1. Introduction

Freight empty cars allocation aims to distribute the available empty cars from origins to destinations in the railway networks so that the demands can be satisfied with the minimized shipment costs. Since the railway freight transportation plays an important role in modern society, reasonable allocation of the empty cars with systematic optimization is actually a crucial issue for the railway companies. In recent decades, a lot of researchers have investigated freight empty cars allocation problem and presented a variety of models and algorithms. In literature, Misra [1] firstly proposed this problem in 1972 and set up a simple freight empty train allocation model in a time period with the minimum cost consumption. Transportation algorithm and simplex method were also designed to search for the optimal solution of the proposed model. Grain and Sinay [2] figured out the freight empty car allocation problem by a linear programming model, in which the objective was to arrange car flow assignment to satisfy the maximum proceeds. They adopted transportation algorithm and simplex method for the solutions of this model. Haghani [3] converted the freight empty car allocation problem into an inventory problem in cargo operation stations in order to allocate the freight cars in these stations. Kikuchi [4] structured the problem as a transshipment problem under the concept that a fleet of single-purpose freight cars were pooled at many loading points and emptied at unloading terminals in order to reduce empty car miles and time. Liu [5] firstly applied the computer technology to solve the freight empty car allocation problem, which provided technical guidance for the future study. Moreover, Xiong et al. [6] propounded an effective genetic algorithm to solve the large-scale network empty cars allocation problem and designed an adapted matrix coding mode, crossover operator, and mutation operator to improve the searching performance of the genetic algorithm.

Besides the above mentioned researches, recent studies always formulated this problem based on the standard transportation problem and traffic flow problem. As for the transportation problem based methods, Liu [7] investigated both the balanced transportation model and the unbalanced transportation model for the problem, which were solved by minimum element methods. X. Zhang and Q. Zhang [8] considered the problem based on the balanced transportation problem with the influence of the constraint conditions. Liang et al. [9] presented the concepts of alternative vehicles in unbalanced transportation model and constructed an optimization model with maximized profit. Lei et al. [10] introduced a stochastic programming model with probability chance-constraints and designed an efficient genetic algorithm to seek an approximate optimal solution. Narisetty et al. [11] proposed an optimization model for the empty car
assignment problem based on the customer demand, which was implemented at Union Pacific Railroad to assign empty freight cars. The proposed approaches can further reduce transportation costs and improve delivery rate efficiency and customer satisfaction.

More recently, some researchers developed network flow models and optimization methods which could be regarded as references for the freight empty car allocation problem. For instance, Wang et al. [12] considered freight empty car allocation problem on the basis of quantity distribution and network matching flow and then designed ant colony algorithm to solve the proposed model. Laporte et al. [13] applied a game theoretic framework to research the railway transit network design problem. Dorfman and Medanic [14] developed a discrete event model for the feedback travel advance strategy, which can quickly handle perturbations in the schedule. Erkut and Gzara [15] applied the network flow strategy to hazardous material transportation and proposed a heuristic solution method to search a stable solution. In a large-scale railway network, Guo et al. [16] proposed the methods to merge the traffic flow in the main support stations and reduce nodes in railway networks, which can improve the efficiency of operations.

To understand the contributions of this research clearly, the detailed features will be summarized in Table 1 in comparison with some related works.

It is easy to see in the literature that the majority of researches investigate this problem in certain environments, in which all the parameters are assumed to be static. In practice, since railway traffic is a complicated system, the demands of empty freight cars are always dynamic in most circumstances. To clearly state this characteristic, in this study, we particularly treat the demand at each destination as a stage-based demand, in which one stage corresponds to a prespecified time period. Then, we formulate the problem as a mathematical optimization model with multistage demands. Also, a genetic algorithm is particularly designed for the problem to efficiently generate the approximate optimal solution of the model.

The rest of this paper is organized as follows. In Section 2, an optimization model for freight empty cars allocation in railway networks and some preliminaries are presented. In Section 3, a genetic algorithm is designed to find an optimal solution for freight empty cars allocation problem. Some instances are given to show the effectiveness of the model and algorithm in Section 4. Conclusions are finally made in Section 5.

2. Formulation of the Problem

2.1. Problem Statement. Allocating empty cars is an important operation for railway companies to guarantee the normal operations of transportation activity. In traditional operation modes, empty cars are in general required to be transferred to the destination according to the prespecified demand that is always treated as a fixed quantity. However, due to the uncertainty of decision environments, the demand of freight empty cars at the destination is always changeable in the real-world applications (namely, the needs vary in different time periods). Thus, the traditional model will potentially cause a large number of empty backlogs or supply shortage, leading to the transportation task being unable to be accomplished effectively. Aiming to provide a framework for practical decisions, in this research we are particularly interested in the formulation of the model with dynamic demands and effective algorithms for the freight empty cars allocation problem.

In dynamic case, the time horizon will be divided into a variety of time periods, denoted by stages. Then, some important parameters in the problem will be reconsidered as the stage-based quantities, including the demand, cost, capacity, and so forth. For different stages, all of these parameters can be different according to the practical requirements, while they are assumed to be fixed quantities within each given stage. To formulate the dynamic empty cars allocation problem, we need to specify the following three types of constraints. (1) The first constraint concerns the traffic flow volume from supply stations, which is determined by the predicted demands in organization plans and train schedules. In particular, the traffic flow volume is required to be consistent with the supply capacity of each supply station. (2) The second constraint refers to the transportation capacities of stations and railway links. The total traffic flow on these stations and links cannot exceed the corresponding capacities. (3) The last constraint is associated with the demand of traffic flow volume. That is, all the dynamic demands need to be satisfied in each stage.

To clearly state the considered problem, we shall give an illustration in Figure 1, where stations 1 and 2, respectively, represent the supply station and demand station in the
railway network. We here consider the following scenarios: in day 1, the demand of the destination is 20 empty cars, while the demand is 30 empty cars in day 2. Without considering stage-based demands, a total of 50 empty cars will be delivered to station 2 in day 1, leading to the waste of resources and the storage cost for unoccupied cars. On the other hand, if the demand is divided into two stages of resources and the storage cost for unoccupied cars. On the one hand, if the demand is divided into two stages associated with different days, that is, \( D_1 = 20 \) and \( D_2 = 30 \), then the extra cost can be avoided if we allocate the empty cars according to the stage-based demands.

Hereinafter, some assumptions will be given in order to formulate the mathematical model.

(A1) In order to characterize the considered railway network, an abstract graph, denoted by \( G(N, A) \), will be extracted from the physical network (see an illustration in Figure 2), in which \( N \) denotes the set of nodes representing the railway stations, and \( A \) is the set of directed links between adjacent nodes. Corresponding to the different stages, stage-based transportation cost will be treated as the weight of each link. Directed links indicate the transportation direction between two nodes.

(A2) The demand, storage cost, and unit transportation cost in the considered time horizon are all dynamic, while these parameters in each stage are treated as constants.

(A3) In the railway network, the total supply capacity in original stations is supposed to be greater than the total demand in destinations to guarantee the transportation activities.

(A4) The demand in each stage should be satisfied. Unused empty cars can be left for the next stage with an extra storage cost.

Table 2: Parameters used in formulation.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>The set of links;</td>
</tr>
<tr>
<td>( N )</td>
<td>The set of nodes;</td>
</tr>
<tr>
<td>( i, j )</td>
<td>The index of nodes;</td>
</tr>
<tr>
<td>( (i, j) )</td>
<td>The index of link from node ( i ) to node ( j );</td>
</tr>
<tr>
<td>(</td>
<td>A</td>
</tr>
<tr>
<td>(</td>
<td>N</td>
</tr>
<tr>
<td>( k )</td>
<td>The different stages, ( k = 1, 2, \ldots, K );</td>
</tr>
<tr>
<td>( c_{ij}^k )</td>
<td>The unit transportation cost on link ((i, j)) in stage</td>
</tr>
<tr>
<td>( o_s )</td>
<td>Origin station which supplies empty cars, ( s = 1, 2, \ldots, S );</td>
</tr>
<tr>
<td>( R_s )</td>
<td>The supply amount of empty cars in station ( o_s );</td>
</tr>
<tr>
<td>( d_t )</td>
<td>Destination station which requires empty trains, ( t = 1, 2, \ldots, T );</td>
</tr>
<tr>
<td>( D_t )</td>
<td>The total demand of empty cars in station ( d_t );</td>
</tr>
<tr>
<td>( Q_{ij} )</td>
<td>The link capacity on link ((i, j));</td>
</tr>
<tr>
<td>( G_i )</td>
<td>The turnover capacity at station ( i );</td>
</tr>
<tr>
<td>( e_s )</td>
<td>The unit storage cost for stage ( k );</td>
</tr>
<tr>
<td>( U^k )</td>
<td>The demand of empty cars at station ( d_t ) for stage ( k ).</td>
</tr>
</tbody>
</table>

and three links, is given in Figure 3. To show the dynamics in the network, the time horizon is firstly divided into four stages. Then the right side of Figure 3 depicts the space-time network for the empty car allocating process, where each node represents the state of a physical node in each stage (for more detailed description for space-time networks, interested readers may refer to Yang and Zhou [18], Yang et al. [19]). Suppose that nodes \( a \) and \( c \), respectively, denote the supply station and demand station. Then, two paths can be employed for the operations, that is, \( a \rightarrow b \rightarrow c \) and \( a \rightarrow c \), as shown in the arrow links. In the first stage, there will be some empty cars required to be transferred to station \( c \) from station \( a \). If the transferred cars just satisfy the demand at station \( c \), no storage cost occurs. Otherwise, if the number of transferred cars is greater than the demand at station \( c \), then the unused empty cars should be delayed at this station, leading to an extra storage cost. The dotted arrow links in Figure 3 represent the waiting arcs for the unused empty cars at station \( c \). Likewise, in the following stages, it is required that the summation of transferred cars and stored cars of the previous stages should also satisfy the stage-based demands. Then, in the process of allocating empty cars, the total cost can be separated into two parts: the transportation cost and storage cost. The aim is to generate the optimal allocation strategies in railway networks.

2.2. Mathematical Model. In this section, we shall formulate the freight empty cars allocation problem with dynamic demands as a stage-based network flow problem below.

2.2.1. Parameters and Notations. To formulate the mathematical model for this problem, some relevant parameters are firstly listed in Table 2.
2.2.2. Decision Variables. In this problem, the purpose is to generate an optimal transportation plan over the entire time horizon. As the multistage strategy will be considered, the decision variable can be treated as the amount of empty car flow corresponding to each link and stage. Then, we have the decision variable as

\[ x^k_{ij} \] denotes the number of empty cars delivered on link \((i, j)\) for stage \(k\), satisfying \(x^k_{ij} \in \mathbb{Z}^+ \cup \{0\}\), where \(\mathbb{Z}^+\) denotes the set of positive integers.

With this variable, the freight empty car allocation problem can be essentially treated as a traffic flow assignment process with the following constraint conditions.

2.2.3. System Constraints. To guarantee the feasibility of the allocation plans, we need to consider some system constraints in the process of allocating empty cars. Specifically, there are six types of system constraints in this model, which are presented as follows.

(i) The Supply Capacity Constraint. The available empty cars in supply station \(o_s\) should be more than or equal to the cars that are dispatched from this station \(o_s\). We then have the following constraint:

\[
\sum_{(o_s, j) \in A} \sum_{k=1}^{K} x^k_{o,j} \leq R_s, \quad s = 1, 2, \ldots, S. \tag{1}
\]

(ii) The Demand Constraint. When the flow is delivered at the station \(d_t\), it is required that the delivered flow volume \(\sum_{(i, d_t) \in A} \sum_{k=1}^{K} x^k_{i,d_t}\) should meet the total demand \(D_t\), which can be represented as

\[
\sum_{(i,d_t) \in A} \sum_{k=1}^{K} x^k_{i,d_t} \geq D_t, \quad t = 1, 2, \ldots, T. \tag{2}
\]

(iii) The Flow Balance Constraint. To keep the balance of the total flow, it is required that the amount of the inflow cars should be equal to the outflow cars at each station. We then have

\[
\sum_{(i,j) \in A} x^k_{ij} = \sum_{(j,i) \in A} x^k_{ji}, \quad i \in N/\{ o_s, d_t, s = 1, 2, \ldots, S, t = 1, 2, \ldots, T \}, \quad k = 1, 2, \ldots, K. \tag{3}
\]

(iv) The Link Capacity Constraint. In general, the transportation capacity on each link is not finite. To show this characteristic in the formulation, we particularly consider the link capacity constraint. That is, for each link \((i, j)\), the flow volume cannot exceed the transportation capacity \(Q_{ij}\), which can be represented as

\[
\sum_{k=1}^{K} x^k_{ij} \leq Q_{ij}, \quad (i, j) \in A. \tag{4}
\]

(v) The Turnover Capacity Constraint. According to the real-world applications, a turnover capacity constraint should also be considered at each station. Then, at a transfer station \(t\), the volume of traffic flow cannot exceed the turnover capacity \(G_t\). The constraint can be represented as follows:

\[
\sum_{k=1}^{K} \sum_{i \in N} x^k_{i, t} \leq G_t, \quad i \in N/\{ o_s, d_t, s = 1, 2, \ldots, S, t = 1, 2, \ldots, T \}. \tag{5}
\]

(vi) The Stage-Based Demand Constraint. As the total demand is separated as stage-based demand, the flow volume delivered to each destination and its stored empty cars should satisfy the demand \(U^1_t\) at station \(d_t\) for each stage; namely, if \(k = 1\), there will be one constraint:

\[
\sum_{(i,j) \in A} \sum_{k=1}^{K} x^k_{i,j} \geq U^1_t, \quad t = 1, 2, \ldots, T. \tag{6}
\]

And if \(k = 2\), the constraints should be

\[
\sum_{(i,j) \in A} \sum_{k=1}^{K} x^k_{i,j} \geq U^1_t, \quad t = 1, 2, \ldots, T;
\]

\[
\sum_{(i,j) \in A} \left( x^1_{i,j} + x^2_{i,j} \right) \geq U^1_t + U^2_t, \quad t = 1, 2, \ldots, T. \tag{7}
\]

Considering all the stages, we formulate these constraints as follows:

\[
\sum_{(i,j) \in A} \sum_{k=1}^{K} x^k_{i,j} \geq \sum_{k=1}^{n} U^k_t, \quad t = 1, 2, \ldots, T, \quad n = 1, 2, \ldots, K. \tag{8}
\]

With the stage-based demand constraint, we can ensure that the demand can be satisfied in each stage. When \(n = K\), it is easy to see that (8) is the same to (2). Then, in the following discussion, we will not explicitly consider (2).
2.2.4. Objective Function. This problem aims to find the optimal allocation strategies with the minimized cost. As addressed in the illustration, the total cost in essence consists of two parts, namely, delivery cost in the network and storage cost in destinations.

The delivery cost refers to the expenses in the process of delivering empty cars. Since in stage $k$, the flow volume on link $(i, j)$ is denoted by $x_{ij}^k$, considering all the stages and links, we finally formulate the total delivery cost $f_1(X)$ below:

$$f_1(X) = \sum_{k=1}^{K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k$$  \hspace{1cm} (9)

In this problem, the freight empty cars will be delivered in different stages, which produce the following two situations. (1) One is that the traffic flow volume just meets the demand. That is, the delivered total empty cars are equal to the number of cars required in this stage. Then, no storage cost will be produced in this case. (2) The other one is that the number of empty cars is larger than that required in this stage. For this case, the redundant cars will be used in the next stage, resulting in the extra storage cost, which can be formulated as

$$e_1 \left( \sum_{(o,s) \in A} x_{id}^1 - U_i^1 \right)$$

$$+ e_2 \left( \sum_{(o,s) \in A} \left( x_{id}^1 + x_{id}^2 \right) - \left( U_i^1 + U_i^2 \right) \right)$$

$$+ \cdots + e_K \left( \sum_{(o,s) \in A} \left( x_{id}^1 + x_{id}^2 + \cdots + x_{id}^K \right) - \left( U_i^1 + U_i^2 + \cdots + U_i^K \right) \right).$$ \hspace{1cm} (10)

So the total storage cost at destination $d_i$ can be summarized as the following form:

$$\sum_{n=1}^{K} e_n \left( \sum_{(o,s) \in A} \sum_{k=1}^{n} x_{id}^k - \sum_{k=1}^{n} U_i^k \right).$$ \hspace{1cm} (11)

Considering all the destinations, we finally formulate the total storage cost as

$$f_2(X) = \sum_{t=1}^{T} \sum_{n=1}^{K} e_n \left( \sum_{(o,s) \in A} \sum_{k=1}^{n} x_{id}^k - \sum_{k=1}^{n} U_i^k \right).$$ \hspace{1cm} (12)

Then, the total cost for these two parts is

$$f(X) = f_1(X) + f_2(X).$$ \hspace{1cm} (13)

2.2.5. Mathematical Model. Consider the objective and constraints, the mathematical model of this problem can be formulated as follows:

$$\min \quad f(X)$$

$$\text{s.t.} \quad \sum_{(o,s) \in A} \sum_{k=1}^{K} x_{ij}^k \leq R_s, \quad s = 1, 2, \ldots, S$$

$$\sum_{(i,j) \in A} x_{ij} = \sum_{(j,i) \in A} x_{ij}^t,$$

$$i \in N/ \{ o, d_i, s = 1, 2, \ldots, S, t = 1, 2, \ldots, T \},$$

$$k = 1, 2, \ldots, K$$

$$\sum_{k=1}^{K} x_{ij}^k \leq Q_{ij}, \quad (i,j) \in A$$

$$\sum_{(i,j) \in A} \sum_{k=1}^{K} x_{ij}^k \leq G_i,$$

$$i \in N/ \{ o, d_i, s = 1, 2, \ldots, S, t = 1, 2, \ldots, T \}$$

$$\sum_{t=1}^{T} \sum_{n=1}^{K} x_{ij}^k \geq \sum_{k=1}^{n} U_i^k,$$

$$t = 1, 2, \ldots, T, \quad n = 1, 2, \ldots, K$$

$$x_{ij}^k \in \mathbb{Z}^+ \cup \{0\}.$$ \hspace{1cm} (14)

In this model, the objective is to minimize the total cost in the railway transportation system. The first two constraints guarantee the rationality of the transportation activities. The third and fourth constraints ensure that the amount of traffic flow volume cannot exceed the capacities of each link and each node, respectively. The fifth constraint ensures that the stage-based demand can be satisfied with the demand constraints.

2.3. Model Analysis. To discuss the complexity of the proposed formulation, the following discussion aims to analyze the characteristics of the proposed model. We firstly focus on the number of constraints in the model, which are displayed in Table 3.

An illustration will be given in the following to demonstrate the complexity of the proposed model. Consider a network with 20 nodes and 50 links. In this network, suppose that there are three origin stations and three destination stations. A total of three stages will be taken into consideration. With the assumption above, since a total of 9 paths need to be considered, the number of decision variables should be 150 over the entire network. Moreover, a total of 118 constraints need to be included in the model.

Note that when the network scale increases, the number of constraints and variables will increase to a great extent accordingly, which potentially leads to the difficulties in
Table 3: The amount of constraints in each constraint condition.

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>( S )</td>
</tr>
<tr>
<td>(3)</td>
<td>( K \cdot (</td>
</tr>
<tr>
<td>(4)</td>
<td>(</td>
</tr>
<tr>
<td>(5)</td>
<td>(</td>
</tr>
<tr>
<td>(8)</td>
<td>( K \cdot T )</td>
</tr>
<tr>
<td>Total number</td>
<td>( K \cdot (</td>
</tr>
</tbody>
</table>

calculation. Then, in order to simplify the computational complexity in the process of allocating cars, in the next section, we shall transform the arc-based solutions into route-based solutions, in which constraint (4) and constraint (5) can be merged into one constraint. Thus, it can decrease the complexity of the solution methods expectedly. The following section will design genetic algorithm to search for an approximate optimal solution.

3. Genetic Algorithm

In practice, it is rather difficult to solve the large-scale freight empty cars allocation problem efficiently. Because of this, we aim to design an approximation algorithm in this section, seeking a feasible solution which can replace the best solution approximatively.

The approximation algorithm is also called heuristic algorithms, which include tabu search algorithm, simulated annealing algorithm, genetic algorithm, neural network algorithm, and ant colony algorithm. As the genetic algorithm is a kind of direct searching method which does not depend on the specific characteristics of problems, it is efficient and effective in solving a variety of real-world problems. The principle of genetic algorithm is to simulate the mechanism of the living beings evolving and natural choosing. It was well developed by many researchers. Up to now, the genetic algorithm has been successfully used to solve practical optimization problems, such as transportation problem (Gen et al. [20]), machine scheduling problem (Vallada and Ruiz [21]), vehicle routing problem (Baker and Ayechew [22]), network optimization (Ukkusuri et al. [23]) and railway operation problem (Yang et al. [24, 25], Xu et al. [26]).

In general, the procedure of genetic algorithm is as follows: (1) randomly generate a certain number of chromosomes; (2) evaluate the quality of those chromosomes with fitness function according to some criteria; (3) obtain fine chromosomes via selection, crossover, and mutation operations. Figure 4 is the procedure of this algorithm.

3.1. Solution Representation. In solving the empty car allocation problem, we need firstly to determine the solution representation in the genetic algorithm. As described in the model, the solution is represented as link-based flow in each link. To simplify the complexity of computation, we here particularly introduce the path-based solution representation in the genetic algorithm. Since link-based flow can be finally split into the flow on different paths between different OD pairs (here, “OD pair” is an abbreviation for the origin and destination terminals), in this method, potential paths should be determined firstly, and then we determine the flow volume on each path to satisfy the demand of each destination.

An illustration is given to show the difference between link-based solution and path-based solution, depicted in Figure 5.

In Figure 5, each path consists of three nodes, namely, a supply station, an intermediate station and a demand station. Suppose that two-stage demands exist in this network. According to the link-based solution, there will be 4 variables and 8 constraints in total. If we set the variables and constraints in a path instead of in links, there will be only 2 variables and 5 constraints totally, both less than those in link-based representation. In reality, the scale of network structure will be more complex than this simple transportation network. So the number of decision variables with link-based representation is more than that with path-based representation. In order to reduce the algorithm complexity and obtain an approximate optimal solution, the following discussion particularly adopts the path-based solution representation.

In this procedure, a path, denoted by \( P \), is represented by a sequence of ordered nodes. For instance, the set \( P = \{1, 3, 5, 9\} \) denotes the following path:

\[
P : 1 \rightarrow 3 \rightarrow 5 \rightarrow 9.
\] (15)

In genetic algorithm, all the potential paths should be generated between each OD pair, which serve as the transport paths in the process of delivering empty cars.

In the genetic algorithm, each population consists of \( \text{pop size} \) feasible solutions as chromosomes. Since we use the path-based solution in the algorithm, each solution
in population will be generated randomly. Moreover, for
each generated solution, its feasibility will be checked by
transforming it into link-based solution. If all the constraints
can be satisfied, this solution is a feasible solution. Otherwise,
the solution will be regenerated randomly. At the beginning
of the algorithm, a total of pop\_size solutions should be
produced. In this paper, we set path-based decision variables
as nonnegative integers, denoting the number of empty cars
transferred on the corresponding path.

3.2. Selection Operation. In the genetic algorithm, the role
of selection operation is to produce a new population for
the following crossover and mutation operations. In the
initial population, the chromosomes are first ranked from the
good to the bad according to their objectives. Without loss
of generality, the rearranged chromosomes are denoted by
\(X_1, X_2, \ldots, X_{\text{pop\_size}}\), where \(X_1\) and \(X_{\text{pop\_size}}\) denote the best
and the worst chromosomes, respectively. The rank-based
fitness of each chromosome can be calculated by the following
formula:

\[
\text{fitness}(X_i) = \beta, \\
\text{fitness}(X_i) = (1 - \beta) \cdot \text{fitness}(X_{i-1}), \quad (16)
\]

where \(\beta \in (0, 1)\) is a predetermined parameter.

Here, we give the detailed procedure to show the selection
process below.

Step 1. Compute the fitness of each individual in population
fitness \(X_i), i = 1, 2, \ldots, \text{pop\_size}.

Step 2. Compute the cumulative probability \(q_l\) of each individual in population. And \(q_l\) can be calculated by the following formula:

\[
q_l = \frac{\sum_{i=1}^{l} \text{fitness}(X_i)}{\sum_{h=1}^{\text{pop\_size}} \text{fitness}(X_i)}, \quad l = 1, 2, \ldots, \text{pop\_size}, \quad (17)
\]

Step 3. Generate a random number \(r\) in \((0, 1)\).

Step 4. If \(r < q_1\), select \(X_1\). If \(q_{l-1} < r \leq q_l\), then chromosome
\(X_l\) will be selected as a member of the new population.

Step 5. Repeat Step 3 and Step 4 for \text{pop\_size} times.

3.3. Crossover Operation. Crossover operations aim to generate
a new population in the searching process. Before this
operation, the parents should be specified according to the
predetermined crossover probability \(P_c\). To this end, the
following procedure is designed.

Step 1. Let \(h = 1\).

Step 2. Randomly generate a number \(r\) in the interval \((0, 1)\).

Step 3. If \(r < P_c\), chromosome \(X_h\) is selected to take part in
crossover operation; otherwise, \(X_h\) will not be selected.

Step 4. Let \(h ++\), \(h \leq \text{pop\_size}\), go to Step 2.

Step 5. Record the selected chromosomes.

After the parent chromosomes are selected, any two
chromosomes can be used for crossover operation. Suppose
that the two parent chromosomes are \(X_1^l\) and \(X_2^l\) and
the children chromosomes are \(X_1^l'\) and \(X_2^l'\). The following process
is designed to perform the crossover operation.

Step 1. Randomly generate a parameter \(\alpha\) in interval \((0, 1)\), let
\(X_1^l' = \alpha \cdot X_1 + (1 - \alpha) \cdot X_2\)
and \(X_2^l' = (1 - \alpha) \cdot X_1 + \alpha \cdot X_2\).

Step 2. Reset \(X_1^l'\) and \(X_2^l'\) as integer vectors by rounding
operation for each element.

Step 3. Check the feasibility of the two children chromosomes
\(X_1^l'\) and \(X_2^l'\). If the children chromosomes are feasible, replace
the parents chromosomes by them; Otherwise, remain the
parents chromosomes.

Step 4. Record the new population.

3.4. Mutation Operation. In order to avoid premature conver-
gence, the mutation operation is necessary in the searching
process of genetic algorithm. Like the crossover operation,
we firstly need to determine the chromosomes for mutation
operations. The number of chromosomes selected to perform
mutation operation is also completely based on the predeter-
mined mutation probability \(P_m\).

Step 1. Let \(h = 1\).

Step 2. Randomly generate a number \(r\) in the interval \((0, 1)\).

Step 3. If \(r < P_m\), chromosome \(X_h\) is selected to take part in
mutation operation; otherwise, \(X_h\) is not the one.

Step 4. Let \(h ++\), \(h \leq \text{pop\_size}\), go to Step 2.

Step 5. Record the selected chromosomes.

For each selected parent chromosome \(X_h\), we are aiming
to generate the children chromosome \(X_h'\). The following
procedure is designed to perform mutation operations.

Step 1. Randomly generate a mutation direction vector \(\tilde{a}\), in
which all of the elements are generated in the interval \((-1, 1)\).

Step 2. Let \(X_h' = X_h + m \cdot \tilde{a}\), where \(m\) is a positive integer.

Step 3. Reset \(X_h'\) as an integer vector by rounding
operation for each element.

Step 4. Check the feasibility. If \(X_h'\) is feasible, let \(X_h'\) replace
\(X_h\). Otherwise, set \(m = m/2\), go to step 2.
3.5. Procedure of Algorithm. For the completeness of this paper, we shall give the detailed procedure of the algorithm in the following.

Step 1. Randomly generate pop_size chromosomes in the population.

Step 2. Perform the selection operation.

Step 3. Perform the crossover operation.

Step 4. Perform the mutation operation.

Step 5. Repeat Step 2 to Step 4 for a given number (denoted by Generation) of times.

Step 6. Output the best solution.

4. Model Test and Computational Results

In this section, we give two numerical examples with different scales to illustrate the effectiveness of the proposed methods, in which the experiments are implemented by using C++ software in a personal computer with Intel(R) Core(TM) i5-3317U 1.70 GHz.

4.1. Small-Scale Instance. In the first set of experiments, we consider a small-scale network, shown in Figure 2. This network consists of five nodes and four links, in which nodes 1 and 2 represent origin stations, node 3 denotes an intermediate station, and nodes 4 and 5 represent destination stations. In this network, it is easy to see that four potential paths exist between different OD pairs; that is, 1 → 3 → 4, 1 → 3 → 5, 2 → 3 → 4, 2 → 3 → 5.

In this network, we consider two-stage based demands. The detailed data are given as link 1 → 3: (3, 3, 65), link 3 → 4: (2, 5, 80), link 2 → 3: (2, 2, 55), and link 3 → 5: (3, 2, 40), where the numbers in brackets denote unit transportation cost for stage one, unit transportation cost for stage two, and transfer capacity of the link. Additionally, at transfer station 3, the turnover capacity is 150. In origin stations 1 and 2, the numbers of available cars are 110 and 60, respectively.

Besides, it is possible that the storage cost can occur in the allocation plan. Then, the unit storage cost for different stages are given in Table 4.

It is easy to see in this problem that there are a total of 8 variables and 13 constraints in the allocation process. As we formulate this problem as a linear programming model, we firstly use the Lingo software to calculate this problem, where the optimal objective turns out to be 830 and the optimal solution is given in Table 5.

In the following, we shall investigate the performance of the genetic algorithm. To this end, we particularly list the best solutions encountered during the first 16 generations to analyze the convergence of the algorithm, shown in Table 6.

In Table 6, it is easy to see that all decision variables can approach the best solution gradually in the searching process, and the approximate optimal objective can be quickly achieved in a short time. To further show the convergence process of the solution, Figure 6 gives the detailed variation of decision variables $x_{13}^1$ and $x_{13}^2$ in the first 16 iterations.

It follows from Figure 6 that, for decision variables $x_{13}^1$ and $x_{13}^2$, they can approach to their optimal values (obtained by Lingo software) quickly with a relative small undulation, and this tendency is almost kept in the following generations. Additionally, the encountered optimal objective will maintain 840 (see Figure 7) after about 500 generation, with +10 error to the best objective obtained by Lingo software, leading to a relative small error 1.2%, which shows the effectiveness of genetic algorithm for solving this problem.

4.2. A Large-Scale Instance in China. In the following, we propose a computational experiment derived from real-life instances of the railway network in Beijing, Hebei, and Shanxi province of China. In this instance, a total of 103 stations are located on the network. In particular, we omit some intermediate points for displaying convenience in Figure 8, where some important terminal stations are indexed by numbers, corresponding to their positions in the real-life network. In this network, a path can be represented by a sequence of terminal stations. Figure 9 gives an illustration for a path from station 1 to station 12, denoted by
In the real case, there still exist a total of 47 intermediate stations on this path (besides origins and destinations). In this set of experiments, suppose that there are two origins (node 1 and node 2) and three destinations (node 12, node 13, and node 14). We use the path-based solution for the empty car allocation process. Then, we first determine eleven potential paths for the solution process, listed in Table 7. Furthermore, the unit transportation cost is also given for different links. For instance, the first path consists of five links, then (3, 2, 1, 2, 4) represents the corresponding unit transportation cost on each link (here, we adopt invariant link unit transportation cost over different stages).

In addition, we assume that supply capacities at stations 1 and 2 are 275 and 240, respectively. The empty cars allocating process are divided into three stages, and stage-based demands and storage costs are listed in Table 8, respectively.

This set of experiment is implemented by genetic algorithm in C++ software on a personal computer. To test the robustness of the proposed algorithm, we perform ten experiments with different parameters in the genetic algorithm. The results are given in Table 9 for analysis convenience.

As shown in Table 9, by using different parameters, we can produce different approximation optimal solutions and objectives. When setting pop\_size = 30, \( P_c = 0.6 \), and \( P_m = 0.8 \), we finally produce the best objective 4985. In addition, the relative errors are also computed to show the performance of the algorithm, which can be calculated according to the following equation:

\[
\text{error} = \frac{\text{Current Objective} - \text{Best Objective}}{\text{Current Objective}} \times 100\%.
\]

Obviously, for different experiments, the errors of objective values are not larger than 1.52%. In particular, the average error and error variance are only 0.628% and 0.0022%, respectively, which shows the steadiness of the algorithm.

To analyze the searching process of the proposed algorithm, we especially consider the computational results with parameters \( P_c = 0.6 \) and \( P_m = 0.8 \). Moreover, three experiments with pop\_size = 15, 20, and 30, respectively, are implemented. Figure 10 gives the convergence of the optimal objectives with respect to different experiments.
Table 7: The considered paths and unit transportation cost on arcs.

<table>
<thead>
<tr>
<th>Number</th>
<th>OD Paths</th>
<th>Unit cost on arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 → 12</td>
<td>1 → 3 → 5 → 6 → 8 → 12</td>
</tr>
<tr>
<td>2</td>
<td>1 → 12</td>
<td>1 → 3 → 5 → 7 → 8 → 12</td>
</tr>
<tr>
<td>3</td>
<td>1 → 13</td>
<td>1 → 3 → 5 → 7 → 8 → 10 → 13</td>
</tr>
<tr>
<td>4</td>
<td>1 → 13</td>
<td>1 → 3 → 5 → 7 → 9 → 10 → 13</td>
</tr>
<tr>
<td>5</td>
<td>1 → 14</td>
<td>1 → 3 → 5 → 6 → 11 → 14</td>
</tr>
<tr>
<td>6</td>
<td>2 → 12</td>
<td>2 → 4 → 6 → 8 → 12</td>
</tr>
<tr>
<td>7</td>
<td>2 → 12</td>
<td>2 → 4 → 7 → 8 → 12</td>
</tr>
<tr>
<td>8</td>
<td>2 → 13</td>
<td>2 → 4 → 6 → 8 → 10 → 13</td>
</tr>
<tr>
<td>9</td>
<td>2 → 13</td>
<td>2 → 4 → 7 → 8 → 10 → 13</td>
</tr>
<tr>
<td>10</td>
<td>2 → 13</td>
<td>2 → 4 → 7 → 9 → 10 → 13</td>
</tr>
<tr>
<td>11</td>
<td>2 → 14</td>
<td>2 → 4 → 6 → 11 → 14</td>
</tr>
</tbody>
</table>

Table 8: The demand and unit storage cost in different stages.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Time period</th>
<th>Node 12</th>
<th>Node 13</th>
<th>Node 14</th>
<th>Storage cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8:00–16:00</td>
<td>75</td>
<td>50</td>
<td>40</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>16:00–24:00</td>
<td>90</td>
<td>85</td>
<td>50</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>24:00–8:00</td>
<td>35</td>
<td>20</td>
<td>30</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 9: The comparison of the optimal objectives.

<table>
<thead>
<tr>
<th>Pop_size</th>
<th>$P_c$</th>
<th>$P_m$</th>
<th>Generation</th>
<th>Objective value</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.4</td>
<td>0.4</td>
<td>1000</td>
<td>5062</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0.4</td>
<td>0.4</td>
<td>1000</td>
<td>5036</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>0.6</td>
<td>0.4</td>
<td>1000</td>
<td>5036</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.6</td>
<td>0.4</td>
<td>5000</td>
<td>5022</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>0.6</td>
<td>0.4</td>
<td>1000</td>
<td>5010</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>0.6</td>
<td>0.8</td>
<td>1000</td>
<td>5026</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>0.6</td>
<td>0.8</td>
<td>5000</td>
<td>4994</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>0.8</td>
<td>0.4</td>
<td>1000</td>
<td>5010</td>
</tr>
<tr>
<td>9</td>
<td>30</td>
<td>0.6</td>
<td>0.8</td>
<td>2000</td>
<td>4985</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
<td>0.6</td>
<td>0.8</td>
<td>5000</td>
<td>4985</td>
</tr>
<tr>
<td>Average value</td>
<td>0.628</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variance</td>
<td>0.0022</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is easy to see from this figure that, with different scales of population, the approximate optimal solution can be quickly achieved almost within 4000 generations. As expected, when we set a larger population, the optimal solution can be obtained with less generation in the searching process.

Finally, we give a comparison with the single-stage model to further show the effectiveness of the proposed approaches in this study. Specifically, the sum of the stage-based demands given in Table 8 will be considered as the demand in the single-stage model, and storage cost will be omitted in the process of allocating empty cars. Under this consideration, the best objective value for allocating empty cars turns out to be 7130 by using Lingo optimization software, which enhances the total transportation cost by 30% compared with the stage-based model.

5. Conclusions

This paper investigated a railway freight empty car allocation problem in the dynamic decision-making environment, in which the stage-based demands are considered, and the cost is divided into transfer cost and storage cost at destinations. To characterize this problem mathematically, an integer linear programming model was formulated with the minimization of total involved cost based on the network flow optimization. To generate approximate optimal solutions, a genetic algorithm with path-based solution representation is
designed to seek the optimal empty car distribution strategies in railway networks. The numerical experiments are executed to show the performance of the proposed approaches. In particular, compared to the single-stage model, the total transportation cost in our model can be reduced by almost 30%, which implies the effectiveness of the proposed approaches.

The future research can be focused on the following two aspects. (1) The model can be generalized to the uncertainty environments within the framework of uncertain programming methods, since the real-life decision systems for railway operations are essentially in the state of uncertainty (see Yang et al. [19, 25] and Li et al. [27]). (2) More exact algorithms can be expectedly developed to further enhance the quality of the generated solutions.

**Conflict of Interests**

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

**Acknowledgments**

This research was supported by the National Natural Science Foundation of China (no. 71271020), Research Foundation of State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University (no. RCS2014ZT02), the National Basic Research Program of China (no. 2012CB725400), and the Beijing Natural Science Foundation (no. 9144031).

**References**


Submit your manuscripts at http://www.hindawi.com