Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2015, Article ID 618436, 12 pages
http://dx.doi.org/10.1155/2015/618436

Research Article

Genetic Algorithm and Its Performance Analysis for Scheduling a Single Crane

Xie Xie,[1] Yongyue Zheng,[2] and Yanping Li[1]

[1]*Key Laboratory of Manufacturing Industrial and Integrated Automation, Shenyang University, No. 21, WangHua South Street, Dadong District, Shenyang, Liaoning 110044, China*
[2]*Liaoning Institute of Standardization, No. 8, YongAn North Road, Heping District, Shenyang, Shenyang, Liaoning 110004, China*

Correspondence should be addressed to Xie Xie; xiexie8118@gmail.com

Received 21 August 2014; Revised 29 September 2014; Accepted 30 September 2014

Academic Editor: Peng Liu

This paper studies a single crane scheduling problem arising in the cold-rolling material warehouse in an iron and steel enterprise. A set of coils stored in two levels are needed to be picked up and transported to designated positions. If a required coil is at the upper level, it can be picked up right away and transported to its designated position (transportation operation). A required coil at the lower level cannot be picked up until all its blocking coils at the upper level are moved to other positions (shuffling operation). One overhead crane is used to perform all the transportation and shuffling operations. Our problem is to schedule the crane operations so as to all required coils retrieve to their designated positions in the shortest possible time (*makespan*). Since the problem is shown to be NP-hard, a genetic algorithm (GA) is proposed. We identify some analytical properties which enable us to develop an effective heuristic algorithm as initial solutions of the GA. We further analyze its performance from the worst-case point of view. To evaluate the average performances, a numerical test compared with some existing methods is carried out and its results show the good quality of the proposed algorithm.

1. Introduction

In iron and steel enterprises, cold rolling is the last main stage of production process that produces final steel products with customer required dimension, surface finish, mechanical properties, and so forth. The materials used in the cold-rolling process are steel coils from hot rolling. These coils are stored in the cold-rolling material warehouse and are retrieved when they are required according to the cold-rolling production plan. Usually one area in warehouse is served by a single bridge crane. Please see Figure 1 with three rows as an example. The crane moves along the track over the area while its pickup device (hoist) can move along the crane bridge. In this way, the hoist of the crane can reach any position in the area. From here onwards, we will refer to the hoist position of the crane simply as the crane position. In each area, coils are stored in rows. According to technological requirements, the coils may be stored in at most two levels. Any coil stored at the upper level must be supported by two coils at the lower level. When a coil is required, it can be retrieved directly if it is stored at the upper level or stored at the lower level and not being blocked by coils at the upper level. If it is stored at the lower level and there is one or two coils at the upper level blocking it, the blocking coils have to be shuffled to other empty positions in the area before the required coil can be retrieved. The retrieval of all required coils and the shuffling of all blocking coils are performed by the overhead crane.

Throughout the paper, picking-up of a required coil and moving it to its designated place will be called a crane transportation operation. Picking-up of a blocking coil and moving it to an empty position will be called a crane shuffling operation. In either a transportation operation or a shuffling operation, the coil concerned needs to be lifted up from its current position, moved to another position, and then dropped off. The whole process is considered as a loaded move for the crane. After the crane drops off a coil, the empty crane can move to another coil to perform the next loaded move. We explore the single crane scheduling problem

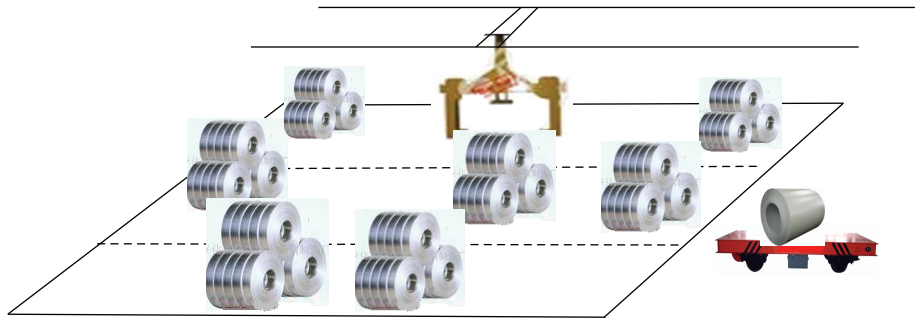

Figure 1: Coil storage in an area of steel coil warehouse.

to determine the sequence for transportation and shuffling moves and to decide the positions that the blocking coils should be shuffled to. The objective is to minimize the *makespan*, that is, the time by which the retrieval of all required coils is completed.

Effective solution of this scheduling problem helps to achieve better use of the expensive overhead crane and provide better logistics support for the steel production process. Similar problems also exist in other warehouses, such as material and product warehouses for hot-galvanizing and acid-rolling operations and stack yards at container terminals, but have not received much research attention. One apparent reason for this phenomenon is the difficulty associated with the simultaneous consideration of interrelated transportation and shuffling operations.

Up to date, research on related crane scheduling problem has been mainly on different contexts, such as quay crane scheduling problem (QCSP) and yard crane scheduling problem (YCSP) in container terminal to determine a sort of crane operational planning for improving the efficiency of the terminal, and hoist scheduling problem (HSP) in electroplating line by optimizing hoist movements to minimize the production cycle. There is a rich literature on these various crane scheduling problems. Interested readers may refer to Steenken et al. [1], Stahlbock and Voß [2], and Bierwirth and Meisel [3] for surveys in container terminal and Lee et al. [4], Hall et al. [5], and Crama et al. [6] for surveys in HSP system. However, the amount of researches on crane scheduling in steel coil warehouses is relatively small. Zäpfel and Wasner [7] investigated a single crane scheduling problem in a distribution center of steel coils to store incoming coils and retrieve coils required by customers. The problem is viewed as a coil shop scheduling problem and formulated as a nonlinear integer programming model which is hard to solve. Local search based heuristics is proposed and tested through computation. Rei et al. [8] considered a single crane scheduling problem to store and retrieve steel items with known arrival and retrieval dates to minimize the number of crane movements. The items are stacked one on top of another in a similar way to container stacking. Simulation based heuristics is proposed to solve the problem. Most related to our problem studied is researched by Tang et al. [9], but they proposed a heuristic algorithm with worst-case performance analyzed without using genetic algorithm for solving the problem. Xie et al. [10] further studied the multicrane scheduling problem in steel coil warehouse and still they did not propose genetic algorithm.

Moreover, the problem studied here concerns material handling activities in warehouses and may be viewed as one of the warehousing operation management problems. There is a rich literature on various warehousing operation management problems. Interested readers may refer to van den Berg and Zijm [11] for a discussion of warehousing systems and a classification of warehouse management problems and de Koster et al. [12] for a review of studies on warehouse design and control issues. Research on these, order picking involves only retrieving products from the storage area in response to specific customer orders. Ratliff and Rosenthal [13] solved the problem of routing order pickers, considering horizontal travel time and as a special case of the traveling salesman problem (TSP), in a linear computation time in terms of the number of aisles and the number of pick locations. Ascheuer et al. [14] modeled the picker route problem, considering both horizontal and vertical travel times, as an online asymmetric TSP. Kim et al. [15] solved an order picking problem using an intelligent agent-based model where there are separated storage areas each having a dedicated picker, and goods are stored at multiple locations and the picking location of the goods can be selected dynamically. Petersen and Aase [16] examined the effect of picking, storage, and routing using a simulation model based on the operations of a distribution center. Rei et al. [8] presented a situation where a stacking crane delivers all items for a client order in a steel stacking warehouse to minimize the number of movements (total delivery and reshuffling or shifting operations). Petersen and Aase [16] examined the effect of picking, storage, and routing using a simulation model based on the operations of a distribution center. They took entities (goods or parts) and resources (storage area and order pickers) as intelligent agents which cooperate with each other so as to fulfill individual and whole system goals. However, these order picking problems do not involve shuffling decisions which is a main concern in our problem.

Research involving shuffling activities is usually on container handling problems. Kim and Hong [17] mentioned a branch and bound procedure and presented a heuristic for determining the storage positions for shuffled containers during retrieving a given sequence of export containers to minimize the number of shuffles needed. Wan et al. [18]

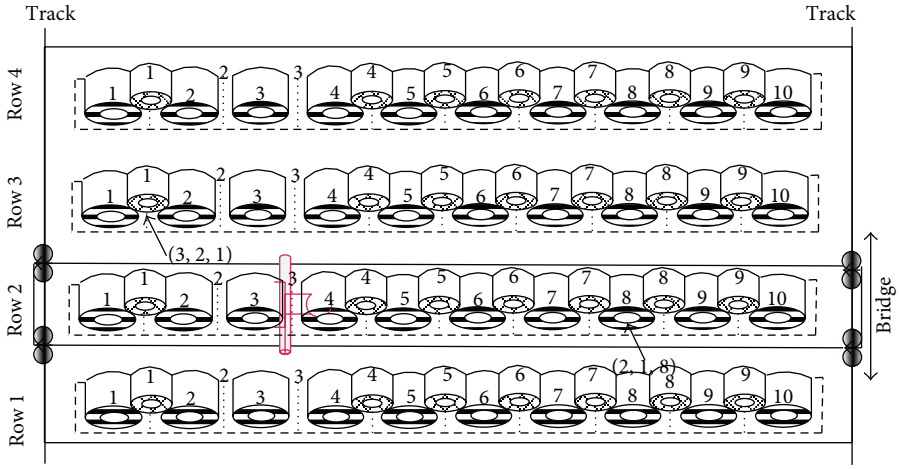

Figure 2: Top view of an area in steel coil warehouse.

developed a linear integer programming model for the container retrieval and shuffling problems and presented heuristics for handling container storage and retrieval dynamically. Y. Lee and Y. J. Lee [19] presented a three-phase heuristic for retrieving containers in a given sequence to minimize the weighted sum of the number of container movements and the crane's working time. Given the storage configuration and a sequence of containers to be retrieved, Lee and Hsu [20] proposed an integer programming model for premarshalling containers with minimum number of marshalling movements so that the required containers can be retrieved later without shuffling. Lee and Chao [21] solved larger instances of the problem by neighborhood search heuristics. Avriel et al. [22] studied a ship stowage planning problem to minimize the number of shuffles needed in unloading at destination terminals. At a more aggregated level, Zhang et al. [23] assign storage spaces in the yard for incoming containers to balance the workloads of cranes in different blocks in each period. The problem under our consideration has its own features which makes the existing models are not able to be applied to our problem directly. The stacking of coils is different from that of containers or other flat items. These problems are easier than ours as coils are placed on top of each other in stack and therefore one required coil may be blocked by only one coil.

The paper is organized in the following way. The problem is described and formulated in Section 2. In Section 3, we develop an MILP to describe our problem in detail. Section 4 is devoted to develop a genetic algorithm with worst-case performance analysis. Computational results are reported in Section 5 comparing the solutions with those given by the MILP model and the genetic algorithm. Some conclusions are finally given in Section 6.

2. Problem Statement and Notation

In the section, we unify the problem notation which is proposed by Tang et al. [9] and the problem studied here is described as follows. Consider a warehouse area with N positions served by one bridge crane. The area is arranged into R rows. Each row can store at most two levels of coils. The lower and upper levels are called levels 1 and 2, respectively. There are P_l storage positions at level l in each row. Obviously $P_2 = P_1 - 1$, due to the special stacking structure. Let P be the set of all positions in the area. Each position $k \in P$ can be identified uniquely by its row-level-position coordinates (r_k, l_k, p_k). In the rest of the paper, we may refer to a position using its number or its coordinates, whichever being more convenient. Figure 2 shows a top view of an area where large and small patterned circles represent coils at lower and upper positions, respectively, and an empty circle indicates an empty position. The number in a circle is the number for that position at that level in that row.

There are N_0 coils stored in the warehouse area. The coils are numbered in such a way that the set of coils required to be retrieved and moved to their designated places is $\Omega_t = \{1, 2, \ldots, n\}$. For convenience of expression, the required coils are also called target coils. Let $\Omega_b = \{n+1, n+2, \ldots, n+b\}$ be the set of all blocking coils and let Ω_{bi} be the set of coils blocking target coil i ($i \in \Omega_t$). Then set $\Omega_o = \{n+b+1, n+b+2, \ldots, N_0\}$ includes other coils in the area. Each coil i has a known initial storage position o_i.

To perform a loaded move, the time needed for the crane to lift up or drop off a coil is μ (this can be easily extended to situations with different lift-up and drop-off times). For loaded moves, the crane traveling speeds along the track and along the bridge are v_1 and v_2, respectively. Similarly, the empty crane speeds in the two directions are λ_1 ($\lambda_1 \geq v_1$) and λ_2 ($\lambda_2 \geq v_2$), respectively. As all the positions are known in advance, and the distance between positions k and q in the two directions, d^1_{kq} and d^2_{kq} ($k, q \in P$), is also known. Then the time for a loaded move between positions k and q can be calculated by $t_{kq} = \max\{d^1_{kq}/v_1, d^2_{kq}/v_2\} + 2\mu$ and that for an empty move between positions k and q can be calculated by $e_{kq} = \max\{d^1_{kq}/\lambda_1, d^2_{kq}/\lambda_2\}$. Here, the traveling time satisfies the triangular inequality. Considering that a position can also be represented by its coordinates, each move time can be denoted in two ways: $t_{kq} \equiv t_{r_k l_k p_k, r_q l_q p_q}$, $e_{kq} \equiv e_{r_k l_k p_k, r_q l_q p_q}$.

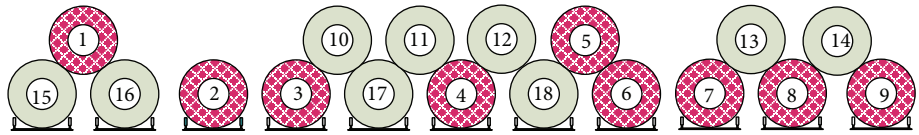

Figure 3: Side view of a row in steel coil warehouse.

Our problem is to determine the retrieval sequence of the target coils and to decide the positions that the blocking coils should be shuffled to with the objective of minimizing the time by which the retrieval of all target coils is completed. Although a blocking coil may be shuffled to any empty position, to avoid further shuffling, we do not allow any blocking coil to be shuffled to a position that blocks another target coil.

According to their positions, the target coils in Ω_t may be categorized into four classes as described below. A side view of a row shown in Figure 3 can help understand the classification. The grid circles in the figure represent the target coils.

Class 1. These are the target coils located on level 2 (e.g., coils 1 and 5 in Figure 3) or located on level 1 but not blocked by other coils (e.g., coil 2 in Figure 3). Let Ω_1 denote the subset of coils in this class.

Class 2. These are the target coils located on level 1 and only blocked by other target coils (e.g., coil 6 in Figure 3). Let Ω_2 denote the subset of coils in this class.

Class 3. These are the target coils located on the lower level, each of which is blocked by one nontarget coil (e.g., coil 3 blocked by coil 10, coil 7 blocked by coil 13, and coil 9 blocked by coil 14 in Figure 3). Let Ω_3 denote the subset of coils in this class and let Ω_3' denote the set of the nontarget coils blocking the coils in Ω_3.

Class 4. These are the target coils located on the lower level, each of which is blocked by two nontarget coils (e.g., coil 4 blocked by coils 11 and 12 and coil 8 blocked by coils 13 and 14). Let Ω_4 denote the subset of coils in this class and let Ω_4' denote the set of the coils blocking the coils in Ω_4.

According to the above definitions, we have $\Omega_t = \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4$. Note that some blocking coils may block two target coils simultaneously, for example, coil 13 blocks both coil 7 and coil 8. In addition, a blocking coil such as 13 and 14 may belong to both Ω_3' and Ω_4'. Hence, it is possible that $\Omega_3' \cap \Omega_4' \neq \emptyset$. The number of the blocking coils is thus $b = |\Omega_3'| + |\Omega_4'| - |\Omega_3' \cap \Omega_4'|$. It is clear from the above classification that coils in Ω_1 can be retrieved directly and those in Ω_2 can also be retrieved directly after retrieving the ones in Ω_1 blocking them. In order to retrieve a coil in Ω_3 or Ω_4, the coils blocking it need to be shuffled first to other positions.

For easy reference, we summarize all the notations mentioned above in the Notations Section. Some further notations will be defined later when needed.

3. Mathematical Formulation

To present our problem more clearly, in this section, we continue to use a mixed integer linear programming (MILP) model proposed by Tang et al. [9].

3.1. Problem Parameters

3.1.1. Input Data

$(r_{o_i}, l_{o_i}, p_{o_i})$ is the initial position of coil i ($i \in \Omega_t \cup \Omega_b \cup \Omega_o$). With this information we can also work out the indicator X^0_{irlp}, for every i, r, l, p, which equals 1, if the initial position of coil i is (r, l, p) or otherwise equals 0. That is, $X^0_{ir_{o_i}l_{o_i}p_{o_i}} = 1$, all other $X^0_{irlp} = 0$. Note that each X^0_{irlp} is a known parameter, rather than a variable, representing the fact whether coil i is initially at position (r, l, p).

$(r_{d_i}, l_{d_i}, p_{d_i})$ is the designated place of target coil i ($i \in \Omega_t$).

(r_0, l_0, p_0) is the initial position of the crane.

$t_{rlp,r'l'p'}$ is the time needed for loaded move from position (r, l, p) to position (r', l', p').

$e_{rlp,r'l'p'}$ is the time needed for empty move from position (r, l, p) to position (r', l', p').

M is a large positive number.

3.1.2. Decision Variables. One has the following:

$$X_{irlp} = \begin{cases} 1, & \text{if coil } i \text{ is shuffled to position } (r, l, p) \\ 0, & \text{otherwise,} \end{cases}$$

$$\text{for } i \in \Omega_b$$

$$Z_{ij} = \begin{cases} 1, & \text{if the loaded move for coil} \\ & i \text{ is the } j\text{th loaded move} \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

$$\text{for } i \in \Omega_t \cup \Omega_b, \quad j = 1, 2, \ldots, n + b$$

S_j = the starting time of the jth loaded move,

$$\text{for } j = 1, 2, \ldots, n + b$$

C_j = the completion time of the jth loaded move,

$$\text{for } j = 1, 2, \ldots, n + b.$$

3.2. Mathematical Model. The problem can be formulated as the following model:

$$\text{Minimize} \quad C_{n+b} \tag{2}$$

$$\text{Subject to} \quad \sum_{i\in\Omega_t\cup\Omega_b} Z_{ij} = 1, \quad j = 1,2,\ldots,n+b \tag{3}$$

$$\sum_{j=1}^{n+b} Z_{ij} = 1, \quad i \in \Omega_t \cup \Omega_b \tag{4}$$

$$\sum_{j=1}^{n+b} jZ_{kj} \le \sum_{j=1}^{n+b} jZ_{ij}, \quad k \in \Omega_{bi},\ i \in \Omega_t \tag{5}$$

$$\sum_{j=1}^{n+b} jZ_{ij} \le \sum_{j=1}^{n+b} jZ_{kj} + M\left(1 - X_{kr_{o_i}l_{o_i}p_{o_i}}\right), \quad k \in \Omega_b, \quad i \in \Omega_t \cup \Omega_b, \quad k \ne i \tag{6}$$

$$\sum_{r=1}^{R}\sum_{l=1}^{2}\sum_{p=1}^{P_l} X_{irlp} = 1, \quad i \in \Omega_b \tag{7}$$

$$\sum_{i\in\Omega_b} X_{irlp} \le 1 - \sum_{i\in\Omega_o} X^0_{irlp}, \quad r = 1,2,\ldots,R, \quad p = 1,2,\ldots,P_l, \quad l = 1,2, \tag{8}$$

$$2\sum_{i\in\Omega_b} X_{ir2p} \le \sum_{i\in\Omega_o} X^0_{ir1p} + \sum_{i\in\Omega_b} X_{ir1p} + \sum_{i\in\Omega_o} X^0_{ir1(p+1)} + \sum_{i\in\Omega_b} X_{ir1(p+1)}, \quad r = 1,2,\ldots,R, \quad p = 1,2,\ldots,P_2 \tag{9}$$

$$\sum_{j=1}^{n+b} jZ_{ij} \le \sum_{j=1}^{n+b} jZ_{kj} + M\left(2 - X_{ir1p} - X_{ir1(p+1)} - X_{kr2p}\right), \quad k \in \Omega_b, \quad r = 1,2,\ldots,R, \quad p = 1,2,\ldots,P_2, \quad i \in \Omega_b, \tag{10}$$

$$C_{j-1} + \sum_{r=1}^{R}\sum_{l=1}^{2}\sum_{p=1}^{P_l} e_{rlp,r_{o_i}l_{o_i}p_{o_i}} X_{krlp} \le S_j + M\left(2 - Z_{ij} - Z_{k(j-1)}\right), \quad j = 2,\ldots,n+b, \quad k \in \Omega_b, \quad i \in \Omega_t \cup \Omega_b, \quad k \ne i, \tag{11}$$

$$C_{j-1} + e_{r_{d_k}l_{d_k}p_{d_k},r_{o_i}l_{o_i}p_{o_i}} \le S_j + M\left(2 - Z_{ij} - Z_{k(j-1)}\right), \quad j = 2,\ldots,n+b, \quad k \in \Omega_t, \quad i \in \Omega_t \cup \Omega_b, \quad k \ne i, \tag{12}$$

$$\sum_{i\in\Omega_t\cup\Omega_b} e_{r_0l_0p_0,r_{o_i}l_{o_i}p_{o_i}} Z_{i1} \le S_1 \tag{13}$$

$$S_j + \sum_{r=1}^{R}\sum_{l=1}^{2}\sum_{p=1}^{P_l} t_{r_{o_i}l_{o_i}p_{o_i},rlp} X_{irlp} \le C_j + M\left(1 - Z_{ij}\right), \quad j = 1,2,\ldots,n+b, \quad i \in \Omega_b, \tag{14}$$

$$S_j + t_{r_{o_i}l_{o_i}p_{o_i},r_{d_i}l_{d_i}p_{d_i}} \le C_j + M\left(1 - Z_{ij}\right), \quad j = 1,2,\ldots,n+b, \quad i \in \Omega_t, \tag{15}$$

$$S_j, C_j \ge 0, \quad j = 1,2,\ldots,n+b, \tag{16}$$

$$X_{irlp} \in \{0,1\}, \quad i \in \Omega_b, \quad r = 1,2,\ldots,R, \quad p = 1,2,\ldots,P_l, \quad l = 1,2, \tag{17}$$

$$Z_{ij} \in \{0,1\}, \quad i \in \Omega_t \cup \Omega_b, \quad j = 1,2,\ldots,n+b. \tag{18}$$

The objective (2) in the model is to minimize the *makespan*. Constraints (3) guarantee that the crane performs a loaded move for one coil at a time. Constraints (4) guarantee that the retrieval of each target coil and the shuffling of each blocking coil are performed only once. Constraints (5) ensure that, before retrieving a target coil, its blocking coils must be shuffled. Constraints (6) ensure that if a blocking coil k is shuffled to a position (r, l, p) which is initially occupied by another coil i, coil i must be moved away before shuffling k. Constraints (7) guarantee that a blocking coil must be shuffled to only one position. Constraints (8) ensure that a blocking coil cannot be shuffled to positions occupied by other coils that are not moved during the whole operation. Constraints (9) ensure that a blocking coil can be shuffled to a level-2 position only if both positions beneath it are occupied by coils always being there or being shuffled to there. Note that the value of $\sum_{i\in\Omega_o} X^0_{ir1p} + \sum_{i\in\Omega_b} X_{ir1p}$ in (9) can be at most 1 as position $(r, 1, p)$ can only hold at most one coil. Similarly, the value of $\sum_{i\in\Omega_o} X^0_{ir1(p+1)} + \sum_{i\in\Omega_b} X_{ir1(p+1)}$ can also be at most 1. Constraints (10) ensure that the shuffling of a blocking coil to a level-1 position must be done before another blocking coil is shuffled to a position above it. Note that at most one of X_{ir1p} and $X_{ir1(p+1)}$ in (10) can be 1, since coil i can only be shuffled to one position. Constraints (11) and (12) ensure that there is enough time for the empty move of the

crane between the completion time of the $(j-1)$th loaded move and the start time of the jth loaded move. The two sets of constraints are for the cases where the $(j-1)$th loaded move is for a blocking coil and for a target coil, respectively. Constraint (13) ensures that the first loaded move can start only after the empty crane reaches the starting position of the loaded move. Constraints (14) and (15) state that between the start and completion times of a loaded move, there must be enough time for the crane to perform the move. The two sets of constraints are for the loaded moves of blocking coils and target coils, respectively. Constraints (16)–(18) are nonnegativity and binary constraints on the variables.

Note that the problem definition assumes that each blocking coil will be shuffled to a position that is not blocking other target coils; that is, a blocking coil needs to be shuffled only once during the process of retrieving all the target coils. This is in line with what is observed in practice. The above model is developed under this assumption. Extension of the model to include the case where each blocking coil is allowed to be shuffled more than once will need more variables to represent the positions of the blocking coils after each target coil is retrieved. Interested readers are referred to Wan et al. [18] for the method of modeling such a shuffling problem in a container yard environment.

4. The Proposed Genetic Algorithm and Performance Analysis

Since our problem is NP-hard in the strong sense (see Tang et al. [9]) and the fast growth of the solution time with the size of the instance remains the major drawback of MILP, a genetic algorithm presented offers, in contrast, a reasonable comprise in terms of computational burden and solution quality. A genetic algorithm is a well-known metaheuristic approach inspired by the natural evolution of the living organisms that works on a population of the solutions simultaneously. It is a randomized optimization technique that draws it inspiration from the biological science. Specially, it uses the idea that genetics determines the evolution of any species in the nature world. Integer strings are used to encode an optimization problem and these strings are subject to combinatorial operations called reproduction, crossover, and mutation, which improve these strings and cause them to "evolve" to an optimal or nearly optimal solution.

4.1. The Proposed Genetic Algorithm. Genetic algorithms (GAs) have been used extensively in combinatorial optimization problems, such as sequencing and scheduling problems. GA is a well-known metaheuristic approach inspired by the natural evolution of the living organisms that works on a population of the solutions simultaneously. The exploration process is performed both by a genetic operator, namely, crossover, and another genetic operator, namely, mutation. The trade-off between these two processes is controlled by the parent selection and offspring acceptance strategies. A single iteration is called a generation. The individuals of the new generation are obtained from the individuals of the previous one by applying reproduction and mutation procedures.

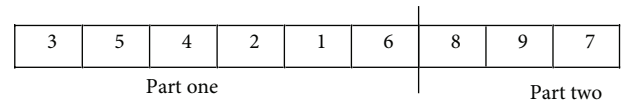

Figure 4: Chromosome representation.

4.2. Genetic Operations Design and Chromosome Representation. As we know, GA starts with a population of individuals and each solution is called a chromosome consisting of genes that represent the decision variables. The chromosome representation used in this paper demonstrates each coil in the schedule as a gene in the chromosome. The total number of decision variables plays a significant role in the computational time. Problem with more variables will require longer computation time. In the mathematical formulation presented in Section 3, the total number of decision variables is $2(nN + n^2)$. In the paper we define a chromosome to have two parts. The first part represents the sequence of the shuffled coils and the required coils, and the second part represents the shuffled positions for each shuffled block coils correspondingly. A GA chromosome encoding is depicted as in Figure 4. This chromosome represents our problem with 3 required coils and 3 shuffled coils. As indicated in the first part of the chromosome, there are six loaded moves (numbered 1–6). Coils 1, 2, and 3 are the required coils and coils 4, 5, and 6 are the shuffled ones. The second part of the chromosome indicates that empty positions 8, 9, 7 are for the shuffled coils 4, 5 and 6 correspondingly. This chromosome encoding approach can reduce the number of decision variables.

4.3. Initial Solution Procedure. It is a well-known fact that the structure of the initial population plays an essential role in determining the efficiency of GAs (Goldberg [24]). However, most GA implementations in the literature employs randomly generated populations for initiation. To generate the random solutions for the initial population, we consider the following steps.

Step 1. Transport all coils belonging to Ω_1 and Ω_2, by choosing coil with the minimal $(e_{r_0 l_0 p_0, r_{o_i} l_{o_i} p_{o_i}} - e_{r_{o_i} l_{o_i} p_{o_i} r_{d_i} l_{d_i} p_{d_i}})$ as the first coil and the left $|\Omega_1| + |\Omega_2| - 1$ coils are in arbitrary sequence. Go to Step 2.

Step 2. If there are enough positions for coils in $\Omega_3' \cup \Omega_4'$ to be shuffled without making coils in Ω_2 and Ω_3 to be blocked, shuffle the blocking coils one by one according to the following principle: choose one coil with the nearest distance from crane current position and shuffle it to the nearest position. Else, go to Step 3.

Once all the blocking coils in $\Omega_3' \cup \Omega_4'$ are shuffled completely, then go to Step 1.

Step 3. Compare the two distances created by (3.1) and (3.2) for choosing the nearest one to perform. If two distances are equal, then choose the coils in $\Omega_1 \cup \Omega_2$ prior to coils in $\Omega_3' \cup \Omega_4'$. If there is no shuffled position for any one coil in $\Omega_3' \cup \Omega_4'$, go to Step 1 for transporting current coils in $\Omega_1 \cup \Omega_2$. Once all

coils in Ω are transported to the designated place, stop. One has the following:

(3.1) the total distances from crane current position to one coil in $\Omega_3' \cup \Omega_4'$ and then to its nearest shuffled position;

(3.2) the total distances from crane current position to one coil in Ω and then to the designated place.

4.4. Arithmetic Crossover. It is important to maintain the feasibility of the newly generated offspring for the problem at hand. Thus, we use the arithmetic crossover (AC) operator to explore the solution space and maintain the feasibility of the newly generated offspring simultaneously. The AC produces a new offspring as complimentary linear combination of the parents as follows.

Offspring = $\lambda \times$ Parent 1 + $(1 - \lambda) \times$ Parent 2, where λ is a randomly generated number within an interval from 0 to 1. Thus, each gene value (i.e., allele) in the newly generated offspring is obtained. To increase the effect of the parent with better fitness, we set the parent with better fitness to Parent 1. The AC guarantees that the generated offspring will remain feasible if its parents are feasible. Since values in genes are integer values, the rounded of gene value (offspring) must be considered as a true value.

4.5. Parent Selection Strategy. Parent selection is important in regulating the bias in the reproduction process. The parent selection strategy means how to choose the chromosomes in the current population that will create offspring for the next generation. Generally, it is better that the best solutions in the current generation have more chance for being selected as parents for creating offspring. The most common method for the selection mechanism is the "roulette wheel" sampling, in which each chromosome is assigned a slice of a circular roulette wheel and the size of the slice is proportional to the chromosome's fitness.

4.6. Offspring Acceptance Strategy. We use a semigreedy strategy to accept the offspring generated by the genetic operators. In this strategy, an offspring is accepted for the new generation if its fitness is less than the average fitness of its parent(s). This strategy reduces the computational time of the algorithm and leads to a monotonous convergence toward the optimum solution neighborhood.

4.7. Stoppage Rules. We use two criteria as stoppage rules: (1) maximum number of elapsed generation that is a common criterion and (2) standard deviation of the fitness value of chromosomes in the current generation. This parameter implies the degree of diversity or similarity in the current population in terms of the objective function value. If this criterion reduces below an arbitrary constant, say ε, then the algorithm is stopped.

4.8. Performance Analysis. Let the schedule derived from the genetic algorithm be σ^H. A lower bound LB of the problem can be derived from the following two bounds: for the convenience of expression, we will refer to the distance between crane initial position and coil J_i as d_i^0, the distance between coil J_i and its designated place as d_i, and the distance between coils J_i and J_j as d_{ij}. Let $\lambda = \min\{\lambda_1, \lambda_2\}$ and $\nu = \min\{\nu_1, \nu_2\}$. Consider

$$\mathrm{LB}_1 = \min_{J_i \in \Omega_t} \frac{\left\{\left(d_i^0 - d_i\right)\right\}}{\lambda} + (\nu + \lambda) \sum_{i=1}^{n} \frac{d_i}{\lambda\nu} + n4\mu,$$

$$\mathrm{LB}_2 = \frac{\left[n \min_{J_i \in \Omega_t} \{d_i\} + \left(\left|\Omega_3'\right| + \left|\Omega_4'\right|\right) \min_{J_i, J_j \in \Omega_t \cup \Omega_b} \left\{d_{ij}\right\}\right]}{\nu} + 4\mu\left(n + \left|\Omega_3'\right| + \left|\Omega_4'\right|\right), \tag{19}$$

where $\mathrm{LB} = \max\{\mathrm{LB}_1, \mathrm{LB}_2\}$.

LB_1 is the optimal objective value if all the required coils are in Ω_1 and Ω_2. The rationale for LB_2 is that we consider the minimal sum of transporting and shuffling times.

As a single crane must transport all the required coils to designated place and shuffle all the blocking coils, the *makespan* depends on crane performing sequence. The aim of the algorithm is to minimize crane transporting and shuffling distance as soon as possible. Since the algorithm includes all the possible cases, we analyze the worst-case performance in the following three cases.

Case 1 ($\Omega = \Omega_1 \cup \Omega_2$). In this case, all the required coils are in Ω_1 and Ω_2, and we can get the optimal scheduling σ^* by applying Step 1 of the algorithm and the objective value which is equal to optimal objective value $C_{\max}(\sigma^*)$. Hence, we derive

$$C_{\max}\left(\sigma^H\right) = C_{\max}\left(\sigma^*\right) = \mathrm{LB}_1 = \min_{J_i \in \Omega_t} \frac{\left\{\left(d_i^0 - d_i\right)\right\}}{\lambda} + (\nu + \lambda) \sum_{i=1}^{n} \frac{d_i}{\lambda\nu} + n4\mu. \tag{20}$$

Case 2 ($N^0 - n \geq \left|\Omega_3'\right| + \left|\Omega_4'\right|$). In this case, there are enough positions for all the blocking coils in $\Omega_3' \cup \Omega_4'$ shuffled without making coils in $\Omega_2 \cup \Omega_3$ blocked. According to Step 2, before transporting all the required coils, we first shuffle all the blocking coils. The objective value of the scheduling σ^H cannot be larger than the following expression where the sum of the first two items is the longest time for shuffling all the coils and the sum of the last three items is the shortest time for transporting all the required coils

$$C_{\max}\left(\sigma^H\right) \leq \max_{J_i \in \Omega_t} \frac{\left\{d_i^0\right\}}{\lambda} + \left(4\mu + \max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\lambda} + \max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\nu}\right)$$

$$\times \left(\left|\Omega_3'\right| + \left|\Omega_4'\right|\right) + \min_{J_i \in \Omega_t} \frac{\left\{\left(d_i^0 - d_i\right)\right\}}{\lambda} + (v + \lambda) \sum_{i=1}^{n} \frac{d_i}{\lambda v} + n4\mu. \tag{21}$$

Case 3 ($N^0 - n < |\Omega_3'| + |\Omega_4'|$). In this case, as there are not enough positions for all the blocking coils in $\Omega_3' \cup \Omega_4'$ shuffled without making coils in $\Omega_2 \cup \Omega_3$ blocked, one blocking coil may be shuffled more than once. For the sake of decreasing shuffling time as much as possible, the crane may shuffle and transport coils alternately. Thereby, the *makespan* $C_{\max}(\sigma^H)$ for completing all required coils always satisfies the following expression where the first item is the maximal time for crane empty move from its initial position, the second and third items are the maximal time for shuffling coils and transporting all required coils, respectively, and the forth item is the sum of loading and unloading time:

$$C_{\max}\left(\sigma^H\right) \le \max_{J_i \in \Omega_t} \frac{\left\{d_i^0\right\}}{\lambda} + \left(\max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\lambda} + \max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{v} \right) \times \left(\left|\Omega_3'\right| + \left|\Omega_4'\right|\right) + \left(\max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\lambda} + \max_{J_i \in \Omega_t} \frac{\left\{d_i\right\}}{v} \right) n + 4\mu\left(\left|\Omega_3'\right| + \left|\Omega_4'\right| + n\right). \tag{22}$$

Together with Case 2 above and $C_{\max}(\sigma^*) \ge \mathrm{LB} \ge \mathrm{LB}_1$, we get one worst-case performance bound combined with $|\Omega_3| \le n$ and $|\Omega_4| \le 2n$,

$$\begin{aligned}
&\frac{\left[C_{\max}\left(\sigma^H\right) - C_{\max}\left(\sigma^*\right)\right]}{C_{\max}\left(\sigma^*\right)} \\
&\le \frac{\left[C_{\max}\left(\sigma^H\right) - C_{\max}\left(\sigma^*\right)\right]}{\mathrm{LB}_2} \\
&\le \left(\max_{J_i \in \Omega_t} \frac{\left\{d_i^0\right\}}{\lambda} + \left(4\mu + \max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\lambda} + \max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{v} \right) \times \left(\left|\Omega_3'\right| + \left|\Omega_4'\right|\right) \right) \\
&\quad \times \left(\frac{\left[n \min_{J_i \in \Omega_t}\left\{d_i\right\} + \left(\left|\overline{\Omega}_3\right| + \left|\overline{\Omega}_4\right|\right) \min_{J_i, J_j \in \Omega_t \cup \Omega_b}\left\{d_{ij}\right\}\right]}{v} + 4\mu\left(n + \left|\Omega_3'\right| + \left|\Omega_4'\right|\right) \right)^{-1} \\
&\le \frac{D/v + \left(4\mu + (2D)/v\right) 3n}{nd/v + 4\mu n} \le \frac{D + 12\mu nv + 6Dn}{nd + 4\mu nv} \\
&\le \frac{7Dn + 12\mu nv}{nd + 4\mu nv} \le \frac{3\left(D + 4\mu v\right)}{d + 4\mu v} + \frac{4D}{d} \le \frac{7D}{d}.
\end{aligned} \tag{a}$$

Together with Case 3 above and $C_{\max}(\sigma^*) \ge \mathrm{LB} \ge \mathrm{LB}_2$, we get the other worst-case performance bound combined with $|\Omega_3| \le N$ and $|\Omega_4| \le 2N$,

$$\begin{aligned}
&\frac{\left[C_{\max}\left(\sigma^H\right) - C_{\max}\left(\sigma^*\right)\right]}{C_{\max}\left(\sigma^*\right)} \\
&\le \frac{\left[C_{\max}\left(\sigma^H\right) - C_{\max}\left(\sigma^*\right)\right]}{\mathrm{LB}_2} \\
&\le \left(\left\{ \max_{J_i \in \Omega_t} \frac{\left\{d_i^0\right\}}{\lambda} + \left[\max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\lambda} + \frac{\left(\max_{J_i \in \Omega_t}\left\{d_i\right\} - \min_{J_i \in \Omega_t}\left\{d_i\right\}\right)}{v} \right] n + \left[\max_{J_i, J_j \in \Omega_t \cup \Omega_b} \frac{\left\{d_{ij}\right\}}{\lambda} + \frac{\left(\max_{J_i, J_j \in \Omega_t \cup \Omega_b}\left\{d_{ij}\right\} - \min_{J_i, J_j \in \Omega_t \cup \Omega_b}\left\{d_{ij}\right\}\right)}{v} \right] \right. \\
&\qquad \left. \times \left(\left|\Omega_3'\right| + \left|\Omega_4'\right|\right) \right\} \right) \\
&\quad \times \left(\frac{\left[n \min_{J_i \in \Omega_t}\left\{d_i\right\} + \left(\left|\Omega_3'\right| + \left|\Omega_4'\right|\right) \min_{J_i, J_j \in \Omega_t \cup \Omega_b}\left\{d_{ij}\right\}\right]}{v} + 4\mu\left(n + \left|\Omega_3'\right| + \left|\Omega_4'\right|\right) \right)^{-1} \\
&\le \frac{(4n+1) D/v + 3n(D - d)/v}{nd/v + 4\mu n} \le \frac{(7Dn + D)/v}{nd/v + 4\mu n} \\
&\le \frac{8Dn}{nd + 4\mu nv} \le \frac{8D}{d}.
\end{aligned} \tag{b}$$

Hence, max{(a), (b)} is the worst-case performance bound.

Theorem 1. *For any schedule σ^H of our problem generated by the genetic algorithm, it provides the worst-case performance guarantee, where $D = \max_{J_i, J_j \in \Omega_t \cup \Omega_b}\{d_i^0, d_{ij}, d_i\}$, $d = \min_{J_i, J_j \in \Omega_t \cup \Omega_b}\{d_i^0, d_{ij}, d_i\}$,*

$$\frac{\left[C_{\max}\left(\sigma^H\right) - C_{\max}\left(\sigma^*\right)\right]}{C_{\max}\left(\sigma^*\right)} \le 8\frac{D}{d}. \tag{23}$$

Table 1: The performance for smaller scale instances.

	Problem				ARD (%)				
	R	P_1	$U(N_0)$	n	MILP	Sequential approach	Heuristic algorithm	Modified heuristic	GA
	2	3	0.5	2	2.93	2.93	5.33	5.33	2.93
			0.7		3.52	3.52	3.52	3.52	3.52
			0.9		5.50	5.50	6.34	6.17	5.54
	3	3	0.5	3	4.74	4.90	5.97	5.97	4.74
$v_1 = 1$			0.7		5.31	5.31	6.05	6.05	5.31
$v_2 = 2$			0.9		4.99	5.22	6.58	6.47	5.16
$\lambda_1 = 2$	3	4	0.5	5	5.06	5.48	7.35	7.35	5.18
$\lambda_2 = 4$			0.7		4.32	4.54	7.31	7.16	5.22
$\mu = 2.5$			0.9		8.05	9.12	11.20	11.68	8.34
	3	5	0.5	6	4.70	5.06	9.73	9.60	4.70
			0.7		$5.53^{\{2\}}$	6.13	7.88	8.74	6.01
			0.9		$9.25^{\{10\}}$	$12.03^{\{4\}}$	14.69	13.04	11.74
	2	3	0.5	2	1.80	1.80	3.30	3.30	1.80
			0.7		2.12	2.12	2.12	2.12	2.12
			0.9		3.29	3.29	3.79	3.69	3.32
	3	3	0.5	3	3.00	3.00	3.65	3.65	3.00
$v_1 = 1$			0.7		3.25	3.25	3.70	3.70	3.25
$v_2 = 2$			0.9		3.13	3.13	3.95	3.89	3.36
$\lambda_1 = 2$	3	4	0.5	5	3.14	3.39	4.55	4.55	3.14
$\lambda_2 = 4$			0.7		2.64	2.83	4.48	4.39	2.64
$\mu = 5$			0.9		$4.82^{\{1\}}$	5.50	6.71	6.99	5.29
	3	5	0.5	6	2.92	3.14	6.06	5.97	2.92
			0.7		$3.42^{\{2\}}$	3.93	4.88	5.39	3.89
			0.9		$6.42^{\{10\}}$	$7.32^{\{3\}}$	9.51	7.96	7.26

Note that this ratio only bounds the performance for the potential worst case. This bound correlates with the distance parameters.

5. Computational Experiments

To test the performances of the proposed GA, the computational experiments used the methods which were provided by Tang et al. [9]. These methods are MILP, sequential approach, heuristic algorithm, and modified heuristic. For each problem setting, 10 problem instances were solved. Different set of problem instances were generated, each of which is characterized by the number of rows R in the storage area, the number of storage positions P_1 at the lower level in each row ($P_2 = P_1 - 1$ positions at the upper level), the space utilization U, and the number of target coils n. The possible positions for designated places are at one end of the rows (near position 1 of each row). For each instance, the number of designated places is generated randomly from 1 to R, and their positions are randomly selected from the potential positions. The designated place for each target coil is then randomly chosen from these selected positions. The distance of two adjacent positions in the same row (As shown in Figure 2, the two adjacent positions in the same row are defined by the respective positions of two contact coils located at different level in the same row) is 1 and the distance between two adjacent rows is 2. The crane traveling speeds for loaded and empty moves in the two directions are $v_1 = 1, v_2 = 2, \lambda_1 = 2, \lambda_2 = 4$. The time for lifting up and dropping off a coil is $\mu = 2.5$. To test the effect of μ that is not related to distance, we doubled its value and solved the problem instances again. The corresponding results are shown in the lower part of Tables 1 and 2.

A time limit of 10 hours was set to solve the MILP of each instance. In case an optimal solution was not obtained within the time limit, the best feasible solution was recorded. $\{\cdot\}$ the number in the brackets indicates the number of instances in the group for which the MILP models were not solved to optimality within their time limit. These algorithms were implemented using Visual C++ and ran on a PC with a 2.33 GHz Pentium CPU and 1 G RAM. The MILP models of the instances were solved using CPLEX version 11.0 on a PC with the same configuration. We use the lower bound LB, as a basis to compare the solution quality. For a solution with a *makespan* of $C_{\max}$, the quality is measured by its relative deviation from the LB, $(C_{\max} - \text{LB})/\text{LB} * 100\%$.

From Tables 1 and 2 we can make the following observations.

For most of the smaller instances, GA can obtain an optimal solution. Although the computation time of GA is

Table 2: The average computational time for smaller scale instances.

	Problem				Avg. CPU time			
	R	P_1	$U(N_0)$	n	MILP	Sequential approach	Heuristic algorithms	GA
$v_1 = 1$ $v_2 = 2$ $\lambda_1 = 2$ $\lambda_2 = 4$ $\mu = 2.5$	2	3	0.5	2	0.002	0.016	0.000	0.000
			0.7		0.025	0.014	0.000	0.000
			0.9		0.025	0.021	0.000	0.000
	3	3	0.5	3	0.036	0.021	0.000	0.000
			0.7		0.043	0.024	0.000	0.000
			0.9		0.169	0.048	0.000	0.001
	3	4	0.5	5	5.508	0.953	0.000	0.000
			0.7		23.426	1.580	0.000	0.000
			0.9		5684.544	27.935	0.002	0.002
	3	5	0.5	6	83.014	6.170	0.000	0.000
			0.7		8217.837	10.973	0.002	0.003
			0.9		36000.000	1090.988	0.003	0.005
$v_1 = 1$ $v_2 = 2$ $\lambda_1 = 2$ $\lambda_2 = 4$ $\mu = 5$	2	3	0.5	2	0.023	0.010	0.000	0.000
			0.7		0.031	0.008	0.000	0.000
			0.9		0.047	0.020	0.000	0.000
	3	3	0.5	3	0.037	0.020	0.000	0.000
			0.7		0.051	0.002	0.000	0.000
			0.9		0.127	0.045	0.000	0.001
	3	4	0.5	5	4.704	0.858	0.000	0.000
			0.7		40.274	1.543	0.000	0.000
			0.9		7377.501	23.781	0.002	0.002
	3	5	0.5	6	76.869	6.003	0.000	0.000
			0.7		9213.951	10.712	0.002	0.003
			0.9		36000.000	1262.419	0.003	0.005

a little longer than that of the heuristic algorithms, the ARD of GA is smaller than that of the MILP. As problem size or the space utilization increases, the GA solution time taken for problems with 90% space utilization is longer than those for lower (50%) and (70%) space utilizations. This may be because for very high space utilization there are fewer empty slots for the shuffled coils and so is the number of iterations, and for low space utilization, there are fewer blocking coils, which also helps reducing search time.

From the results in Table 3, we can see that the results of the GA continued the trend shown for the smaller problems. As the problem size increases, the ARD to LB increases steadily. Therefore, looking at the trend of ARD in Tables 1 and 3, we can see that the GA solution may be not far from optimal. The ARD for the GA is very similar to that of the heuristic algorithm and the modified heuristic. The computation time of the GA increases with the problem size. However, it only takes a fraction of a second to solve a problem with over 100 target coils. Even a problem with 800 coils can be solved within half a minute. Since in practice it is rare for the problem size to reach $n = 100$, the proposed GA is capable of generating good solutions within very short time. Therefore they are suitable for practical use.

6. Conclusions

To conclude, the contributions of this paper are as follows. Unlike the previous related literature, the purpose of our study is to shed light on this practical problem through some theoretical analyses. The problem is formulated as an MILP model. As the strongly NP-hard nature, we develop a genetic algorithm for producing good and quick approximate solutions for practical applications. The performance of this algorithm has been demonstrated efficiently and effectively which is related with some problem parameters. It provides a solution for improving the efficiency of the cranes and output of the production in warehouse of the iron and steel enterprise. These contributions can be used for enterprise to achieve a more efficient productivity in warehouse management and utilize expensive facilities.

Note that our study is fit for solving single crane scheduling problem for picking up required coils in warehouse of the steel enterprises. Further research may consider applying other well-known population-based metaheuristics such as a memetic algorithm to solve our problem involving other similar shuffle and reshuffling processes in steel plate warehouse and in container terminals while considering implementation details.

Notations

N: The number of all positions in the warehouse area being considered
P: The set of all positions in the area
R: The number of rows in the area
l: Index of levels; $l = 1$ and $l = 2$ represent the lower and upper levels, respectively
(r_k, l_k, p_k): The row-level-position coordinates for $k \in P$
N_0: The number of all stored coils in the area

Table 3: Computing results of the heuristic algorithms for larger scale instances.

Problem				ARD (%)			Avg. CPU time	
R	P_1	$U(N_0)$	n	Heuristic algorithm	Modified heuristic	GA	Heuristic algorithms	GA
3	6	0.5	10	11.09	11.09	11.00	0.000	0.000
		0.7		12.69	13.34	11.90	0.003	0.005
		0.9		11.87	12.83	12.07	0.002	0.003
5	6	0.5	16	12.92	12.96	12.66	0.000	0.000
		0.7		10.62	11.35	10.37	0.003	0.005
		0.9		13.11	13.62	13.09	0.000	0.002
10	6	0.5	33	18.54	18.68	17.89	0.005	0.005
		0.7		16.25	16.50	16.14	0.004	0.005
		0.9		15.66	17.23	15.30	0.007	0.010
15	6	0.5	50	23.47	23.47	23.00	0.006	0.006
		0.7		21.54	21.72	21.21	0.015	0.018
		0.9		20.41	21.05	19.99	0.015	0.020
20	6	0.5	65	26.57	26.68	26.18	0.003	0.003
		0.7		24.78	24.98	24.54	0.015	0.018
		0.9		23.03	24.22	22.91	0.021	0.030
20	11	0.5	130	26.17	26.18	26.02	0.060	0.071
		0.7		27.48	27.98	27.19	0.109	0.132
		0.9		22.49	23.84	22.00	0.153	0.169
50	26	0.5	800	34.60	34.54	34.52	6.572	7.121
		0.7		33.00	32.92	32.73	19.647	23.409
		0.9		31.33	31.75	30.90	32.838	35.117

Ω_t: The set of target coils (required to be retrieved and moved to their designated places), $\Omega_t = \{1, 2, \ldots, n\}$

Ω_b: The set of blocking coils, $\Omega_b = \{n+1, n+2, \ldots, n+b\}$

Ω_{bi}: The set of coils blocking target coil i ($i \in \Omega_t$)

Ω_o: The set of other coils, $\Omega_o = \{n+b+1, n+b+2, \ldots, N_0\}$

Ω_1: The subset of target coils located on level 2 or located on level 1 but not blocked by other coils

Ω_2: The subset of target coils located on level 1 and only blocked by other target coils

Ω_3: The subset of target coils located on level 1, each of which is blocked by one nontarget coil

Ω_3': The set of the nontarget coils blocking the coils in Ω_3

Ω_4: The subset of target coils located on level 1, each of which is blocked by two nontarget coils

Ω_4': The set of the coils blocking the coils in Ω_4

μ: The time needed for the crane to lift up or drop off a coil

v_1 and v_2: The crane traveling speeds along the track and along the bridge, respectively

λ_1 and λ_2: The empty crane traveling speeds along the track ($\lambda_1 \geq v_1$) and along the bridge ($\lambda_2 \geq v_2$), respectively

d_{kq}^1 and d_{kq}^2: The distance between positions k and q along the track and along the bridge, respectively

t_{kq} and e_{kq}: The time for a loaded move and for an empty move, respectively, between positions k and q, $t_{kq} = \max\{d_{kq}^1/v_1, d_{kq}^2/v_2\} + 2\mu$, $e_{kq} = \max\{d_{kq}^1/\lambda_1, d_{kq}^2/\lambda_2\}$. They can also be denoted in another way: $t_{kq} \equiv t_{r_k l_k p_k, r_q l_q p_q}$, $e_{kq} \equiv e_{r_k l_k p_k, r_q l_q p_q}$.

Conflict of Interests

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

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant no. 71201104).

References

[1] D. Steenken, S. Voß, and R. Stahlbock, "Container terminal operation and operations research—a classification and literature review," *OR Spectrum*, vol. 26, no. 1, pp. 3–49, 2004.

[2] R. Stahlbock and S. Voß, "Operations research at container terminals: a literature update," *OR Spectrum*, vol. 30, no. 1, pp. 1–52, 2008.

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

[4] C.-Y. Lee, L. Lei, and M. Pinedo, "Current trends in deterministic scheduling," *Annals of Operations Research*, vol. 70, pp. 1–41, 1997.

[5] N. G. Hall, H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: complexity and steady state analysis," *European Journal of Operational Research*, vol. 109, no. 1, pp. 43–65, 1998.

[6] Y. Crama, V. Kats, J. van de Klundert, and E. Levner, "Cyclic scheduling in robotic flowshops," *Annals of Operations Research*, vol. 96, pp. 97–124, 2000.

[7] G. Zäpfel and M. Wasner, "Warehouse sequencing in the steel supply chain as a generalized job shop model," *International Journal of Production Economics*, vol. 104, no. 2, pp. 482–501, 2006.

[8] R. J. Rei, M. Kubo, and J. P. Pedroso, "Simulation-based optimization for steel stacking," *Communications in Computer and Information Science*, vol. 14, pp. 254–263, 2008.

[9] L. X. Tang, X. Xie, and J. Y. Liu, "Crane scheduling in a warehouse storing steel coils," *IIE Transactions*, vol. 46, no. 3, pp. 267–282, 2014.

[10] X. Xie, Y. Y. Zheng, and Y. P. Li, "Multi-crane scheduling in steel coil warehouse," *Expert Systems with Applications*, vol. 41, no. 6, pp. 2874–2885, 2014.

[11] J. P. van den Berg and W. H. M. Zijm, "Models for warehouse management: classification and examples," *International Journal of Production Economics*, vol. 59, no. 1, pp. 519–528, 1999.

[12] R. de Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: a literature review," *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, 2007.

[13] H. D. Ratliff and A. S. Rosenthal, "Order picking in a rectangular warehouse: a solvable case of the traveling salesman problem," *Operations Research*, vol. 31, no. 3, pp. 507–521, 1983.

[14] N. Ascheuer, M. Grotschel, and A. A. A. Abdel-Hamid, "Order picking in an automatic warehouse: solving online asymmetric TSPs," *Mathematical Methods of Operations Research*, vol. 49, no. 3, pp. 501–515, 1999.

[15] B. I. Kim, R. J. Graves, S. S. Heragu, and A. S. Onge, "Intelligent agent modeling of an industrial warehousing problem," *IIE Transactions*, vol. 34, no. 7, pp. 601–612, 2002.

[16] C. G. Petersen and G. Aase, "A comparison of picking, storage, and routing policies in manual order picking," *International Journal of Production Economics*, vol. 92, no. 1, pp. 11–19, 2004.

[17] K. H. Kim and G.-P. Hong, "A heuristic rule for relocating blocks," *Computers & Operations Research*, vol. 33, no. 4, pp. 940–954, 2006.

[18] Y.-W. Wan, J. Liu, and P.-C. Tsai, "The assignment of storage locations to containers for a container stack," *Naval Research Logistics*, vol. 56, no. 8, pp. 699–713, 2009.

[19] Y. Lee and Y. J. Lee, "A heuristic for retrieving containers from a yard," *Computers and Operations Research*, vol. 37, no. 6, pp. 1139–1147, 2010.

[20] Y. Lee and N.-Y. Hsu, "An optimization model for the container pre-marshalling problem," *Computers & Operations Research*, vol. 34, no. 11, pp. 3295–3313, 2007.

[21] Y. Lee and S.-L. Chao, "A neighborhood search heuristic for pre-marshalling export containers," *European Journal of Operational Research*, vol. 196, no. 2, pp. 468–475, 2009.

[22] M. Avriel, M. Penn, and N. Shpirer, "Container ship stowage problem: complexity and connection to the coloring of circle graphs," *Discrete Applied Mathematics*, vol. 103, no. 1–3, pp. 271–279, 2000.

[23] C. Zhang, J. Liu, Y.-W. Wan, K. G. Murty, and R. J. Linn, "Storage space allocation in container terminals," *Transportation Research Part B: Methodological*, vol. 37, no. 10, pp. 883–903, 2003.

[24] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wiley, Boston, Mass, USA, 1989.