Hindawi
Discrete Dynamics in Nature and Society
Volume 2019, Article ID 2615154, 14 pages
https://doi.org/10.1155/2019/2615154

Research Article

A Metaheuristic Algorithm to Transporter Scheduling for Assembly Blocks in a Shipyard considering Precedence and Cooperating Constraints

Ning-Rong Tao,[1] Zu-Hua Jiang,[2,3] Jian-Feng Liu,[4] Bei-Xin Xia,[5] and Bai-He Li[2,3]

[1]*College of Science, Technology and Engineering, Shanghai Ocean University, Shanghai 201306, China*
[2]*Collaborative Innovation Center for Advanced Ship and Deep-Sea Exploration, Shanghai 20240, China*
[3]*School of Mechanical Engineering, Shanghai Jiaotong University, Shanghai 200240, China*
[4]*Shanghai Waigaoqiao Shipbuilding Co. Ltd., Shanghai 200137, China*
[5]*School of Management, Shanghai University, Shanghai 200444, China*

Correspondence should be addressed to Bei-Xin Xia; bxxia@shu.edu.cn

Received 3 November 2018; Accepted 31 December 2018; Published 15 January 2019

Guest Editor: Xinchang Wang

Special vehicles named flat transporters are used to deliver heavy ship assembly blocks in shipyards. Because each movement of assembly blocks among workshops needs transporters and the transportations are time-consuming, the scheduling of transporters is important for maintaining the overall production schedule of assembly blocks. This paper considers an optimization transporter scheduling problem for assembly blocks. The objective is to minimize logistics time, which includes empty travel time of transporters and waiting time and delay time of block tasks. Considering time windows of ship blocks, carrying capacity of transporters, and precedence relationships of tasks, a mathematical model is proposed. A hybrid topological graph is used to denote precedence and cooperating relationships of tasks. A metaheuristic algorithm based on the hybrid topological graph and genetic algorithm and Tabu search is proposed. The performance of the algorithm was evaluated by comparing the algorithm to optimal result in small-sized instances and several strategies in large-sized instances. The results showed the efficiency and effectiveness of the proposed algorithm.

1. Introduction

A ship hull is constructed by hundreds of assembly blocks. Each block is over 100 tons and 15 meters (length) by 15 meters (width) by 5 meters (height) [1, 2]. Some blocks weigh even more than 500 tons and are much bigger. Normally, before being assembled into a ship hull, a block would be moved over ten times among different workshops like assembly workshop, painting workshop, outfitting workshop, and stockyards for temporary storage. Because of considerable size and weight of blocks, a specially manufactured flat transporter (hereinafter, "transporter") is used as the transportation vehicle, which is a multiple-axle vehicle with hydraulic jack lifts so it can jack up a big and heavy block.

In the transporter scheduling problem for assembly blocks, a transportation operation includes picking up a block, moving it from one workshop/place to another through an accessible path, and unloading it at a proper position. Each transportation operation is quite time-consuming and costly. And since a few huge blocks may extend the load capacity of any available transporter in a shipyard, several transporters should work simultaneously for transporting it, which is called **cooperating transportation**. But lack of well scheduling of transporters causes delay on executing transporting tasks and waste on waiting for preassigned cooperating transporters. It would result in traffic jam and production delay in a shipyard. Actually, in most shipbuilding companies, one of the major management issues is how to efficiently manage assembly blocks in a shipyard, which contains utilizing efficiently the scarce resources such as the block stockyards and transportation vehicles. Ship block transportation problems are crucial issues to address in

reducing the construction cost and improving the productivity of shipyards [3].

Only a few recent studies have addressed the problem of assembly block transportation scheduling. Lee et al. [4] first studied the scheduling of single-type transporters for block transportation. Actually, there are various transporters according to the deadweight. Thus, Roh and Cha [5] studied the block transportation scheduling problem with multitype transporters. Their objective was to minimize the travel distance without loading and interference between transporters while satisfying the constraints on allowable transportation weights of the transporters. Kim and Joo [6] considered a similar block transportation scheduling problem for heterogeneous transporters with different weight capacities. Their objective was to minimize logistics times, including delay time, tardy time, and empty transporter travel time. Joo and Kim [7] expanded the research of reference paper [4] by considering the scheduling problem of block transportation under a delivery restriction to determine when and by which transporter each block is delivered from its source plant to its destination plant. Wang et al. [3] proposed a greedy algorithm with the same objective as Kim and Joo's researches.

Actually, in realistic shipbuilding environments, most workshops in shipbuilding have a schedule of blocks and the door is not big enough to afford several blocks come in/out at the same time. For example, the painting workshop has several sand washing rooms in which at most two blocks can be washed together. Usually they would move out in one day, and the back one must wait for the front one has been loaded by a transporter and leaves. And when there are two blocks in a workshop or in a temporary storage yard ready for transporting, one of them may be on the way of the other because of their big sizes. This is quite often in shipbuilding. In the above studies, the release/due time and the precedence relationship between the assembly blocks was not considered. Hu et al. [8] took account of the resource available time for use and tardiness minimization in a task assignment problem. Park and Seo [9] mentioned the precedence constraints between assembly blocks in their study. Zhang el al. [10] built a model of the assignment and paths of the inbound and outbound objects to shipyards considering the order of blocks. However, the research was just about single-type transporters. It may be not conformed to most shipbuilding companies, which needs to be studied on multitype transporters [7].

The block transportation scheduling is similar to a multiple travelling salesman problem with time windows (m-TSPTW) by regarding each block as a location and transporters as travelling salesmen [6, 11], when the type of transporters is the same and all blocks are predetermined to be delivered by a specific transporter. The m-TSPTW is known as a NP-hard problem. Laporte and Osman [12], Crainic and Laporte [13], and Chao [14] surveyed well known TSP, m-TSP, and general vehicle routing problems. Zhang and Moon [15] proposed an m-TSPTW to model a container truck transportation problem and developed a cluster method and a reactive Tabu search algorithm to solve the problem. Sterzik and Kopfer [16] proposed a Tabu search algorithm for the inland container transportation problem to control the movement of full and empty containers.

The block transportation scheduling problem can also be transferred to a scheduling problem for parallel machines with sequence-dependent setup times and precedence constraints [17]. The parallel machine scheduling problem is one of the classical problems in production systems [18]. In our problems, the machines correspond to the transporter, and the job is the transport of an assembly block. This type of scheduling also belongs to the class of NP-hard problems [9].

However, there is a major difference between the above studies and our study, which is the situation of several transporters working simultaneously for transporting one block. The problem is kind of multivehicle and one-cargo transportation problem where a cargo is simultaneously loaded by several vehicles [19]. Dohn, Rasmussen, and Larsen [20] and Drexl [21] pointed out that synchronization problems are highly relevant in routing practice. Salazar-Aguilar, Langevin, and Laporte [22] introduced a synchronized arc routing problem for snow plowing operations. The street segments with two or more lanes in the same direction are plowed simultaneously by synchronized vehicles. Salazar-Aguilar, Langevin, and Laporte [23] introduced the paint vehicle synchronization problem that several capacitated vehicles painted lines on the roads with a tank vehicle replenishing the painting vehicles. The routes and schedules for the painting and tank vehicles were optimized by synchronizing the painting and replenishment operations. Rousseau, Gendreau, and Pesant [24] solved a real-time vehicle dispatching problem where some customers were serviced with multiple resources synchronously. Derigs and Pullmann [25] introduced the multidepot multitrip VRP with order incompatibilities subjected to interroute synchronization constraints. Hu and Wei [19] studied the multivehicle and one-cargo transportation problem; however, they did not put the problem into a realistic situation; for example, they did not mention priorities of tasks, release time, and due time of tasks or weight capacity of transporters.

On the basis of the above analysis, this paper deals with a transporter scheduling problem for assembly blocks. In the problem, we consider "multivehicle and one-cargo transportation" as well as the constraints of precedence relationship and the release/due time of tasks. The objective is to minimize total logistics time including empty travel time of transporters, delay time, and waiting time of block tasks.

2. Mathematical Model

2.1. Problem Description. The transporter scheduling problem for assembly blocks in a shipyard includes assigning all of block transportation tasks to multiple transporters, sequencing the tasks for each transporter, and determining the start time to fulfill each task.

The problem is studied based on the following conditions.

(1) Since each task would be released by one workshop at a certain time and needed by another workshop at some time, which means the task cannot be operated before its release time, and it is better to be finished before the due time, each task is given a time window [release time, due time].

Table 1: Block transportation tasks.

Task ID	Block ID	Weight (t)	Retrieve Place	Destination Place	Release Time (min)	Due Time (min)	Priority Task
1	B1	250	P1	P3	0	120	-
2	B2	450	P2	P3	30	150	-
3	B3	300	P3	P4	100	250	-
4	B4	200	P1	P2	0	120	1
5	B5	200	P3	P2	100	250	3
6	B6	350	P2	P3	0	180	-

Table 2: Preprocessed block transportation tasks.

Task ID	Block ID	Weight (t)	Retrieve Place	Destination Place	Release Time (min)	Due Time (min)	Priority Task	Synchronous Task
1	B1	250	P1	P3	0	120	-	-
2	B2'	**225**	P2	P3	30	150	-	**7**
3	B3	300	P3	P4	100	250	-	-
4	B4	200	P1	P2	0	120	1	-
5	B5	200	P3	P2	100	250	3	-
6	B6	350	P2	P3	0	180	-	-
7	**B2'**	**225**	**P2**	**P3**	**30**	**150**	-	**2**

(2) The precedence relationships between blocks are considered. The precedence constraint forced the one after started until the one before is loaded by the assigned transporter.

(3) Each transporter can take no more than one block at a time and must not exceed its deadweight.

(4) Each transportation operation must not be interrupted until it is finished.

(5) For the overweight block, it needs two transporters to work synchronously. The cooperating transporters must be the same type.

Here is an example with six transportation tasks in Table 1. Considering the precedence constrains, Task 4 cannot be started until Task 1 is loaded by a transporter, and Task 5 cannot be operated until Task 3 is loaded.

The weight capacities of three transporters are 250t, 250t, and 380t, respectively. It can be found from Table 1 that B2 weights 450t, exceeding the biggest weight capacity of transporters. Thus a feasible solution should assign two transporters to deliver B2 synchronously.

In this case, we create a virtual task. As in Table 2, the original Task 2 is divided into two synchronous tasks: Task 2 and Task 7.

Figure 1 gives an assignment and sequencing plan as an example, where Task 2 needs Transporters #01 and #02 cooperating to transport B2. Based on this sequencing and assignment plan together with the data in Table 2, the start time of each task can be determined.

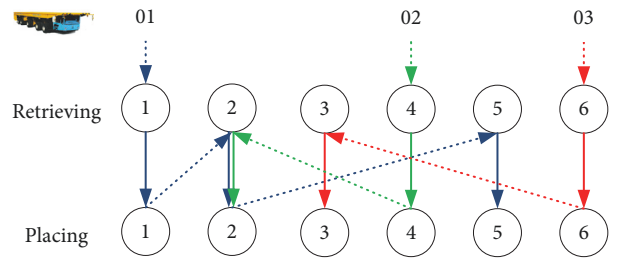

Figure 1: A sequencing and assignment plan for block transportation.

2.2. Model Formulation. A mathematical programming model can be formulated to the problem. The following notations are used:

n: total number of block transportation tasks

m: total number of transporters

w_i: block weight of block transportation task i

rt_i: release time of block transportation task i

dt_i: due time of block transportation task i

rp_i: retrieve place of block transportation task i

dp_i: destination place of block transportation task i

wc_k: weight capacity of transporter k

p_k: initial position of transporter k

vl_k: average speed of transporter k while transporting a block

ve_k: average speed of transporter k while it is empty

$D = \begin{bmatrix} d(1,1) & \cdots & d(1,p) \\ \vdots & \ddots & \vdots \\ d(p,1) & \cdots & d(p,p) \end{bmatrix}$: distance matrix of each pair of work places

lt_i: prepare time needed for loading block in block transportation task i

ut_i: prepare time needed for unloading block in block transportation task i

pr_{ij}: precedence relationship of tasks i and j; if $pr_{ij} = 1$, task j must wait at least until task i is loaded by the assigned transporter; if $pr_{ij} = -1$, task i must wait at least until task j is loaded by the assigned transporter; otherwise $pr_{ij} = 0$

sr_{ij}: cooperating relationship of tasks i and j; if $sr_{ij} = 1$, task i must be operated with task j synchronously; otherwise $sr_{ij} = 0$

M: a big positive number

Variables

x_i: start time of picking up task i

em_i: empty travel time to pick up task i

dl_i: delay time of task i

y_i^k: if $y_i^k = 1$, task i is operated by transporter k; otherwise $y_i^k = 0$

z_{ij}^k: if $z_{ij}^k = 1$, task j is operated by transporter k right before task i; otherwise $z_{ij}^k = 0$.

z_{0i}^k: if $z_{0i}^k = 1$, task i is the first task of transporter k; otherwise, $z_{0i}^k = 0$

Objective Function. The objective is to minimize total logistics time cost, defined as the weighted sum of empty travel time of transporters, delay time, and waiting time of block tasks.

$$\min f = w_1 \cdot \sum_{i \in n} em_i + w_2 \cdot \sum_{i \in n} dl_i + w_3 \cdot \sum_{i \in n} (x_i - rt_i) \quad (1)$$

Constraints. Each task must not be started after its release time rt_i. If the task is released, then the waiting time for being picked up is wa_i.

$$x_i \geq rt_i, \quad for\ \forall i \quad (2)$$

Constraints (3) give the empty travel time of each task. For task i, the empty travel time is the transporter assigned to task i moving from its last determined place to the retrieve place of task i. The last determined place of the transporter assigned to task i is $\sum_{j \in n, j \neq i} z_{ji}^k \cdot dp_j$, but if task i is the first task of transporter k, the last determine place is its initial place $z_{0i}^k \cdot p_k$.

$$em_i = d\left(\sum_{k \in m} z_{0i}^k \cdot p_k + \sum_{k \in m} \sum_{j \in n, j \neq i} z_{ij}^k \cdot dp_j, rp_i\right) \times \sum_{k \in m} \frac{y_i^k}{ve_k}, \quad for\ \forall i \quad (3)$$

Constraints (4) give the delay time, i.e., the time exceeding the due time of tasks. Wherein $d(rp_i, dp_i) \times \sum_{k \in m} (y_i^k / vl_k)$ is with load travel time of task i,

$$dl_i \geq x_i + lt_i + d(rp_i, dp_i) \times \sum_{k \in m} \frac{y_i^k}{vl_k} + ut_i - dt_i, \quad for\ \forall i \quad (4)$$

$$dl_i \geq 0, \quad for\ \forall i \quad (5)$$

For each pair of tasks, if one has a priority to the other, the latter one must be operated after the prior one is loaded by its transporter.

$$x_i - x_j + lt_i \geq M \cdot (1 - pr_{ij}), \quad for\ \forall i, j\ i \neq j \quad (6)$$

If any two tasks are operated by the same transporter, their start time must follow Constraints (7), wherein $d(dp_i, rp_j) \times \sum_{k \in m} (y_j^k / ve_k)$ is empty travel time from the last task i to task j.

$$x_i + lt_i + d(rp_i, dp_i) \times \sum_{k \in m} \frac{y_i^k}{vl_k} + ut_i + d(dp_i, rp_j) \times \sum_{k \in m} \frac{y_j^k}{ve_k} - x_j \leq M \cdot (1 - z_{ji}^k), \quad for\ \forall i, j, k \quad (7)$$

Constraints (8) ensure that each task can be executed only once.

$$\sum_{k \in m} y_i^k = 1, \quad for\ \forall i \quad (8)$$

Constraints (9) ensure that each transporter has at most only one first task.

$$\sum_{i \in n} z_{0i}^k \leq 1, \quad for\ \forall k \quad (9)$$

Constraints (10) ensure that each transporter can take only one task at a time.

$$\sum_{j \in n, j \neq i} z_{ij}^k + z_{0i}^k = y_i^k, \quad for\ \forall i, k \quad (10)$$

Constraints (11) ensure that each task must satisfy the assigned transporter's weight capacity.

$$\sum_{k \in m} y_i^k \cdot wc_k \geq w_i, \quad for\ \forall i \quad (11)$$

Constraints (12) ensure that tasks with cooperating constraint must be operated synchronously. And Constraints (13) ensure that the cooperating tasks must be operated by the same type of transporters.

$$sr_{ij} \cdot (x_i - x_j) = 0, \quad for\ \forall i, j \quad (12)$$

$$sr_{ij} \cdot \left(\sum_{k \in m} y_i^k \cdot wc_k - \sum_{l \in m} y_j^l \cdot wc_l\right) = 0, \quad for\ \forall i, j \quad (13)$$

The mathematical model is a mixed integer programming model. It can be solved by an optimization tool like CPLEX. But in realistic situations, the problem size makes it hard to

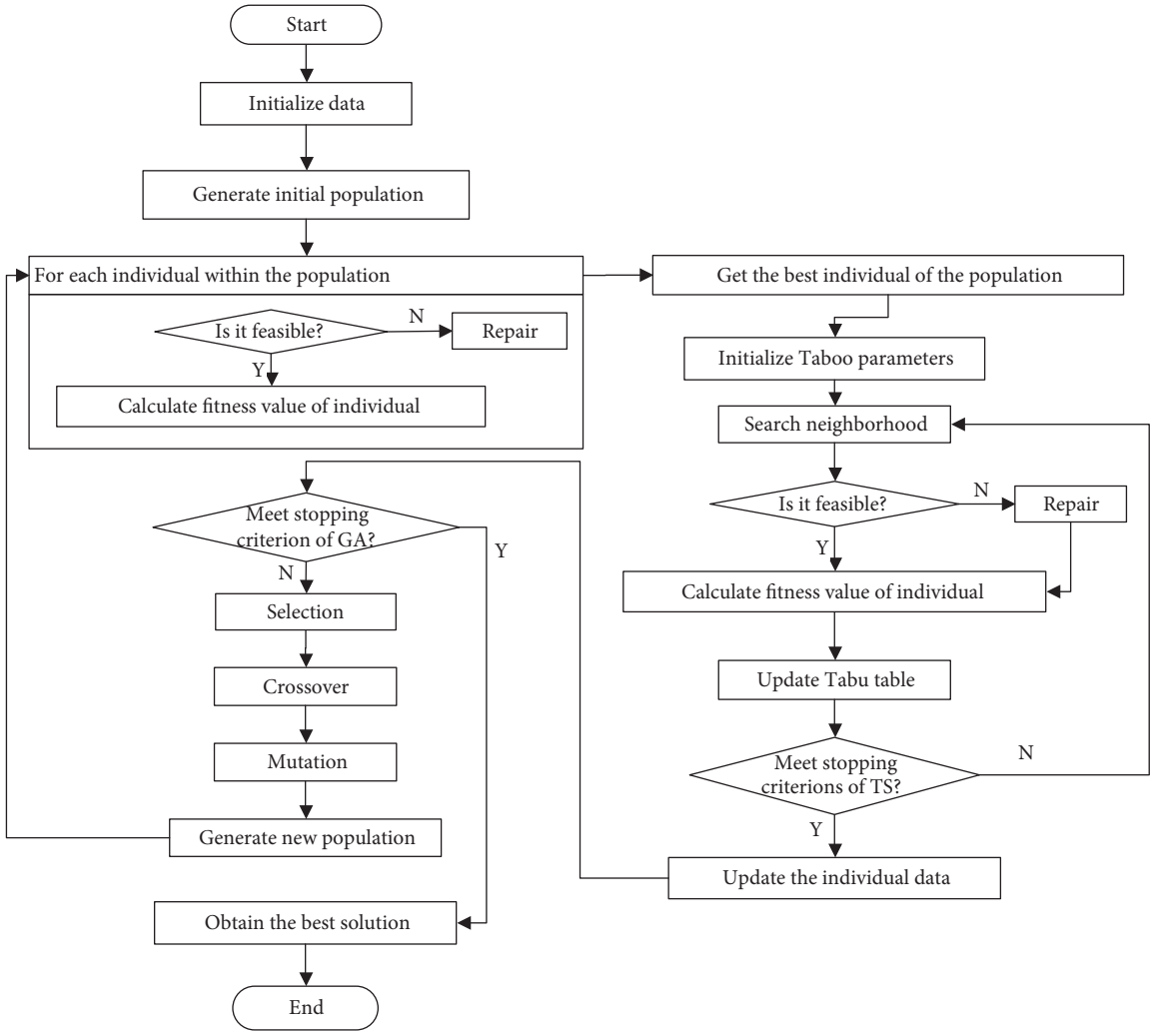

Figure 2: Flow chart of the proposed algorithm of block transportation scheduling.

get a good solution in reasonable computing time of CPLEX. Thus, we propose a metaheuristic algorithm to solve the transporter scheduling problem for assembly blocks.

3. Metaheuristic Algorithm

In this section, a basic flow of the proposed algorithm is first given. The proposed algorithm frame is based on genetic algorithm (GA), and the local search process uses Tabu search (TS). Several scheduling strategies are introduced and involved in the heuristic algorithm. Since the relationships among tasks become more complex as the problem size is growing, the algorithm cannot ensure the feasibility of solution of each chromosome. We propose a chromosome repair method through a topologic structure of task relationships.

3.1. General Framework. The main concept of the proposed metaheuristic algorithm is given in Figure 2.

The following are the main context of the proposed algorithm.

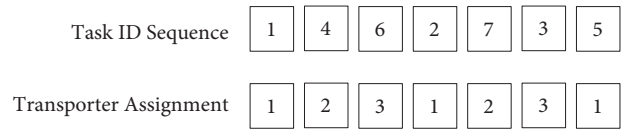

Figure 3: Chromosome representation in block transportation scheduling.

3.1.1. Initial Population. Usually, initial population can be randomly generated. The quality of initial population has some influence on the quality of algorithm. Thus, this paper takes half random generated individuals and the other half are obtained by several strategies which are described in Section 3.2.2.

3.1.2. Encoding Scheme. The genes of the chromosomes are designed as two arrays based on tasks assignment and sequencing information. The length of each chromosome is the number of tasks. Figure 3 is one chromosome of

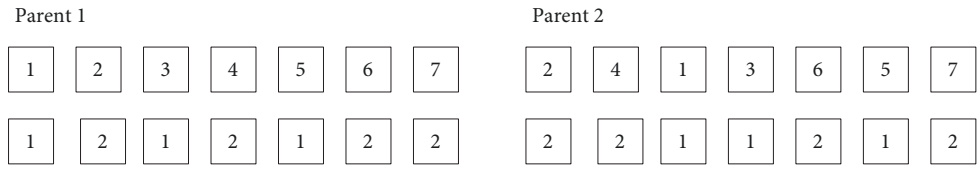

Figure 4: Two parents as an example.

the example shown in Table 2. It has two parts: one is the sequence of task ID, and the other one is transporter assignment information of each task. This sequence is not the order in which the tasks are executed by each transporter. For example, the first column (1,1) means task 1 is assigned to transporter 1 and it is scheduled first. The final order of tasks is obtained after decoding.

3.1.3. Decoding Scheme. The aim of decoding is to translate an individual to a full solution which can be evaluated by fitness function. The full solution is a worksheet telling the workers when and where each task is executed by which one or two transporters.

Based on the assignment and sequencing information in a chromosome together with precedence constraints, cooperating constraints, time window constraints etc., we can decode a chromosome by applying time scheduling strategy to the corresponding solution as follows.

Step 1. Obtain the data of sequencing and assignment list of tasks, and transporters' initial position and available time span, etc. Set counter $h = 1$; then task i =sequencing(h), and transporter k =assignment(h)

Step 2. Decide the start time of task i.

Step 2.1. Check Matrix PR; obtain its priority-task set where $ps_{ij} = -1,\ j \in n$.

Step 2.2. Obtain position p_k and release time et_k of transporter k.

Step 2.3. Calculate the start time st_i for task i:

$$st_i = \max\left\{rt_i, et_k + \frac{d(p_k, rp_i)}{ve_k}, st_j + lt_j\right\}, \quad j \in tb_i \quad (14)$$

Step 3. Update new position $p_k = dp_i$; the transporter's idle time $et_k = st_i + lt_i + d(rp_i, dp_i)/vl_k + ult_i$.

Step 4. Check Matrix SR; if $sr_{ij} = 1,\ j \in n$, obtain task i's cooperating task j. If Row i does not have any cooperating task, go to Step 5.

Step 4.1. If task j is executed before task i, then redefine the start time of these two tasks. Let $st_i = st_j = \max(st_i, st_j)$, with $\Delta t = |st_i - st_j|$; redefine the related time, including et_k.

Step 4.2. If task j is executed after task i, go to Step 5.

Step 5. If $h = n$, the end, let $h = h + 1$ and back to Step 2.

3.1.4. Fitness Evaluation. The objective function described in Section 2.1 is used to evaluate an individual's performance. The lower the value of the objective function, the higher the fitness of the individual.

3.1.5. Selection. Based on roulette selection strategy, two randomly individuals are selected from the population and the one with higher fitness value is retained. Repeat this process until obtaining required number of individuals. Suppose Figure 4 is two selected parents. Then the offspring are generated from them.

3.1.6. Offspring Generation. We divide the operators into two categories. Each of them contains crossover and mutation processes.

(i) Assignment operators.

(ii) Sequencing operators.

Assignment operators only change the assignment property of chromosomes; i.e., the sequencing of tasks is preserved in the offspring. Assignment crossover generates the offspring by exchanging the transporter assignment information of subchromosomes between two parents, as shown in Figure 5(a). The position of the subchromosome is randomly selected. And assignment mutation randomly swaps the transporter assignment information of two genes within a single parent, as shown in Figure 5(b).

Sequencing operators only change the sequence of the tasks in the parent chromosomes. The partially matched crossover (PMX), which is the best performing crossover operator for the scheduling problems [26, 27], is used. The procedures of PMX are listed as follows.

Step 1. Randomly select two positions on the parents.

Step 2. Exchange two subchromosomes between two parents.

Step 3. Determine the mapping relationship between two mapping sections; namely, map each gene in one mapping section with the corresponding gene in another mapping section on the same position.

Step 4. Repair the offspring by replacing the reduplicate genes beyond the mapping section according to the mapping relationship.

An example of PMX procedure is shown in Figure 6(a). After Step 2, some task ID may appear multiple times. So in Step 3, a mapping relationship is constructed and the repaired offspring are obtained finally.

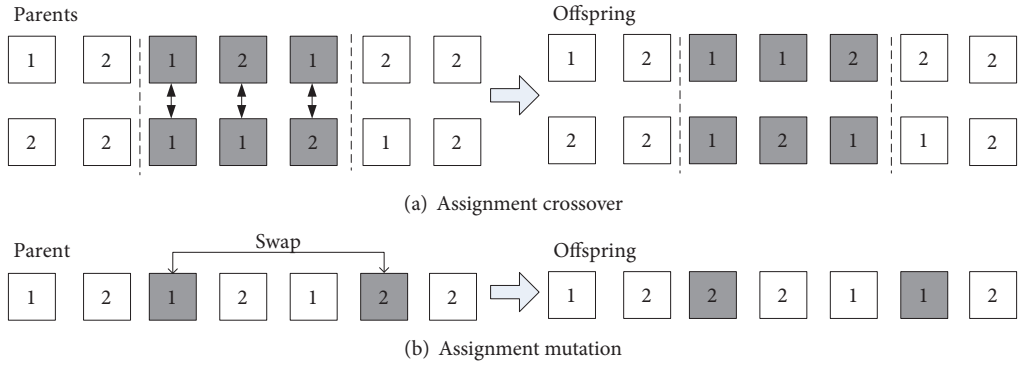

Figure 5: Example of assignment operators.

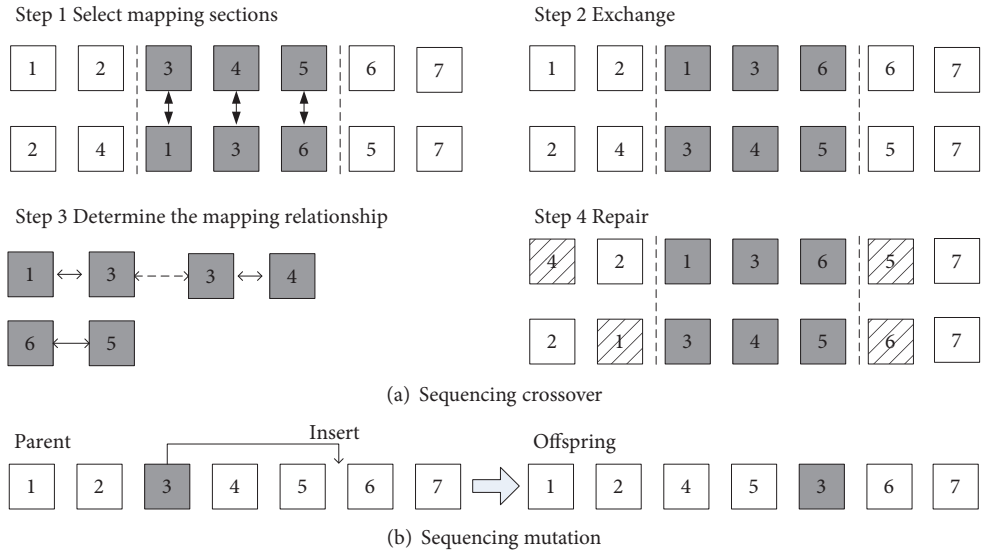

Figure 6: Example of sequencing operators.

And here an insertion operator acts as a mutation. As shown in Figure 6(b), it randomly selects a gene from the individual and inserts it back to a random position.

3.1.7. Tabu Search Subalgorithm. For the best individual in each population, use Tabu search subalgorithm to optimize it. It starts with an empty Tabu table. Two counters itr_1 and itr_2 are used to trace the number of total iterations and the number of iteration performed without improvement, respectively. The subalgorithm terminates when there is no improvement over the best solution obtained after a certain number of iterations (*NonimpIter*), or when the total number of iterations reaches a predetermined value (*MaxIter*).

Neighbourhood of an individual is searched by performing the following steps: (1) choose a gene at position; (2) transfer the transporter randomly to another one; (3) check the new individual's feasibility.

A Tabu table is a short time memory of the last several movements. It is applied to have a trace of the evolution of the search to prevent cycling. It is denoted by a $n \times n$ matrix. Record every transformation on the respective element in the matrix. The Tabu table is updated by keeping only the latest certain number of elements.

Suppose f_{best} is the best individual so far; set an empty Tabu table and two counters equal to 0. Repeat the following steps until one of the two counters reaches the present value.

Step 1. Set $itr_1 := itr_1 + 1$, $itr_2 := itr_2 + 1$.

Step 2. Generate the neighbourhood of f_{best}.

Step 3. For each one of the neighbourhood f_{now}, calculate the fitness value. If the fitness value $E(f_{now}) < E(f_{best})$, then replace the individual f_{best} by f_{now} and set $itr_2 = 0$.

Step 4. Update the Tabu table by recording the neighbourhood search. Go to Step 1 until $itr_1 = MaxIter$ or $itr_2 = NonimpIter$.

3.1.8. Stopping Criterion. The algorithm ends when the maximal number of generations (*GAMaxIter*) is reached, and the best individual, together with the corresponding scheduling

and assignment solutions, is given as output. The solution after decoding the individual chromosome is the optimal solution.

3.2. Strategies Based on Topological Relationships. The problem actually includes three parts, which are assigning the tasks to a special transporter, sequencing the tasks for each transporter, and determining the start time of each task. The relationship constraints among tasks are presented by a topological structure. The topological map would be changed along with solving process. Then several strategies gained from experience are presented for initial solutions.

3.2.1. Topological Description for Tasks. Topological structure is one of the basic methods for scheduling problem. Like most scheduling problems, the precedence constraint is represented by one-way arc. Besides precedence constraints, cooperating constraints are considered in the problem. Here, we use undirected arcs to represent cooperating constraints between tasks. Thus, the relationships among tasks in Table 2 can be represented in Figure 7. For any pair of two tasks which is linked by a one-way arc, the before task should be loaded before the after one. While for any pair of two tasks which is linked by undirected arc, the two tasks must be operated synchronously. These relationships must be fully met in order to guarantee a solution is feasible.

Matrixes *PR* and *SR* are used to express the topological relationships of tasks. For the example in Table 2, $pr_{14} = 1$ means Task 1 is prior to Task 4; oppositely, $pr_{41} = -1$ and, for the same reason, $pr_{35} = 1$, $pr_{53} = -1$; $sr_{27} = 1$ means Task 2 must be operated synchronously with Task 7, and $sr_{72} = 1$.

$$PR = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$SR = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

3.2.2. Strategies for Initial Solution. The problem actually includes three parts, which are assigning the tasks to a special transporter, sequencing the tasks for each transporter, and determining the start time of each task. The topological map would be changed along with solving process. The above two matrixes present the initial relationships of the problem.

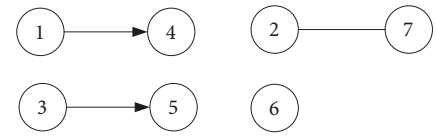

Figure 7: Topological map of block transportation tasks.

Referring to the matrixes, a negative element in a row of matrix *PR* means the cooperating task has a prior task which is not assigned yet. When Task 1 has started, delete Row 1 and make the negative elements in Column 1 to 0. Then, in the new matrix, Row 4 has no negative element. That means Task 4 is executable.

When assigning transporters, there are several rules considered here.

Rule 1. Select the transporter which reaches the retrieving place as early as possible.

Rule 2. Select the transporter which is idle firstly.

The strategies of initial solution are as follows.

Step 1. Choose a row without zero from *PR*, and add the cooperating task to executable task set.

Step 2. Select a task i from executable task set which has the earliest release time.

Step 2.1. Select a transporter which meets the task's weight constraint. If there is more than one selections, use Rule 1 or Rule 2 to choose one from them.

Step 2.2. Calculate the start time of task i, $x_i = \max(t_k + d(p_k, rp_i)/ve_k, rt_i)$.

Step 3. Check Matrix *SR*. If task i does not have any cooperating task, go to Step 4; otherwise repeat Step 2.1 and Step 2.2 for the cooperating task j, calculate the start time $x_j = \max(t_{k'} + d(p_{k'}, rp_j)/ve_{k'}, rt_j)$, and let $x_i = x_j = \max(x_i, x_j)$.

Step 4. Update the transporter's idle time $et_k = x_i + lt_i + d(rp_i, dp_i)/vl_k + ut_i$ and new position $p_k = dp_i$.

Step 5. Delete task i from executable task set and change all the negative value in column i be zero in Matrix *PR*; go back to Step 1.

We take the example in Section 2.1; suppose the distance matrix D is given as follows.

$$D = \begin{bmatrix} 0 & 400 & 500 & 700 \\ 400 & 0 & 800 & 600 \\ 500 & 800 & 0 & 300 \\ 700 & 600 & 300 & 0 \end{bmatrix} \quad (16)$$

In *Step 1*, *executable task set* $=\{1, 2, 3, 6, 7\}$. We first select Task 1 and FT-1, calculate the start time of operating Task 1,

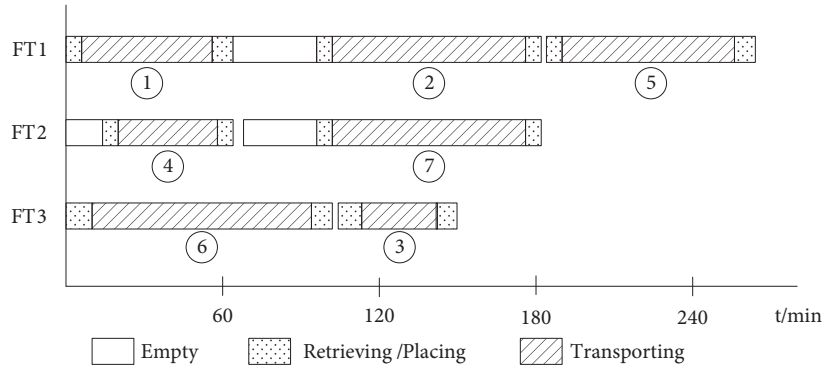

Figure 8: Gantt chart of a solution of block transportation scheduling.

i.e., x_1, and get the idle time of FT-1, i.e., et_1. Then, delete Task 1 from the set. Since $pr_{14} = 1$, release Task 4 and update the set={2, 3, 6, 7, 4}. Go to Step 2 and execute Task 6 and Task 4. Select Task-2. Since $sr_{27} = 1$, select Task-7 too. Assign FT-1 and FT-2 to Task-2 and Task-7. Calculate the respective start times x_2 and x_7 and choose $\max(x_2, x_7)$ as the start time. Then, execute Task 3 and Task 5. Finally, a feasible solution is obtained. Figure 8 gives the Gantt chart of the solution.

3.3. Feasibility Analysis and Gene Repair. As we know, GA needs appropriate constraint handling method to get feasibility and applicability result. The most common methods are penalty function method and chromosome repair method. The penalty function method is hard to get feasible solution for complex constraints problem, while chromosome repair method is more appropriate by repair infeasible solutions. Thus, we use a chromosome repair method to repair the unfeasible offspring.

3.3.1. Feasibility Analysis. When dealing with complex constrained optimization problems, GA needs to take action to maintain the applicability and feasibility. Feasibility analysis is based on the constraints described in the mathematical model, which contains two main parts:

(i) whether satisfying the load transporters' capacity

It is checked through Constraints (11): $\sum_k y_i^k \cdot wc_k \geq w_i$, $for\ \forall i$:

(ii) whether satisfying the task relationships

An individual chromosome can be transferred into a topological diagram. The diagram is structured by nodes and directed and undirected lines. The following are the structure process:

(i) Nodes represent tasks. In Figure 9, ①-⑦ are seven tasks.

(ii) For two tasks with precedence relationship, use solid directed line.

(iii) For two tasks with cooperating relationship, use solid undirected line.

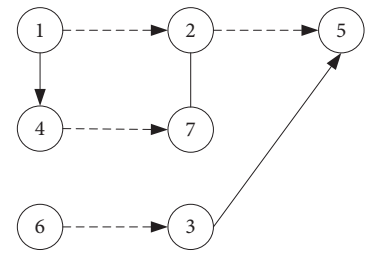

Figure 9: Topological diagram for the scheduling of block transportation tasks (n=7).

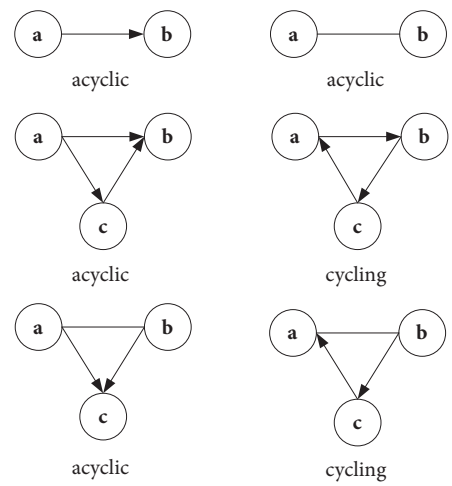

Figure 10: Rules to judge the feasibility of hybrid graphs.

(iv) For transporting list of transporters, use dashed directed line.

Usually, a directed acyclic graph means the plan is feasible. Since this problem has cooperating tasks, a solution can be expressed as a hybrid graph. So to determine the feasibility of a solution, new judgment rules are presented in Figure 10. The acyclic graph means the solution is feasible, while the cycling graph means it is unfeasible.

Based on the rules, we can find that the solution in Figure 9 is feasible. To explain the rules, an example with 10

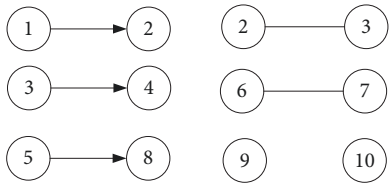

Figure 11: Topological diagram for the precedence and cooperating tasks (n=10).

Task sequence	1	6	4	9	2	5	3	10	7	8
Transporter Assignment	1	2	4	3	2	1	3	2	4	1

Figure 12: A chromosome of block transportation scheduling (n=10).

tasks and 4 transporters is given. The relationships among tasks are $1 \longrightarrow 2$, $3 \longrightarrow 4$, $5 \longrightarrow 8$, and $2 = 3, 6 = 7$, which can be described in Figure 11.

Suppose one chromosome is given in Figure 12.

Based on this chromosome, the solution can be converted into a topological diagram (see Figure 13).

We can find that there is a cycle in Figure 13. The cycle is shown in Figure 14. So it can be judged that the chromosome is unfeasible.

3.3.2. Chromosome Repair. If unfeasibility is caused by transporter overweight, reassign the task to a capable transporter. The process is relatively simple, so we do not detail it anymore.

If unfeasibility is caused by the conflict to topological relationships that means there is a topological cycle, we should change the transporting list to repair it.

Take Figure 13 as an example to detail the repair steps. As shown in Figure 15, within the topological cycle, there are two the dashed arcs $6 \longrightarrow 2$ and $4 \longrightarrow 7$. Choose one of them to break. In Figure 15(a), arc $4 \longrightarrow 7$ is broken. Then, Task 7 is temporarily unscheduled. So, assign Task 7 to another position and a new dashed arc is build, as shown in Figure 15(b). Then check the feasibility of new topological relationships again until there is no cycle, and that means the chromosome is feasible.

4. Computational Results

To verify the practicability and efficiency of the proposed algorithm for transporter scheduling problem considering precedence and cooperating constraints, different numerical simulations are tested and evaluated according to realistic situations.

The generating conditions are as follows.

(i) Planning horizon T is 480 minutes

(ii) Distances between two workshops/stock yards $d = Uniform(200, 2000)$ in meters

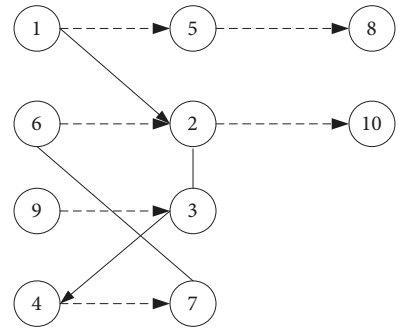

Figure 13: Topological diagram for the scheduling of block transportation tasks (n=10).

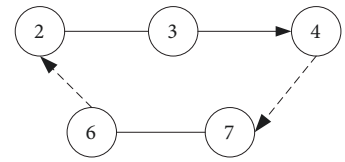

Figure 14: Topological cycle in the chromosome.

(iii) Release time of tasks $rt_i = Uniform(1, 360)$ in minutes

(iv) Due time of tasks $dt_i = Uniform(rt_i + 60, 480)$ in minutes

(v) Weight of blocks are randomly generated from $Uniform(100, 250)$ in tons

(vi) 20% pair of tasks have precedence or cooperating relationships

(vii) Loading and unloading time of blocks are randomly generated from $Uniform(10, 20)$ in minutes

(viii) Two flat transporters types are considered: 1# transporter: weight capacity is 150 tons; speed with load is 60m/min; speed without load is 100 m/min. 2# transporter: weight capacity is 200 tons; speed with load is 50m/min; speed without load is 80 m/min. The number of each type is m_1, m_2

This is a multiobjective optimization problem. In Formula (1), weights w_1, w_2 and w_3. are set depending on actual demand. Representative results are summarized under $\omega_1 = 0.7$ $\omega_2 = 0.2$ $\omega_3 = 0.1$.

In Figure 16, the test is made with $n = 30$ and $m = 10$. The dotted line denotes general genetic algorithm (GA), and the solid line denotes the proposed algorithm (GA-TS). After 500 iterations, the general GA did not converge, but GA-TS converged before 50 iterations. It is obvious that GA-TS converges much more speedily. Compared with the general genetic algorithm, the results show that the proposed algorithm has faster convergence speed.

In Figure 17, Tabu iterations are 0, 10, 20, 30, 40, and 50. After Tabu search, solution quality improves significantly. But when iterations exceed 10, the curve becomes stable. Thus, we select 10 iterations in the following tests.

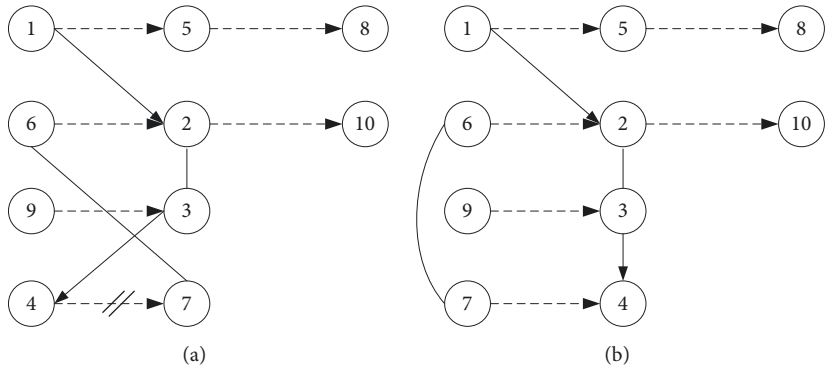

Figure 15: Chromosome repair.

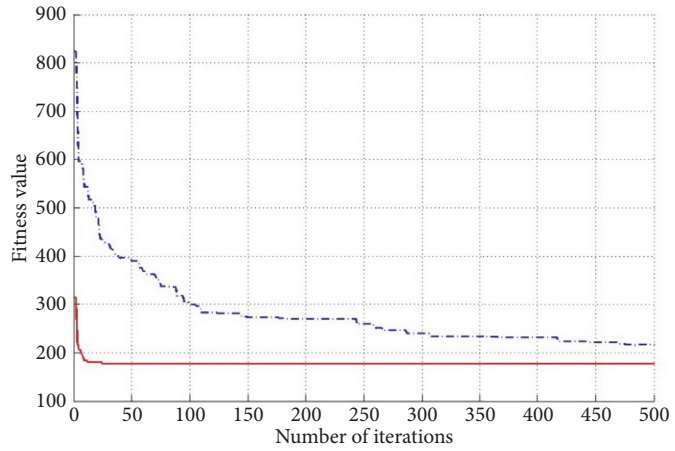

Figure 16: Algorithm convergence charts of GA and GA-TS.

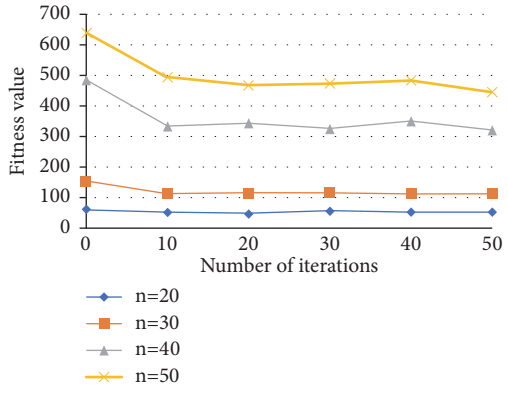

Figure 17: The objective values under different Tabu search iterations.

And with other pertaining to numerical tests, the algorithm parameters in this paper are determined as follows:

(i) Population size: 20

(ii) Number of generations: 100

(iii) Assignment crossover probability: 0.8

(iv) Assignment mutation probability: 0.1

(v) Sequencing crossover probability: 0.6

(vi) Sequencing mutation probability: 0.1

(vii) number of iterations: 20

(viii) number of iteration performed without improvement: 10

(ix) length of Tabu table: (m1+m2)/2

To evaluate the heuristic strategies of initial solutions, the problems are tested with three sets of block transportation tasks n as 20, 30, and 40, respectively, and the number of transporters is 4+4. Each test runs ten times to get an average result. "H1" is the heuristic strategy following the steps in Section 3.2.2 with Rule 1, while "H2" is the heuristic strategy following the steps in Section 3.2.2 with Rule 2. The result is presented in Table 3. We can find that H1 with Rule 1 gets less idle time of transporters, but H2 with Rule 2 gets better performance on time that related to production. Overall, we can choose a rule based on actual optimization requirements. In this paper, the scheduling scheme from these two heuristic rules is used as 60% of the initial population of the genetic algorithm to improve the efficiency of genetic algorithm.

For small-sized problems, we generate 5 instances shown in Table 4. The data of assembly blocks are randomly generated. The experiments are performed on a PC with Inter

Table 3: Comparison result of strategy rules (unit: min).

n	H1				H2			
	idle	delay	wait	f	idle	delay	wait	f
20	65.7	647.3	1054.8	511.2	222.9	209.5	417.3	405.9
30	99.9	1500.7	2089.7	591.4	313.1	324.9	569.3	426.6
40	168.1	3225.1	4262.8	741.4	466.3	801.1	1384.8	470.5

Table 4: Test results in small-sized instances.

n	m	CPLEX		GA-TS		GAP
		Opt (min)	Time (s)	f (min)	Time (s)	
6(1,1)	2	22.5	9.83	22.5	12.78	0
8(1,1)	2	34.4	944.26	34.4	28.43	0
8(1,1)	3	20.7	970.76	21.7	31.27	5%
8(2,1)	3	20.7	1350.81	21.4	50.13	3%
10(1,1)	2	56.3	3600	60.2	45.75	6%
10(2,2)	3	32.6	3600	33.5	62.21	3%

Table 5: Test results in large-sized instances.

n	m	Objective value (min)			
		H1	H2	GA	GA-TS
20	8	338.1	262.9	337.8	161.2
30	8	398.8	290.2	513.3	195.7
40	8	549.9	566.7	620.3	288.6
50	8	2005.2	1266.8	2052.5	866.2
20	10	118.8	206.1	232.2	94.9
30	10	231.7	368.8	459.8	170.1
40	10	633.2	345.7	430.1	226.6
50	10	805	494.4	1143.8	322.6
30	12	362	480	433.9	211.6
40	12	583.5	476.2	538.6	301.8
50	12	1104.4	460.8	834	180

Pentium 4, 3.01GHz, and 2GB RAM. We use ILOG CPLEX for finding the optimal solutions with the mathematical programming presented in Section 2.2. The maximum calculate time of CPLEX is set to 3600s. The codes of proposed algorithm are written in Matlab.

For large-sized problems, we generate 11 instances shown in Table 5. We compare the proposed solution (GA-TS) with the two heuristics strategies (H1 and H2) and a GA algorithm for each test problem.

Table 4 gives the test results of small-sized instances. It shows the optimal solution and computation time of CPLEX and the relative performance of the proposed algorithm for each test instance. In the table, n=6(1,1) means the problem has six tasks. When the transporters are more than two, two types are involved. The first "1" within the parentheses means there is one pair of tasks that has precedence constraint, and the other "1" means one of the six tasks is overweight. GAP is used to evaluate the performance of the proposed algorithm, where $GAP = ((f - opt)/f) \times 100\%$. As the problem size is increasing, the computing time of CPLEX increases dramatically. For some instances with larger problem size, the CPLEX cannot get the optimal solution within 3600s. Comparing the proposed algorithm with CPLEX, the proposed algorithm obtained optimal solutions in several problems, and when its solution is not optimal, the results are closer to the optimum one. Besides, the computing time did not increase rapidly. The number of precedence and cooperating constraints increases the difficulty; however, the proposed algorithm can get a satisfying solution while the computing time of the proposed algorithm does not increase a lot. The experimental results illustrate the efficiency of the proposed algorithm.

Table 5 gives the test results of large-sized instances. In these instances, the precedence and cooperating relationships are generated randomly. Within Table 5, H1 and H2 are two heuristic solutions gained by the experimental strategies which are obtained from experience. GA is designed as a general genetic algorithm flow. The results of GA are gained after 500 times iterations. We can see that results of GA are not stable even after 500 iterations; some result is worse than the solution gained by the experience strategies. While comparing the proposed algorithm (GA-TS) with the three algorithms, GA-TS obtained much better results. It is due to

using the strategies in initial solution and letting Tabu search process during each generation of GA. As the size of problem increases, the advantage of GA-TS becomes more and more obvious.

5. Conclusion

This paper researched a scheduling problem for block transportation with multitype transporters under precedence relationships and cooperating constraints. The problem includes three parts: assigning all of block transportation tasks to multiple transporters, sequencing the tasks for each transporter, and determining the start time to fulfill each task. The objective is to minimize logistics time, which includes empty travel time of transporters, waiting time, and delay time of block tasks. A MIP model was proposed for the problem. A hybrid topological graph was used to denote precedence and cooperating relationships of tasks, and a metaheuristic algorithm based on genetic algorithm (GA) and Tabu search (TS) was proposed. The performance of the algorithm was evaluated by comparing the algorithm to CPLEX in small-sized instances and two experience strategies and ordinary GA in large-sized instances. The results showed the efficiency and effectiveness of the proposed algorithm.

In this paper, it is assumed that the transportation tasks are executed smoothly on the road in shipyards. In reality, there must be other tasks executed on the same time which may cause the task to not be executed as plan and road traffic jar may happen. Therefore, it would be interesting to study the dynamic demands in the problem further.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The research is supported by National Natural Science Foundation of China under Projects (no. 71501125 and no. 71401098), Shanghai Pujiang Program (17PJC051), and MIIT Research Project on High Technology of Ships named "Intelligent Shipyard Top-Level Architecture and Application of Production Logistics".

References

[1] C. Park and J. Seo, "Comparing heuristic algorithms of the planar storage location assignment problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 1, pp. 171–185, 2010.

[2] Z. Zhang and J. Chen, "Solving the spatial scheduling problem: a two-stage approach," *International Journal of Production Research*, vol. 50, no. 10, pp. 2732–2743, 2012.

[3] C. Wang, Y.-S. Mao, B.-Q. Hu, Z.-J. Deng, and J. G. Shin, "Ship block transportation scheduling problem based on greedy algorithm," *Journal of Engineering Science and Technology Review*, vol. 9, no. 2, pp. 93–98, 2016.

[4] W. S. Lee, W. L. Lim, P. H. Koo, and C. M. Joo, "Transporter scheduling for block transportation in shipbuilding," *Journal of the Korea Management Engineers Society*, vol. 11, no. 1, pp. 169–179, 2006 (Portuguese).

[5] M.-I. Roh and J.-H. Cha, "A block transportation scheduling system considering a minimisation of travel distance without loading of and interference between multiple transporters," *International Journal of Production Research*, vol. 49, no. 11, pp. 3231–3250, 2011.

[6] B. S. Kim and C. M. Joo, "Ant colony optimisation with random selection for block transportation scheduling with heterogeneous transporters in a shipyard," *International Journal of Production Research*, vol. 50, no. 24, pp. 7229–7241, 2012.

[7] C. M. Joo and B. S. Kim, "Block transportation scheduling under delivery restriction in shipyard using meta-heuristic algorithms," *Expert Systems with Applications*, vol. 41, no. 6, pp. 2851–2858, 2014.

[8] H. Hu, Y. Wu, and T. Wang, "A Metaheuristic Method for the Task Assignment Problem in Continuous-Casting Production," *Discrete Dynamics in Nature and Society*, vol. 2018, Article ID 8073648, 12 pages, 2018.

[9] C. Park and J. Seo, "A GRASP approach to transporter scheduling and routing at a shipyard," *Computers & Industrial Engineering*, vol. 63, no. 2, pp. 390–399, 2012.

[10] Z.-Y. Zhang, J.-X. Xu, and F. Ji, "Shipbuilding yard scheduling approach based on genetic algorithm," *Shanghai Jiaotong Daxue Xuebao/Journal of Shanghai Jiaotong University*, vol. 47, no. 7, pp. 1036–1042, 2013.

[11] L. Zhen, E. P. Chew, and L. H. Lee, "An integrated model for berth template and yard template planning in transshipment hubs," *Transportation Science*, vol. 45, no. 4, pp. 483–504, 2011.

[12] G. Laporte and I. H. Osman, "Routing problems: a bibliography," *Annals of Operations Research*, vol. 61, no. 1, pp. 227–262, 1995.

[13] T. G. Crainic and G. Laporte, *Fleet Management And Logistics*, Kluwer Academic Publishers, Dordrecht, Mass, USA, 1998.

[14] I.-M. Chao, "A tabu search method for the truck and trailer routing problem," *Computers & Operations Research*, vol. 29, no. 1, pp. 33–51, 2002.

[15] R. Y. Zhang, W. Y. Yun, and I. Moon, "A reactive tabu search algorithm for the multi-depot container truck transportation problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 6, pp. 904–914, 2009.

[16] S. Sterzik and H. Kopfer, "A tabu search heuristic for the inland container transportation problem," *Computers & Operations Research*, vol. 40, no. 4, pp. 953–962, 2013.

[17] R. Tavakkoli-Moghaddam, F. Taheri, M. Bazzazi, M. Izadi, and F. Sassani, "Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints," *Computers & Operations Research*, vol. 36, no. 12, pp. 3224–3230, 2009.

[18] L. Zhen, D. Zhuge, and S. Zhu, "Production stage allocation problem in large corporations," *Omega* , vol. 73, pp. 60–78, 2017.

[19] Z.-H. Hu and C. Wei, "Synchronizing vehicles for multi-vehicle and one-cargo transportation," *Computers & Industrial Engineering*, vol. 119, pp. 36–49, 2018.

[20] A. Dohn, M. S. Rasmussen, and J. Larsen, "The vehicle routing problem with time windows and temporal dependencies," *Networks*, vol. 58, no. 4, pp. 273–289, 2011.

[21] M. Drexl, "Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints," *Transportation Science*, vol. 46, no. 3, pp. 297–316, 2012.

[22] M. A. Salazar-Aguilar, A. Langevin, and G. Laporte, "Synchronized arc routing for snow plowing operations," *Computers & Operations Research*, vol. 39, no. 7, pp. 1432–1440, 2012.

[23] M. A. Salazar-Aguilar, A. Langevin, and G. Laporte, "The synchronized arc and node routing problem: Application to road marking," *Computers & Operations Research*, vol. 40, no. 7, pp. 1708–1715, 2013.

[24] L.-M. Rousseau, M. Gendreau, and G. Pesant, "The synchronized dynamic vehicle dispatching problem," *INFOR. Information Systems and Operational Research*, vol. 51, no. 2, pp. 76–83, 2013.

[25] U. Derigs, M. Pullmann, and U. Vogel, "Truck and trailer routing—problems, heuristics and computational experience," *Computers & Operations Research*, vol. 40, no. 2, pp. 536–546, 2013.

[26] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, USA, 1989.

[27] H. Luo, G. Q. Huang, Y. Zhang, Q. Dai, and X. Chen, "Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 6, pp. 962–971, 2009.