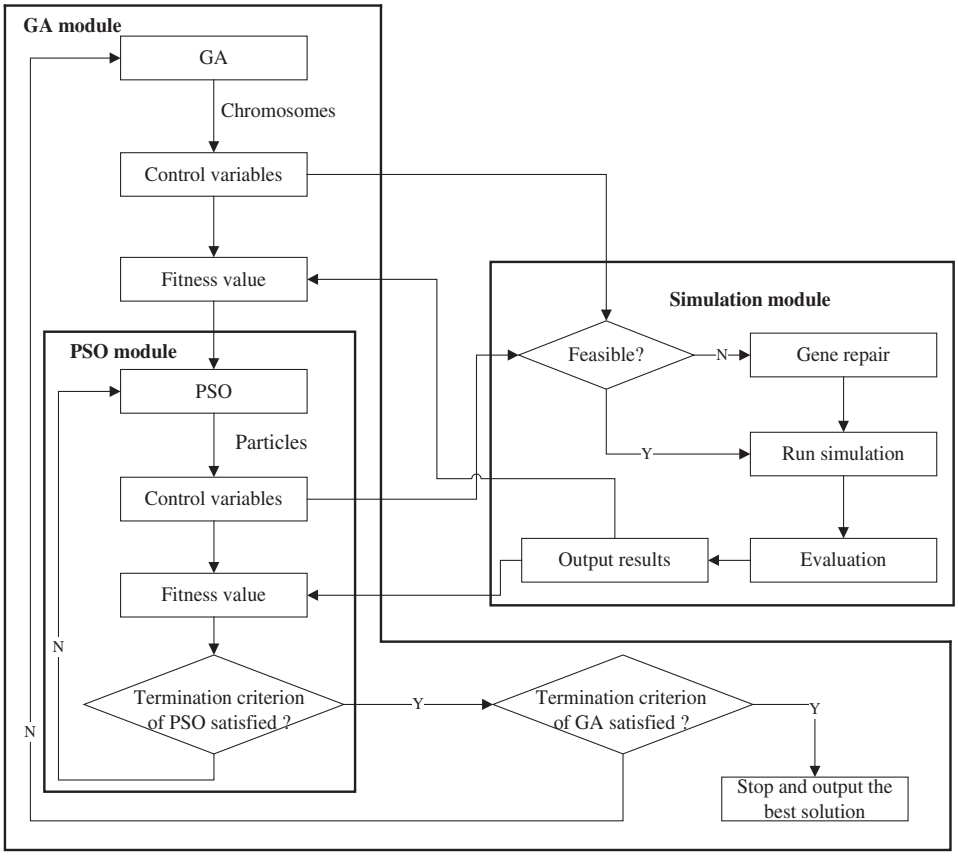


Fig. 3. Framework of the proposed simulation optimization method.

Create N routes, where N is the number of YCs;
Select the first customer of each route according to the parameter δ^j_{0k};
Randomly select the second customer of each route from the set of unrouted customers;
While the set of unrouted customers is not empty **do**
For all routes, indexed by j, **Loop**
For all unrouted customers **Loop**
For all edges in Route j **Loop**
Compute the insertion cost of Customer k at each edge in Route j according to Eqs. (27)–(29);
End For;
Compute the best feasible insertion place of Customer k in Route j according to Eq. (26);
End For;
Compute the best insertion customer in Route j according to Eqs. (30) and (31);
End For;
Compare the insertion costs of the best insertion customers for all routes;
Select a Route r with the least insertion cost among all routes;
Select a Customer u with the least insertion cost for Route r;
Insert Customer u into Route r at the best insertion place;
Exclude Customer u from the set of unrouted customers;
End While;

5.2.3. Individual feasibility and gene repair

All constraints are easily modeled in the simulation module, and the feasibility of all chromosomes should be checked. If an individual is infeasible, it has to be repaired. Therefore a gene repair procedure is proposed to repair infeasible individuals. The details of the gene repair procedure are as follows.

Step 1: If the first customer of Route j is not the customer defined by parameter δ^j_{0k}, swap the two customers.
Step 2: For all customers except the first customer of Route j, if $\varphi^j_{kk'}=1, \varphi^j_{k'k''}=1, a_{k'}>a_{k''}$, and $st_k+(S_k+t_{kk''})\leqslant a_{k''}$, then let $\varphi^j_{kk'}=0, \varphi^j_{k'k''}=0, \varphi^j_{kk''}=1$, and $\varphi^j_{k''k'}=1$.
Step 3: For all customers except the first customer of Route j, if $\varphi^j_{kk'}=1, \varphi^j_{k'k''}=1, a_{k'}>a_{k''}, [st_k+(S_k+t_{kk''})>a_{k''}$, and $st_k+(S_k+t_{kk'})>st_k+(S_k+t_{kk''})]$, then let $\varphi^j_{kk'}=0$, $\varphi^j_{k'k''}=0$, $\varphi^j_{kk''}=1$ and $\varphi^j_{k''k'}=1$.
Step 4: Suppose $\varphi^j_{kk'}=1$ and $\varphi^{j'}_{k''k'''}=1$. If $st_{k''}+S_{k''}+t_{k''k'}<st_k+S_k+t_{kk'}$ and at least one of the two inequalities are satisfied: $\max(st_{k''}+S_{k''}+t_{k''k'},a_{k'})+S_{k'}+t_{k'k'''}\leqslant a_{k'''}$ and $\max(st_{k''}+S_{k''}+t_{k''k'},a_{k'})+S_{k'}+t_{k'k'''}\leqslant st_{k''}+S_{k''}+t_{k''k'''}$, then let $\varphi^j_{kk'}=0$, $\varphi^{j'}_{k''k'''}=0, \varphi^{j'}_{k''k'}=1$, and $\varphi^{j'}_{k'k'''}=1$.
Steps 2–4 are mainly used to shorten the idle times of YCs, which can accelerate the convergence of the solution method. For example, assume the handling sequence of a YC is $0\to2\to1\to3\to0$, while the arriving time of Task group 1 is later than the arriving time of Task group 3, i.e., $a_1>a_3$, and the time of the YC arriving at Task group 3 from Task group 2 is earlier than the arriving time of Task group 3, i.e., $st_2+(S_2+t_{23})\leqslant a_3$. Then the handling sequence of the YC should be repaired as $0\to2\to3\to1\to0$ according to Step 2.

Since the gene repair process is dynamic and needs to be implemented a number of times, it is very complex to execute without using simulation [14]. Therefore, we develop the gene repair process in the simulation module.

5.2.4. Fitness evaluation and parent selection

The YC scheduling is a minimization problem, meaning that a smaller objective function value is associated with a higher fitness value. The fitness function is hence defined as

$$F=1\Big/\left[\sum_{k\in U}f_k\cdot Cp_k+\left(\sum_{j\in Y}\sum_{k\in U}\sum_{k\in U,k\neq k'}\varphi^j_{kk'}\cdot(t_{kk'}\cdot\xi+d_{kk'}\cdot\tau)+\sum_{k\in U}(S_k\cdot\xi+V_k\cdot\mu)\right)\cdot CE\right] \tag{32}$$

To select a parent, we apply a roulette wheel approach [39].

5.2.5. Crossover operator

The order crossover [39] is used as the crossover operator for the 'Customer' chromosome (the first vector in a chromosome). Since the aforementioned initialization procedure and order crossover ensure that the first gene of each offspring is the first customer of a route, we can determine the specified YC for the first gene. After the crossover operation of the 'Customer' chromosome, the 'YC' chromosome (the second vector in a chromosome) is determined by the following procedure.

Assign the specified YC to the first gene of the 'Customer' chromosome of the generated offspring;
For i = 2 **to** L **Loop** // L denotes the length of the 'Customer' chromosome
If gene i is not the first customer of any routes **then**
Assign the YC for gene $i-1$ to gene i;
Else
Assign the specified YC to gene i;
End If;
End For;

5.2.6. Mutation operator

The main task of the mutation is to promote the diversity of population and avoid the GA's premature convergence to local optimal solutions [40]. In this paper, a swap mutation operator is used for the 'Customer' chromosome [40].

5.2.7. Offspring acceptance

After implementing the selection, crossover, and mutation operations, we combine all parents and offspring, the numbers of which are denoted by *np* and *no*, respectively, to form a population. To keep the size of the population unchanged in the next generation, all individuals in the population are ranked via their fitness values in a descending sequence. Only the top *np* individuals are accepted to form the new generation.

5.2.8. Termination rule

An experimentally-determined maximum number of elapsed generations is used as the GA's termination rule.

5.3. Particle swarm optimization algorithm

In this paper, the PSO is used to identify potential better solutions of each generation in the process of the GA search.

5.3.1. Representation of the PSO

In the PSO algorithm, the solution representation is the same as that for the GA. Due to the continuous nature of the basic PSO, the smallest position value (SPV) rule [41] is used to convert a continuous position vector into a discrete sequence. Taking the

'Customer' chromosome of Fig. 4 as an example, the representation of the particle is:

5 1 7 8 4 3 12 9 10 11 6 2

Suppose that after the calculation of the velocities and positions of the particle, the particle becomes:

0.23 1.58 −0.09 2.36 −1.25 4.63 2.06 0.96 1.88 −3.25 4.29 5.35

Then, all elements of the particle should be transformed into the integer domain by the SPV rule, i.e., assigning the smallest floating value to the smallest integer, the next floating value to the next integer, until all the floating values are assigned. The backward transformation gives the result:

11 4 7 5 9 1 10 12 8 3 6 2

Since the first element of the particle must be the specified first customer of a route, we have to further revise the particle (as Customer 11 is not the first customer of any YC). To this end, we check each element of the particle from left to right. Once a customer is the first one of a route, swap the customer and all customers on its right with all customers on its left. In this example, Customer 5 is the first customer of Route 1; thus, the new representation of the particle becomes

5 9 1 10 12 8 3 6 2 11 4 7

The complete representation of the particle is given in Fig. 5.

5.3.2. Particle velocity and position

In this study, constriction particle swarm optimization [42] is used for the calculation of the velocity v_{jd}, which has been validated to perform better for the VRP [43]. The velocity can be calculated as

$$v_{jd}(t+1) = \chi(v_{jd}(t) + c_1 \cdot rand_1 \cdot (pbest_{jd} - x_{jd}(t)) + c_2 \cdot rand_2 \cdot (gbest_d - x_{jd}(t))) \quad (33)$$

where χ is a constriction factor, t is the iterations counter, c_1 and c_2 are acceleration coefficients, $rand_1$ and $rand_2$ are two random numbers between 0 and 1, $pbest_{jd}$ is the personal best position of each particle, and $gbest_d$ is the global best position attained by any particle of the swarm. The constriction factor χ is calculated as

$$\chi = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \quad (34)$$

where $c = c_1 + c_2$ and $c > 4$.

The c_1 and c_2 are respectively defined as follows:

$$c_1 = c_1^{\min} + \frac{c_1^{\max} - c_1^{\min}}{G^{\max}} \cdot g \quad (35)$$

$$c_2 = c_2^{\min} + \frac{c_2^{\max} - c_2^{\min}}{G^{\max}} \cdot g \quad (36)$$

where $c_1^{\min}$, $c_1^{\max}$, $c_2^{\min}$ and $c_2^{\max}$ are the minimum and maximum values of c_1 and c_2, $G^{\max}$ is the maximum number of generations of the PSO, and g is the number of the current generation.

Customer	5	9	1	10	12	8	3	6	2	11	4	7
YC	1	1	1	3	3	3	2	2	2	2	2	2

Fig. 5. An illustration of solution representation of the PSO.

The initial velocity $v_{jd}(0)$ of each particle is calculated as

$$v_{jd}(0) = (x_{\min} - x_{jd}(0)) + ((x_{\max} - x_{jd}(0)) - (x_{\min} - x_{jd}(0))) \cdot U(0,1) \quad (37)$$

where $x_{\min}$ and $x_{\max}$ are the lower and upper bounds on the position of each particle, respectively.

The position of each particle is calculated as

$$x_{jd}(t+1) = x_{jd}(t) + v_{jd}(t+1) \quad (38)$$

5.3.3. Procedure of the PSO

The procedure of the PSO is represented in the following pseudo code.

```
Select N best individuals from the population of the GA as the
  initial population of the PSO;
Initialize the velocity of each particle using Eq. (37);
For i = 1 to T_p Loop // T_p denotes the maximum number of
  generations for the PSO
  Find the personal best position of each particle;
  Find the global best position attained by any particle of the
  swarm;
  Update the velocity of each particle using Eq. (33);
  Update the new position of each particle using Eq. (38);
  Fitness evaluation; // The fitness evaluation is the same as
  that for the GA
End For;
```

5.4. Simulation model

In our problem, simulation is used to evaluate YC schedules obtained by the optimization algorithm and execute gene repair procedure. The simulation model consists of six parts: (1) input module; (2) control module; (3) entity module; (4) output module; (5) evaluation module and (6) optimization module. A framework of the simulation model is presented in Fig. 6.

5.4.1. Input module

The input module is the basis for generating schedules and evaluating the schedules. It mainly consists of two parts: (1) static data and (2) dynamic data. The static data include all parameters presented in Section 4.1, such as the handling volume of each customer, the time window on each customer and the handling efficiency of each YC. A part of parameters are used for simulation process, such as the moving speed of each YC. The other parameters are used for evaluation, such as the cost of unit energy consumption. The dynamic data include all decision variables, which are generated by the optimization module and repaired by the control module. The first decision variable $\varphi_{kk'}^{j}$ is directly generated by the optimization module and repaired by the control module. The second decision variable st_k is directly generated by the control module according to the lower bound of time window on each customer and the completion time of its precedent customer.

5.4.2. Entity module

The entity module is used to undertake all simulation behaviors such as the traveling process of each YC between two customers and the service process of each YC for each customer. This module consists of three parts: (1) customer entities; (2) traveling entities and (3) YC entities. For all YCs' initial locations, the customer entities should be created before running simulation. All YC entities should be created before running simulation. However, to reduce the simulation scale, for each task group, the customer entity should be dynamically created when the task group has arrived. For the same reason, each traveling entity should also be

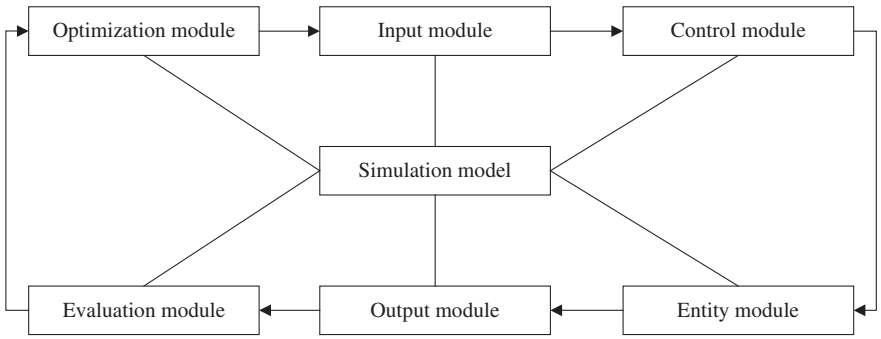


Fig. 6. A framework of the simulation model.

dynamically created. Each customer entity includes five attributes: (1) the lower bound of the time window; (2) the upper bound of the time window; (3) the handling volume; (4) the start service time; and (5) the end service time. Two attributes of each traveling entity are the traveling distance and the YC's traveling time between two customers. The attribute of each YC entity is the handling efficiency. The decision variable $\varphi^j_{kk'}$ is used to determine if a traveling entity should be generated.

5.4.3. Control module

The main functions of the control module are: (1) scheduling control; (2) time control; and (3) behavior control. The scheduling control is used to repair infeasible schedules and control the task groups handled by each YC according its operation sequence. The functions of time control are: (1) controlling the start and end of simulation; (2) controlling the arriving time of each task group; (3) controlling the start service time and the end time of a YC for each customer; and (4) controlling the moving time of each YC between two customers.

The behavior control module is used to control (1) the generation of each task group; (2) the traveling process of each YC between two customers; and (3) the service process of each YC for a customer. After generating a feasible schedule, the behavior control module controls the movement of the YC entities to the corresponding traveling entities in order to simulate the traveling process for the current customer. Once the traveling process is completed, the YC entity immediately moves to the corresponding customer entity to simulate the service process, and the decision variable st_k is also determined. After the service process is completed, the completion delay of the corresponding customer and the energy consumption of the YC for the customer are calculated and are output.

5.4.4. Output module

The output module is used to obtain all statistical indices, which include the completion delay of each task group and the energy consumption of each YC for each task group. These indices are used for fitness evaluation.

5.4.5. Evaluation module

The evaluation module is used to calculate the fitness value of each solution using Eq. (32). Upon the completion of simulation, the statistical indices are input into the optimization module for calculating fitness value of a solution.

5.4.6. Optimization module

The optimization module is used to generate YC schedules and to search the solution space for better solutions by evolving a population. This module consists of the GA and the PSO, which are presented in Sections 5.2 and 5.3, respectively.

6. Computational experiments

To verify the performance of the proposed model and solution method, a series of experiments with different sizes are conducted. This section consists of two parts: (1) performance analysis and (2) energy consumption analysis. In the performance analysis, we discuss the convergence of the proposed solution method, and compare the computational time and solution quality of three methods: the proposed method, the pure GA without PSO, and directly solving the mixed-integer linear programming model [P2] using ILOG CPLEX 12.4. The exact solutions obtained by CPLEX for small size instances and the lower bounds for large size instances could be used as a benchmark for evaluating the solution quality of the proposed method and the pure GA. In the energy consumption analysis, we explore the differences with and without incorporating the energy-saving objective in the scheduling process. All experiments are run on a PC with Intel Core TM i7-2820QM @ 2.3 GHz processors and 16 GB RAM.

6.1. Performance analysis

6.1.1. Generation of test instances

Initial parameters, including the moving speed, the turning time, the handling efficiency of a YC, the working energy consumption of a YC per move, the non-working energy consumption of a YC per unit time, and the moving energy consumption of a YC per unit distance are set at 2 m/s, 150 s, 120 s/move, 2 kW h/move, 0.0206 kW h/m and 80 kW h/h, respectively. In all test instances, some input data, such as the number of YCs, the number of task groups, the handling volume, the arriving time and the planned completion time of each task group, and the traveling distance and time between two customers, are randomly generated. The handling volume of each task group is randomly generated from a uniform distribution U (10,240). The arriving time of each task group is randomly generated from a negative exponential distribution E (4 h $*$ 3600 s/*Number of task groups)*. The planned completion time of each task group is calculated as

$$dp_k = a_k + S_k \cdot U(1.05, 1.1) \tag{39}$$

The traveling distance between any two customers is randomly generated from a uniform distribution U (10, 1800). The traveling time between any two customers is calculated by Eq. (4), where the probability of making a 90-degree turn twice is approximately 0.7.

The sizes of all instances are directly related to the number of task groups and the number of YCs. For each set of experiments, 10 instances are randomly generated using random seeds. In each instance, we use multiple random seeds to generate the initial population. The evaluation indices include the objective value and the CPU time. We take the average values of the evaluation indices for

Table 2
Parameters of all experiment sets.

Small size			Large size		
Experiment set	No. YCs	No. task groups	Experiment set	No. YCs	No. task groups
1	3	4	19	30	45
2	3	5	20	30	54
3	3	6	21	30	60
4	4	5	22	35	52
5	4	6	23	35	63
6	4	8	24	35	70
7	5	6	25	40	60
8	5	8	26	40	72
9	5	10	27	40	80
10	10	15	28	45	68
11	10	18	29	45	81
12	10	20	30	45	90
13	15	24	31	50	75
14	15	27	32	50	90
15	15	30	33	50	100
16	20	30	34	55	83
17	20	36	35	55	99
18	20	40	36	55	110

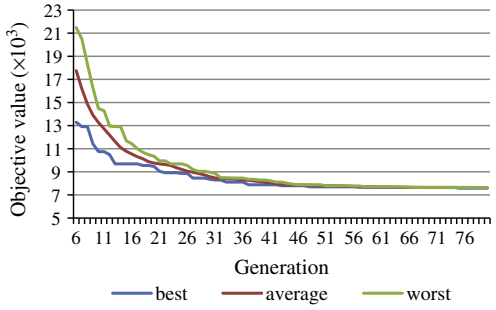


Fig. 7. Convergence of the proposed method for small size instance.

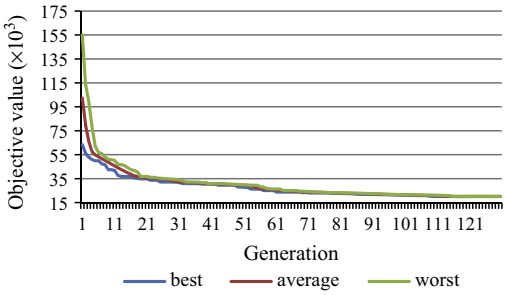


Fig. 8. Convergence of the proposed method for large size instance.

comparison. Table 2 shows the random parameters of 36 sets of test instances.

6.1.2. Discussion of convergence

To avoid the proposed method's premature convergence to local optimal solutions, it is important to choose the optimum control parameters for the proposed method. In this sub-section, preliminary tests are conducted to examine the convergence performance of the proposed method with different control parameter combinations. We use two problem instances as the examples to illustrate the convergence of the proposed method for small size instances and large size instances. The small size instance includes 20 YCs and 40 task groups, and the large size instance includes 55 YCs and 110 task groups.

In the tests, the population size takes the value of 60, 90, 120, 160 and 200, both the crossover rate and mutation rates take the value of 0.1, 0.2, 0.4, 0.6 and 0.8, the maximum number of the generations in the GA takes the value of 60, 80, 100, 120 and 200, the number of particles takes the value of 5, 10, 15, 20 and 25, and the maximum number of generations in the PSO takes the value of 5, 10, 15, 20 and 25. The minimum and maximum values of $c_1$and c_2 in the PSO are $c_1^{\min} = c_2^{\min} = 2$, $c_1^{\max} = c_2^{\max} = 5$, respectively [39].

The test results demonstrate that the following combination of parameters has the best convergence performance for the small size instance: population size 60, crossover probability 0.6, mutation probability 0.1, maximum number of generations in the GA 80, number of particles 20, and maximum number of generations in the PSO 15. The gap between the result obtained by the proposed method and the optimum result obtained by CPLEX is very small (only 1.4%). Therefore, the proposed method's premature convergence to local optimal solutions in the small size instances can be avoided. Fig. 7 illustrates the convergence of the proposed method for the small size instance.

Similarly, the best convergence performance of the proposed method for the large size instance is found when the population size, the crossover probability, the mutation probability, the maximum generations of the GA, the number of particles and the maximum generations of the PSO are set at 90, 0.8, 0.2, 120, 20 and 15, respectively. Fig. 8 illustrates the convergence of the proposed method for the large size instance.

6.1.3. Numerical experiments with small sizes

In this section, 18 sets of random instances with small sizes are tested to compare the results and CPU time obtained by the proposed method, the pure GA and CPLEX. Table 3 shows the results of the 18 sets of small size instances. It can be observed from Table 3 that the gaps between the results obtained from the proposed method and those obtained from CPLEX are very small (for example, the average gap is only 0.67%, the minimum gap is 0, and the maximum gap is only 2.51%). The average gap of the proposed method is only approximately 17% of that of the pure GA. Fig. 9 shows the trend of the gaps obtained by the aforementioned methods in these small size instances, which indicates that with the increment of the number of task groups, the gaps between the proposed method and CPLEX, and between the pure GA and CPLEX, generally increase.

Table 3
Results of all instances with small sizes.

No.	The proposed method			GA			CPLEX	
	Obj. value (1)	CPU (s)	Gap (%) $\frac{(1)-(3)}{(3)} \times 100$	Obj. value (2)	CPU (s)	Gap (%) $\frac{(2)-(3)}{(3)} \times 100$	Obj. value (3)	CPU (s)
1	635.50	4.50	0	635.50	4.08	0	635.50	3.40
2	1203.26	4.93	0	1203.20	4.56	0	1203.20	3.60
3	2009.04	6.74	0	2009.60	6.24	0	2009.60	3.80
4	743.51	5.31	0	766.07	4.85	3.05	743.40	3.40
5	1576.62	6.38	0	1627.91	5.86	3.29	1576.00	4.00
6	2928.40	14.29	0	3086.94	13.47	5.41	2928.40	8.40
7	1316.13	11.43	1.10	1347.84	10.53	2.03	1301.80	4.80
8	1587.66	16.39	0.45	1663.09	15.36	3.91	1580.60	5.40
9	3720.11	22.94	0.24	3912.52	21.66	3.78	3711.20	77.60
10	2327.37	94.23	0.48	2450.43	88.39	4.19	2316.20	208.00
11	2926.23	124.21	0.19	3090.73	112.94	4.75	2920.60	1136.20
12	4632.65	151.58	1.39	4836.93	140.51	3.76	4569.00	1389.60
13	2409.43	168.66	0.70	2525.56	156.63	3.61	2392.60	1589.60
14	3157.06	186.66	1.01	3406.14	173.26	7.08	3125.50	1867.00
15	4916.03	213.24	0.90	5284.01	197.96	7.27	4872.00	4223.00
16	3093.41	244.86	1.04	3238.99	223.81	4.12	3061.71	2122.40
17	3058.00	370.81	2.51	3308.17	345.47	8.93	2983.00	8021.80
18	5081.27	422.57	2.12	5317.29	397.35	6.86	4975.78	17351.80
The gap of the proposed method:			Average gap (%)	0.67	Max gap (%)	2.51	Min gap (%)	0
The gap of the GA:			Average gap (%)	4.00	Max gap (%)	8.93	Min gap (%)	0

Table 4
Results of all instances with large sizes.

No.	The proposed method			GA			CPLEX
	Obj. value (1)	CPU (s)	Gap (%) $\frac{(1)-(3)}{(3)} \times 100$	Obj. value (2)	CPU (s)	Gap (%) $\frac{(2)-(3)}{(3)} \times 100$	Lower bound (3)
19	3095.17	539.49	1.98	3285.3	501.27	8.24	3035.20
20	11823.67	573.09	2.92	12604.14	526.32	9.71	11488.60
21	16417.74	589.52	3.10	17617.61	544.64	10.63	15924.80
22	4969.09	586.55	1.58	5334.29	534.13	9.05	4891.60
23	17405.83	665.34	2.39	18623.5	609.31	9.55	17000.00
24	17713.41	698.03	2.86	19084.76	656.30	10.82	17221.40
25	12063.60	683.04	2.13	12941.23	635.58	9.56	11812.00
26	14811.86	732.89	3.87	15872.36	671.72	11.31	14259.60
27	15690.07	755.03	3.57	17118.14	701.76	13	15148.80
28	12297.23	729.56	1.44	13426.22	680.12	10.75	12123.00
29	16042.27	791.89	2.23	17358.49	721.59	10.62	15692.00
30	16693.73	820.39	3.24	18164.82	738.92	12.34	16169.50
31	12974.52	784.35	1.21	14085.33	713.59	9.87	12820.00
32	16476.47	848.61	3.11	18088.79	786.39	13.2	15979.50
33	20340.07	904.62	3.66	22263.12	810.47	13.46	19622.00
34	16836.05	838.88	2.75	18302.05	778.90	11.7	16385.00
35	20526.94	931.53	2.95	22230.55	856.04	11.49	19939.50
36	21217.36	978.56	4.36	23048.12	906.49	13.37	20330.00
The gap of the proposed method:			Average gap (%)	2.74	Max gap (%)	4.36	Min gap (%) 1.21
The gap of the pure GA:			Average gap (%)	11.04	Max gap (%)	13.46	Min gap (%) 8.24

As to the computational time, we observe according to Table 3 that the CPU time required by CPLEX grows exponentially as the instance size increases. The CPU time of CPLEX is on average 18.37 times of that of the proposed method and 19.78 times of that of the pure GA. Although the CPU times of the proposed method are longer than those of the pure GA, the difference is marginal (the CPU time of the proposed method is only 1.08 times of that of the pure GA on average). The CPU time of the proposed method is also very short in practice for small size instances (for example, the maximum CPU time of these 18 sets of instances is only 422.57 s). However, the maximum CPU time of CPLEX is more than 4.8 h for 20 YCs and 40 task groups, which is unacceptable in practice. Therefore, the proposed method is more suitable for solving small size instances.

6.1.4. Numerical experiments with large sizes

In this section, 18 sets of instances with large sizes are randomly generated to further compare the results and CPU time obtained by the proposed method, the pure GA and CPLEX.

For comparison, we need to find the optimal solution by CPLEX. However, it is difficult to obtain optimal solutions within a reasonable time limit by CPLEX for large size instances. Therefore, the lower bounds corresponding to the instances could be used as a benchmark. However, the lower bounds directly generated by CPLEX are very loose. Thus, we calculate the lower bounds corresponding to the instances by relaxing some complex constraints. Since Constraint (17) includes big-*M* and significantly impacts the CPU time based on the preliminary tests, it is selected for relaxing. A valid Inequality is given to substitute Constraint (17):

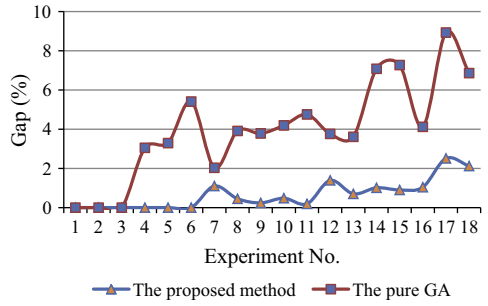


Fig. 9. The performance comparison of the proposed method and the pure GA for small size problems.

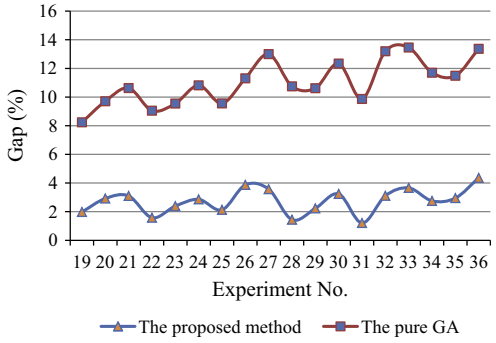


Fig. 10. The performance comparison of the proposed method and the pure GA for large size problems.

$$\sum_{k\in U,k\neq k'}(a_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}\leqslant st_{k'},\ \forall k'\in U,\ j\in Y \tag{40}$$

The validity of Inequality (40) is proven as follows.

Proof. Since $\sum_{j\in Y}\sum_{k\in U,k\neq k'}\varphi^j_{kk'}=1,\ \forall k'\in U,k'\neq 0,\sum_{k\in U,k\neq k'}\varphi^j_{kk'}\leqslant 1,\forall k'\in U,\ j\in Y$. There are two cases:

Case 1 $\sum_{k\in U,k\neq k'}\varphi^j_{kk'}=0,\forall k'\in U,\ j\in Y$.

In this case, $\sum_{k\in U,k\neq k'}(a_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}=0$. Therefore, Inequality (40) is evidently valid.

Case 2 $\sum_{k\in U,k\neq k'}\varphi^j_{kk'}=1,\forall k'\in U,\ j\in Y$.

We have:

$$\sum_{k\in U,k\neq k'}(a_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}=(a_0+S_0+t_{0k'})\cdot\varphi^j_{0k'}+(a_1+S_1+t_{1k'})\cdot\varphi^j_{1k'}+\cdots+(a_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}+\cdots+(a_{k'-1}+S_{k'-1}+t_{(k'-1)k'})\cdot\varphi^j_{(k'-1)k'}$$

Suppose $\varphi^j_{kk'}=1,\sum_{k\in U,k\neq k'}(a_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}=(a_k+S_k+t_{kk'})$ and $(st_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}\leqslant st_{k'}$.

Since $st_k\geqslant a_k, a_k+S_k+t_{kk'}\leqslant st_k+S_k+t_{kk'}$. Thus $\sum_{k\in U,k\neq k'}(a_k+S_k+t_{kk'})\cdot\varphi^j_{kk'}\leqslant st_{k'}$. Therefore, Inequality (40) is also valid in Case 2. □

The relaxed mathematical model can be solved by CPLEX exactly, and the objective value is the lower bound to the original problem. Table 4 shows the results of all large size instances, where Column 8 is the lower bound by CPLEX for each instance.

As observed in Table 4, the gaps between the results obtained by the proposed method and lower bounds are still very small for large size instances (for example, the average gap is only 2.74%, the minimum gap is 1.21%, and the maximum gap is only 4.36%). However, the gaps between the results obtained from the pure GA and the lower bounds are relatively large. The average gap of the proposed method is only approximately 25% of that of the pure GA. Fig. 10 shows the trend of the gaps obtained by the two methods for these large size instances, which indicates that the closer the number of YCs is to that of task groups, the smaller the gaps between the results obtained from the two methods and lower bounds from CPLEX are. The reason for it should be that the difference between the number of task groups and the number of YCs significantly affects scheduling.

The CPU time of the proposed method still is larger than that of the pure GA for large size instances. However, the difference is very small. On average, the CPU time of the proposed method is only 1.09 times of that of the pure GA. The CPU times of the proposed method for all the instances are acceptable in practice. Therefore, the proposed method is satisfactory for solving large size instances.

Overall, the proposed method is capable of obtaining high quality solutions for all the test instances in reasonable time.

6.2. Energy consumption analysis

In this section, the 36 sets of instances presented in Section 6.1.1 are tested to compare two scheduling strategies: (1) the

Table 5
YC scheduling costs for all instances with small sizes.

No.	Energy-saving oriented strategy			Time-saving oriented strategy		
	Delay cost	Energy cost	Total cost	Delay cost	Energy cost	Total cost
1	346.11	289.39	635.50	342.93	344.88	687.81
2	866.45	336.81	1203.26	862.24	448.05	1310.29
3	1649.75	359.29	2009.04	1642.67	508.65	2151.32
4	472.37	271.14	743.51	457.78	352.9	810.68
5	1187.16	389.46	1576.62	1184.31	536.22	1720.53
6	2516.49	411.91	2928.40	2459.49	723.13	3182.62
7	886.52	429.61	1316.13	873.30	531.84	1405.14
8	1148.22	439.44	1587.66	1120.41	641.00	1761.41
9	3078.39	641.72	3720.11	3045.56	1070.07	4115.63
10	1599.27	728.10	2327.37	1596.66	965.68	2562.34
11	2077.99	848.24	2926.23	2056.64	1170.7	3227.34
12	3620.40	1012.25	4632.65	3570.29	1462.82	5033.11
13	1354.81	1054.62	2409.43	1341.86	1309.6	2651.46
14	1791.79	1365.27	3157.06	1779.93	1671.03	3450.96
15	3398.05	1517.98	4916.03	3272.04	2197.36	5469.40
16	1602.59	1490.82	3093.41	1587.08	1837.09	3424.17
17	1847.42	1210.58	3058.00	1827.47	2074.04	3901.51
18	3481.58	1599.69	5081.27	3009.51	2307.78	5317.29
Average	1864.85	764.13	2628.98	1779.45	1119.60	2899.06

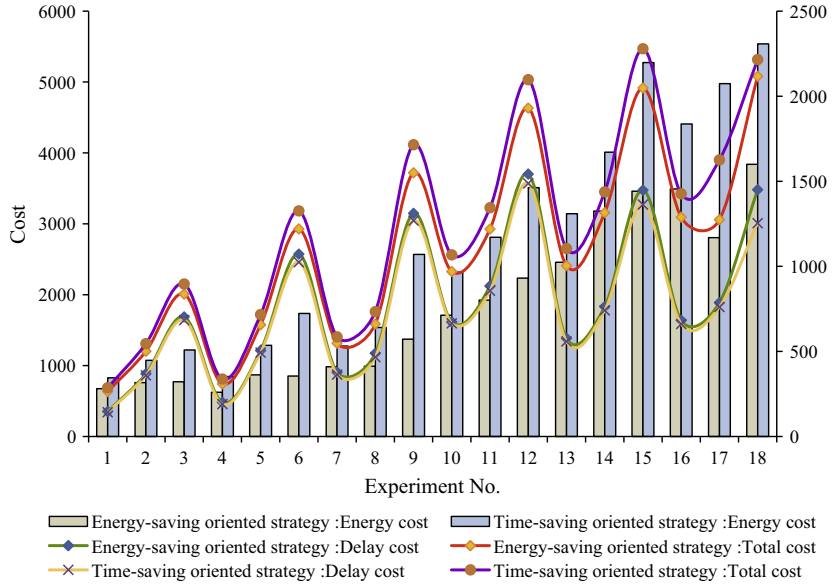


Fig. 11. Energy consumption analysis for all instances with small sizes.

Table 6
YC scheduling costs for all instances with large sizes.

No.	Energy-saving oriented strategy			Time-saving oriented strategy		
	Delay cost	Energy cost	Total cost	Delay cost	Energy cost	Total cost
19	1146.58	1948.59	3095.17	1118.24	2213.68	3331.92
20	10119.50	1704.17	11823.67	9942.64	2812.76	12755.40
21	14690.54	1727.20	16417.74	14373.61	3287.00	17660.61
22	2893.59	2075.50	4969.09	2839.01	2586.11	5425.12
23	15190.82	2215.01	17405.83	14975.41	3924.14	18899.55
24	15517.60	2195.81	17713.41	15168.71	3812.68	18981.39
25	9663.25	2400.35	12063.6	9442.63	3717.56	13160.19
26	12264.16	2547.70	14811.86	11945.24	4129.37	16074.61
27	12932.93	2757.14	15690.07	12601.20	4213.15	16814.35
28	9685.84	2611.39	12297.23	9475.47	3742.12	13217.59
29	13342.70	2699.57	16042.27	13095.08	4208.27	17303.35
30	14063.26	2630.47	16693.73	13881.87	4101.54	17983.41
31	9782.98	3191.54	12974.52	9649.57	4486.93	14136.50
32	13782.20	2694.27	16476.47	13551.56	4384.08	17935.64
33	17224.54	3115.53	20340.07	16884.22	5016.16	21900.38
34	13458.01	3378.04	16836.05	13255.15	4914.84	18169.99
35	17154.20	3372.74	20526.94	16878.31	5515.89	22394.20
36	17611.21	3606.15	21217.36	17268.80	5641.14	22909.94
Average	12251.33	2603.96	14855.28	12019.26	4039.30	16058.56

proposed method considering the energy-saving objective (namely "Energy-saving strategy"), and (2) the proposed method not considering the energy-saving objective (namely "Time-saving strategy"). The two scheduling strategies are measured in terms of delay costs, energy costs and total costs.

Table 5 shows the scheduling results obtained by the three strategies for small size instances. Fig. 11 shows the comparisons between the three strategies. As observed in Table 6, the total costs obtained from the Energy-saving strategy are the smaller than those obtained from the Time-saving strategy (for example, the average total cost obtained from the Energy-saving strategy is only approximately 90.68% of that obtained from the Time-saving strategy). Furthermore, the delay costs obtained from the Energy-saving strategy are larger than those obtained from the Time-saving strategy, but the relative gap is very small, only 1.33% on average. However, the energy costs obtained from the Energy-saving strategy are evidently smaller than those obtained from the Time-saving strategy (for example, the average energy cost obtained from the Energy-saving strategy is only approximately 68.25% of that obtained from the Time-saving strategy). Therefore, the Energy-saving strategy is more suitable for green transportation and green port.

For large size instances, Table 6 shows the scheduling costs obtained from the aforementioned three strategies, and Fig. 12 shows the comparisons between the two strategies. From the table and figure, we can find that the total costs and energy costs obtained from the Energy-saving strategy still are the smallest. The average total cost obtained from the Energy-saving strategy is only approximately 92.48% of that obtained from the Time-saving strategy. The average energy cost obtained from the Energy-saving strategy is only approximately 65.21% of that obtained from the Time-saving strategy. Although the delay costs obtained from the Energy-saving strategy may be larger than those obtained from the Time-saving strategy, the relative gap is very small, only 1.97% on average. Therefore, the proposed method

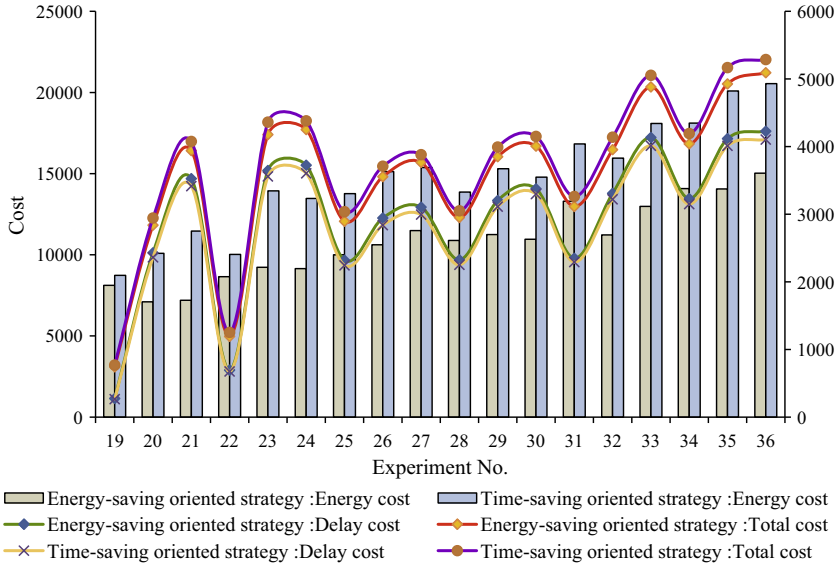


Fig. 12. Energy consumption analysis for all instances with large sizes.

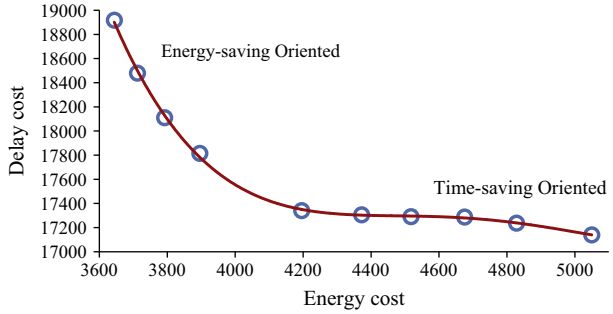


Fig. 13. An example for analyzing coefficient λ between delay cost and energy cost.

considering energy-saving objective is more suitable for green transportation and green port.

Since the relative importance of the total delay to the energy consumption is different in different container terminals, the two objectives' weights can be set as different values. We use coefficient λ to combine the two objectives:

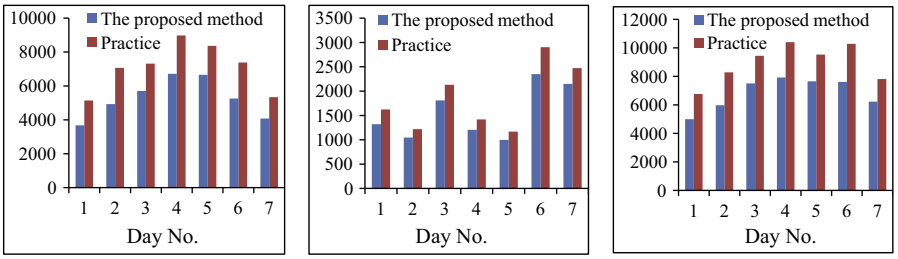
$$\min f = \lambda \cdot \sum_{k\in U} f_k \cdot Cp_k + (1-\lambda) \cdot \left(\sum_{j\in Y} \sum_{k\in U} \sum_{k\in U, k\neq k'} \varphi^j_{kk'} \cdot (t_{kk'} \cdot \xi + d_{kk'} \cdot \tau) + \sum_{k\in U} (S_k \cdot \xi + V_k \cdot \mu) \right) \cdot CE \tag{41}$$

In the objective function, the coefficient λ could be set at $0.1, 0.2, \ldots, 1.0$, with a step size of 0.1. This step could be adjusted manually according to the problem scale and operator's preference. By changing λ, different sets of solutions can be obtained. The efficient frontier obtained by changing λ from 0.1 to 1.0 for an instance with 55 YCs and 110 task groups is shown in Fig. 13. From this figure, we can find that the energy cost gradually becomes higher, and the delay cost gradually becomes lower with the coefficient λ varying from 0.1 to 1.0. Therefore, along this frontier, the solutions in the top-left corner are energy-saving oriented, while the ones in the lower-right corner are time-saving oriented.

Among these solutions, there is no ultimate solution with the minimum energy consumption and the minimum completion delay simultaneously. If the container terminal operators pay more attention to environmental protection, the energy-saving oriented strategy may be adopted; otherwise the time-saving oriented strategy should be the main concern for the operators.

6.3. Actual case analysis

In this section, the proposed method is tested using one-week's real data obtained from a container terminal in Tianjin port. The results from the proposed method are compared with those from the practical results. The comparisons are shown in Fig. 14. From this figure, it can be observed that the proposed method performs much better than the practice in terms of the delay cost, the energy cost and the total cost. The average total cost obtained from the proposed method is only approximately 76.53% of that of practical



(a) Delay cost (b) Energy cost (c) Total cost

Fig. 14. The scheduling results of the proposed method in the actual test case.

results, the average energy cost obtained from the proposed method is only approximately 74.47% of that of practical results, and the average delay cost obtained from the proposed method is only approximately 84.31% of that of practical results. In sum, these results demonstrate the applicability of the proposed optimization method to practical container terminal YC scheduling.

7. Conclusions

Since container terminals are facing the pressure to reduce pollutant emission and protect the environment, energy-saving has become one of the main goals for container terminals. This paper proposes a YC scheduling problem considering energy-saving. The optimal trade-off between time-saving and energy-saving for YC scheduling is the objective of this paper. By comparing with other research in this area, the major contributions are as follows:

(1) Traditional methods on YC scheduling generally aim to improve the efficiency and do not take into consideration energy-saving. This paper provides a novel idea that considering the trade-off between energy-saving and time-saving in the YC scheduling problem. Energy-saving is part of the optimization objective.
(2) The YC scheduling problem is converted into a vehicle routing problem with soft time windows, which makes the YC scheduling problem easier to formulate. A MIP model is formulated, whose two objectives are the minimization of the total delay for all task groups and total energy consumption of all YCs. Furthermore, we use cost functions to combine the two objectives.
(3) An integrated simulation optimization method is developed for solving the YC scheduling problem, where the simulation is designed for evaluation and the optimization algorithm is designed for global search. The optimization algorithm integrates a GA and a PSO algorithm, where the GA is used for global search and PSO is used as a local search method to identify potential better solutions in each generation. Numerical experiments show that the proposed method is capable of solving YC scheduling problems with different sizes in short time.

In this paper, the arriving time and handling volume of each task group are deterministic. However, in reality they may fluctuate and cannot be predicted accurately. Therefore, a valuable future research direction is extending the YC scheduling problem to capture uncertainty.

Acknowledgements

This work is sponsored by National Natural Science Foundation Project (71101090), Shanghai Top Academic Discipline Project-Management Science & Engineering, Shanghai Yangfan Program (14YF1411200), Shanghai Municipal Education Commission Project (13YZ080 and 14YZ112), Doctoral Fund of the Ministry of Education (20133121110001) and Shanghai Maritime University Research Project (20120102). We also thank anonymous referees and the editor-in-chief.

References

[1] J. Akerman, M. Hojer, How much transport can the climate stand? – Sweden on a sustainable path in 2050, Energy Policy 34 (14) (2006) 1944–1957.
[2] C. Nanaki, E. Koroneos, Environmental assessment of the Greek transport sector, Energy Policy 35 (11) (2007) 5422–5432.
[3] E. Zervas, C. Lazarou, Influence of European passenger cars weight to exhaust CO_2 emissions, Energy Policy 36 (1) (2008) 248–257.
[4] A.H. Ja, S.-W. Cho, M.-S. Pak, Fuel consumption within cargo operations at the port industry – a simulation analysis on the case of S Port company in the UK, Asian J. Shipp. Log 28 (2) (2012) 227–254.
[5] APM (A.P. Moller), APM terminals to retrofit and electrify RTG fleet worldwide, 2011 <http://www.apmterminals.com> (accessed 21.02.12).
[6] W. Yan, Y.F. Huang, D.F. Chang, J.L. He, An investigation into knowledge-based yard crane scheduling for container terminals, Adv. Eng. Inform. 25 (3) (2011) 462–471.
[7] J.X. Cao, D.-H. Lee, J.H. Chen, Q.X. Shi, The integrated yard truck and yard crane scheduling problem: benders' decomposition-based methods, Transport. Res. E-Log. 46 (3) (2010) 344–353.
[8] D. Steenken, S. Voβ, R. Stahlbock, Container terminal operation and operations research – a classification and literature review, OR Spectrum 26 (1) (2004) 3–49.
[9] R. Stahlbock, S. Voβ, Operations research at container terminals: a literature update, OR Spectrum 30 (1) (2008) 1–52.
[10] Y.H.V. Lun, Green management practices and firm performance: a case of container terminal operations, Resour. Conserv. Recy 55 (2011) 559–566.
[11] H. Geerlings, R.V. Duin, A new method for assessing CO_2-emissions from container terminals: a promising approach applied in Rotterdam, J. Clean. Prod. 19 (2011) 657–666.
[12] C.-H. Liao, P.-H. Tseng, K. Cullinane, C.-S. Lu, The impact of an emerging port on the carbon dioxide emissions of inland container transport: an empirical study of Taipei port, Energy Policy 38 (2010) 5251–5257.
[13] C.-C. Chang, C.-M. Wang, Evaluating the effects of green port policy: case study of Kaohsiung harbor in Taiwan, Transport. Res. D-Tr. E. 17 (3) (2012) 185–189.
[14] J. Li, X. Liu, B. Jiang, An exploratory study on low-carbon ports development strategy in China, Asian J. Shipp. Log 27 (1) (2011) 91–111.
[15] D.F. Chang, Z.H. Jiang, W. Yan, J.L. He, Integrating berth allocation and quay crane assignments, Transport. Res. E-Log. 46 (6) (2010) 975–990.
[16] Y.Q. Du, Q.S. Chen, X.W. Quan, L. Long, R.Y.K. Fung, Berth allocation considering fuel consumption and vessel emissions, Transport. Res. E-Log. 47 (6) (2011) 1021–1037.
[17] S. Esmemr, I.B. Ceti, O. Tuna, A simulation for optimum terminal truck number in a Turkish port based on lean and green concept, Asian J. Shipp. Log 26 (2) (2010) 277–296.
[18] M.M. Golias, G.K. Saharidis, M. Boile, S. Theofanis, M.G. Ierapetritou, The berth allocation problem: optimizing vessel arrival time, Maritime Econ. Logis. 11 (4) (2009) 358–377.
[19] J.L. He, W.M. Zhang, Y.F. Huang, W. Yan, A simulation optimization method for internal trucks sharing assignment among multiple container terminals, Adv. Eng. Inform. 27 (4) (2013) 598–614.
[20] J.F. Alvarez, T. Longva, E.S. Engebrethsen, A methodology to assess vessel berthing and speed optimization policies, Maritime Econ. Logis. 12 (4) (2010) 327–346.
[21] K.Y. Kim, K.H. Kim, An optimal routing algorithm for a transfer crane in port container terminals, Transport. Sci. 33 (1) (1999) 17–33.
[22] K.Y. Kim, K.H. Kim, Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals, Nav. Res. Log. 50 (5) (2003) 498–514.
[23] A. Narasimhan, U.S. Palekar, Analysis and algorithm for the transtainer routing problem in container port operation, Transport. Sci. 36 (1) (2002) 63–78.
[24] W.C. Ng, K.L. Mak, Yard crane scheduling in port container terminals, Appl. Math. Model 29 (3) (2005) 263–275.
[25] C. Zhang, Y.W. Wan, J.Y. Liu, R. Linn, Dynamic crane deployment in container storage yards, Transport. Res. B-Meth. 36 (6) (2002) 537–555.
[26] R.K. Cheung, C. Li, W. Lin, Interblock crane deployment in container terminals, Transport. Sci. 36 (1) (2002) 79–93.
[27] W.C. Ng, Crane scheduling in container yards with inter-crane interference, Eur. J. Oper. Res. 164 (1) (2005) 64–78.
[28] L. Chen, N. Bostel, P. Dejax, J.G. Cai, L.F. Xi, A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, Eur. J. Oper. Res. 181 (1) (2007) 40–58.
[29] Z. Cao, D.-H. Lee, Q. Meng, Deployment strategies of double-rail-mounted gantry crane systems for loading outbound containers in container terminals, Int. J. Prod. Econ. 115 (1) (2008) 221–228.
[30] W.K. Li, Y. Wu, M.E.H. Petering, M. Goh, R. de Souza, Discrete time model and algorithms for container yard crane scheduling, Eur. J. Oper. Res. 198 (1) (2009) 165–172.
[31] M.E.H. Petering, K.G. Murty, Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal, Comput. Oper. Res. 36 (2009) 711–1725.
[32] J.L. He, D.F. Chang, W.J. Mi, W. Yan, A hybrid parallel genetic algorithm for yard crane scheduling, Transport. Res. E-Log. 46 (1) (2010) 136–155.
[33] D.F. Chang, Z.H. Jiang, W. Yan, J.L. He, Developing a dynamic rolling-horizon decision strategy for yard crane scheduling, Adv. Eng. Inform. 25 (3) (2011) 485–494.
[34] R.J. Linn, J.Y. Liu, Y.W. Wan, C.Q. Zhang, K.G. Murty, Rubber tired gantry crane deployment for container yard operation, Comput. Ind. Eng. 45 (3) (2003) 429–442.
[35] H.J. Carlo, I.F.A. Vis, K.J. Roodbergen, Storage yard operations in container terminals: literature overview, trends, and research directions, Eur. J. Oper. Res. 235 (2) (2014) 412–430.
[36] C. Archetti, D. Feillet, M. Gendreau, M.G. Speranza, Complexity of the VRP and SDVRP, Transport. Res. C-Emer. 19 (5) (2011) 741–750.

[37] Y. Marinakis, M. Marinaki, A hybrid genetic – particle swarm optimization algorithm for the vehicle routing problem, Expert. Syst. Appl. 37 (2) (2010) 1446–1455.
[38] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35 (1987) 254–265.
[39] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, John Wiley & Sons Inc., New York, 1997.
[40] D.-H. Lee, H.Q. Wang, L.X. Miao, Quay crane scheduling with non-interference constraints in port container terminals, Transport. Res. E-Log. 44 (2008) 124–135.
[41] M.F. Tasgetiren, M. Sevkli, Y.-C. Liang, G. Gencyilmaz, Particle swarm optimization algorithm for single machine total weighted tardiness problem, in: Proceedings of the Congress on Evolutionary Computation (CEC2004), vol. 2, 2004, pp. 1412–1419.
[42] M. Clerc, J. Kennedy, The particle swarm: explosion, stability and convergence in a multi-dimensional complex space, IEEE Trans. Evolut. Comput. 6 (2002) 58–73.
[43] Y. Marinakis, G.-R. Iordanidou, M. Marinaki, Particle swarm optimization for the vehicle routing problem with stochastic demands, Appl. Soft. Comput. 13 (2013) 1693–1704.