inventions

MDPI

Article

XGBoost-Based Heuristic Path Planning Algorithm for Large Scale Air–Rail Intermodal Networks

Shengyuan Weng [1], Xinghua Shan [2,*], Guangdong Bai [2], Jinfei Wu [2] and Nan Zhao [2]

[1] Postgraduate Department, China Academy of Railway Sciences, Beijing 100081, China; wengshengyuan@gmail.com
[2] Institute of Computing Technologies, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China; 15201253362@163.com (G.B.); wujinfei@rails.cn (J.W.); xingxuanjimeng@126.com (N.Z.)
* Correspondence: shanxinghua@rails.cn

Abstract: It is particularly important to develop efficient air–rail intermodal path planning methods for making full use of the advantages of air–rail intermodal networks and providing passengers with richer and more reasonable travel options. A Time-Expanded Graph (TEG) is used to model the timetable information of public transportation providing a theoretical basis for public transportation path planning. However, if the TEG includes a large amount of data such as train stations, airports, train and air schedules, the network scale will become very large, making path planning extremely time-consuming. This study proposes an XGBoost-based heuristic path planning algorithm (XGB-HPPA) for large scale air–rail intermodal networks, which use the XGBoost model to predict transfer stations before path planning, and quickly eliminate unreasonable transfer edges by adding a heuristic factor, reducing the network scale, thus accelerating the computation speed. Comparative results indicate that XGB-HPPA can markedly enhance computational speed within large-scale networks, while obtaining as many valid solutions as possible and approximating the optimal solution.

Keywords: path planning algorithm; XGBoost; intermodal transportation; air–rail transportation network; heuristics algorithm

Academic Editor: Anastasios Doulamis

Received: 11 January 2025
Revised: 26 January 2025
Accepted: 4 March 2025
Published: 7 March 2025

Citation: Weng, S.; Shan, X.; Bai, G.; Wu, J.; Zhao, N. XGBoost-Based Heuristic Path Planning Algorithm for Large Scale Air–Rail Intermodal Networks. *Inventions* **2025**, *10*, 27. https://doi.org/10.3390/inventions10020027

1. Introduction

With the rapid urbanization and economic development in China, the demand for efficient passenger transport has never been greater. Air–rail intermodal transport is perceived as a strategic approach to accommodate the escalating demand for efficient travel. China's high-speed rail network, which is among the world's leading infrastructures, covers over 45,000 km and connects major cities with unprecedented speed and reliability, making the scale of China's air–rail intermodal network considerable [1]. Launching integrated passenger travel services is beneficial for making full use of the resources of various modes of transportation to form complementary advantages, providing passengers with a more rational and convenient new service model for cross-modal travel. To fully leveraging the transportation advantages of both air and rail and maximize the value of the air–rail intermodal network, it is extremely important to study efficient methods for air–rail intermodal path planning.

Fundamentally, the air–rail intermodal path planning problem entails identifying the K shortest paths within an air–rail intermodal network [2]. In the research on shortest path search algorithms, scholars both domestically and internationally have conducted extensive research on the problem, with primary emphases on the construction of networks [3] and

the search for the K shortest paths [4,5]. Among these, the shortest path search algorithms have become relatively mature as core issues, such as Dijkstra's algorithm [6], Breadth First Search (BFS) [7], Depth First Search (DFS) algorithm [8], A* algorithm [9], as well as various heuristic algorithms such as Ant Colony algorithm [10], Genetic Algorithms [11], and hierarchical search algorithms like Heuristic Hierarchical Search (HHS) [12].

As long as a rational air–rail intermodal network structure is designed and the shortest path search algorithm is implemented on the designed network, the result of path planning can be obtained, these shortest path search algorithms have also been widely applied in the research of intermodal route planning. Some studies take stations and airports as nodes and take air routes and railway tracks as edges, constructing an intermodal transport network for path planning. Xu designed an air–rail network model with minimum total transportation cost as the objective function and designed an algorithm to search for paths with minimum total transportation cost [13]. Zheng designed an air–rail intermodal network model based on a hyper-graph model [14]. These algorithms mainly focus on quickly obtaining the optimal path in a static network, they do not fully consider the impact of factors such as hub connection conditions, operation time constraints between different transportation modes, and passengers' transfer preferences, which cannot meet actual travel planning needs [15,16]. Therefore, some scholars have proposed using a Time-Expanded Graph (TEG) to represent the intermodal transport route planning problem involving transfers, which has been widely applied in the optimization of public transportation routes, transfer strategies, and other areas, achieving good results [17]. However, when using TEG to construct an air–rail intermodal network, the number of nodes and edges in the network can become large. China has nearly 3000 train stations with over 10,000 trains operating daily. At the same time, it has about 300 civil airports with approximately 20,000 flights operating daily, so the network scale would be extremely large. Moreover, in practical application scenarios, a single path is hardly sufficient to meet the travel plans of passengers, we need to provide multiple alternative paths to meet the travel needs of passengers for selection. Considering the network scale and computational complexity, it is difficult to quickly calculate all the paths in such a vast network. Therefore, high computational efficiency is required for the shortest path search algorithms. Some scholars have attempted to accelerate route planning by reducing the scale of the network; Geisberger et al. proposed the Contraction Hierarchies algorithm to accelerate path planning [18]. However, the process of network contraction requires determining the weights of edges, which cannot meet the dynamic weight requirements during route planning. Other scholars have turned to heuristic algorithms to speed up the route planning process. Shi et al. proposed an improved A* algorithm to improve the searching efficiency by reducing the route searching space through the heuristic function [19]. Current research generally uses spatial distance, user preferences, and other conditions as heuristic factors to make path planning converge more quickly to the optimal solution. However, there is relatively little research on using the prediction results of models similar to XGBoost as heuristic factors to accelerate path planning algorithms.

In this study, we propose an XGBoost-based heuristic path planning algorithm (XGB-HPPA). Specifically, to address the need for swiftly solving the K shortest paths from an extremely large-scale network within a limited timeframe. The XGB-HPPA algorithm utilizes the XGBoost model to predict potential transfer stations, and the heuristic factors of relative transfer edges are set according to the predicted results in the route planning process, quickly eliminating unreasonable transfer edges, thus accelerating calculation speed.

The structure of the paper is as follows. In Section 2, we provide essential information of the air–rail intermodal path planning problem and the mathematical model we build based on TEG and describe the methods. In Section 3, we analyze the results of the

numerical experiments. Finally, we close the paper with some conclusions and future research ideas in Section 4.

2. Methodology

2.1. Problem Description

Air–rail path planning entails furnishing passengers with a multi-modal transportation plan spanning from the point of origin to the destination terminal. This kind of problem needs to consider many factors, such as operational time constraints between disparate transportation modes and hub connectivity constraints, etc., in order to provide effective results. The purpose of this study is to obtain effective results based on air–rail intermodal networks with high search efficiency.

2.2. Network Design

2.2.1. Time-Expanded Graph Based Network

Conventional transport networks employ weighted and directed graphs wherein nodes represent stations and airports, and edges represent air routes and railway tracks. Factors such as travel time and expenses are considered as generalized travel costs for the edges, which are used to find the optimal travel plan. This methodology, however, fails to directly encapsulate the connection information pertaining to the operational times of vehicles and the transfer behaviors of passengers within stations. Therefore, it is necessary to dynamically adjust the weights of edges based on the connection status of the running time of vehicles at both ends of the node, which poses a problem of complex calculation methods.

This study endeavors to model the air–rail intermodal transportation network based on TEG forming a directed graph network $N = (V,\ W)$, where V denotes the set of nodes representing various events during the passengers' traveling process, and W represents the set of edges symbolizing the transitions between these events. S is the set of stations and airports, and U is the collection of vehicles.

The node set V is constituted of four types of node sets, $V = E \cup L \cup A \cup D$. The specific explanation is as follows:

1. $E = (e_1, e_2, e_3, \ldots, e_s), s \in S$, e_s represents the set of events corresponding to passengers entering a station or airport s.
2. $L = (l_1, l_2, l_3, \ldots, l_s), s \in S$, l_s represents the set of events corresponding to passengers leaving a station or airport s.
3. $A = \left(a_1^1, a_2^1, a_1^2, \ldots, a_s^u\right), s \in S, u \in U$, a_s^u represents the set of events corresponding to vehicle u arriving at a station or airport s.
4. $D = \left(d_1^1, d_2^1, d_1^2, \ldots, d_s^u\right), s \in S, u \in U$, d_s^u represents the set of events corresponding to vehicle u departing from a station or airport s.

Let $t(v)$ be the time of occurrence of the event v ($v \in V$), the process between events (edges) can be represented by (v_i, v_j). Based on the different types of nodes at the start and end of the edges, the edges can be categorized into the following six types:

1. (e_s, a_s^u) represents the entering process in which a passenger enters a station and waits for the arrival of the train or airplane.
2. $(a_s^u,\ l_s)$ represents the leaving process in which a passenger gets off a train or airplane and ultimately leaves the station or airport.
3. $(a_s^u,\ d_s^u)$ represents the waiting process in which a train or an airplane arrives at a station or airport, stops for a period of time and departs, during which the passenger remains inside the vehicle without leaving.

4. $(d_s^u,\ a_s^u)$ represents the hopping process in which a passenger travels in a vehicle from one station or airport to the next.
5. $(a_s^u,\ d_s^u)_t$ represents the intra-station transfer process in which a passenger transfers to a different mode of transportation without leaving the station or airport.
6. $(l_s,\ e_s)$ represents the inter-station transfer process in which a passenger transfers between two different stations or an airport to continue their journey on different modes of transportation.

We denote the time taken for the event transformation process (edge) as $f(w)$. For transportation modes adhering to a fixed timetable, such as trains and airplanes, a comprehensive TEG can be meticulously designed based on the timetable. The schematic of the network structure is illustrated in Figure 1.

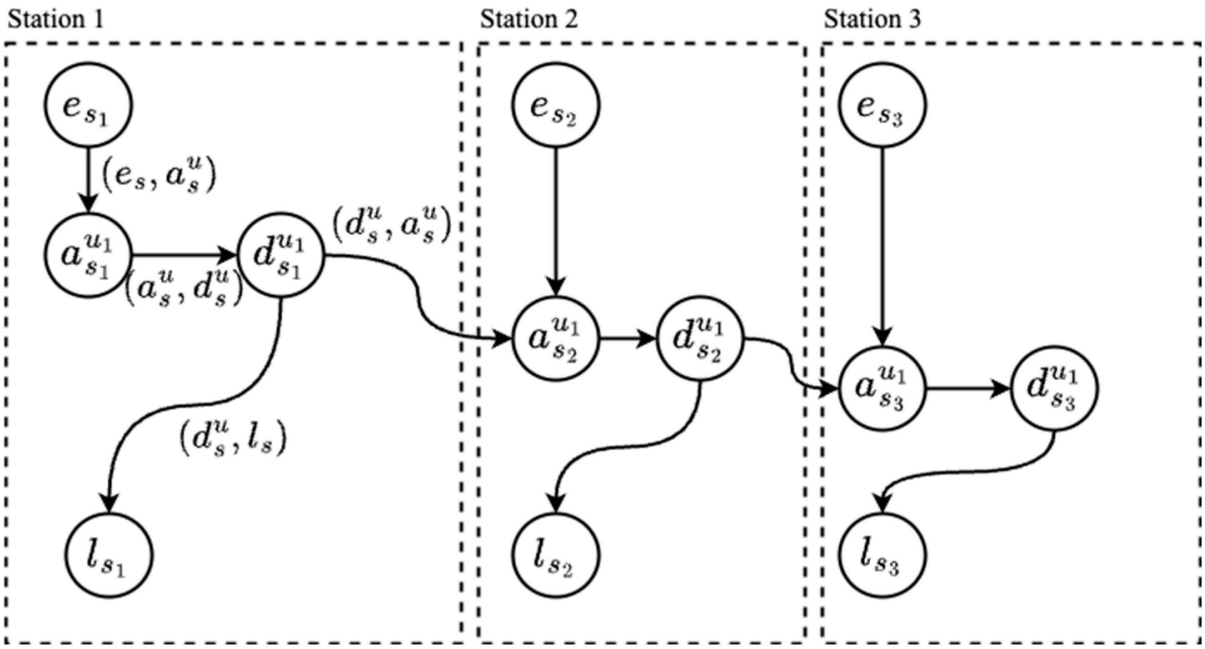

Figure 1. A timetable-based passenger transportation network diagram.

In Figure 1, the occurrence times of events like a_s^u and d_s^u are determined by the timetable. The latest arrival time of event e_{s_1} is determined by the actual time needed for passengers to enter station s_1 and $t\left(a_{u_1}^{s_1}\right)$, the time of occurrence of the event $a_{u_1}^{s_1}$. The earliest departure time of event l_{s_3} is determined by the actual time needed for passengers to leave station s_3 and $t\left(d_{s_3}^{u_1}\right)$, the time of occurrence of the event $d_{s_3}^{u_1}$.

2.2.2. Transfer Edge Generation

Based on the train and airplane schedules generated for TEG, they are independent of each other, and we cannot get the air–rail intermodal route out of two separated TEGs. We establish transfer edges that connect nodes both within the same TEG and across different TEGs, thereby facilitating air–rail intermodal travel path planning. Establishing transfer edges can create potential quicker paths by transfer between different stations, breaking the constraints of same-station transfers to support the generation of more optimal travel plans. An example of a TEG graph with transfer edges is illustrated in Figure 2.

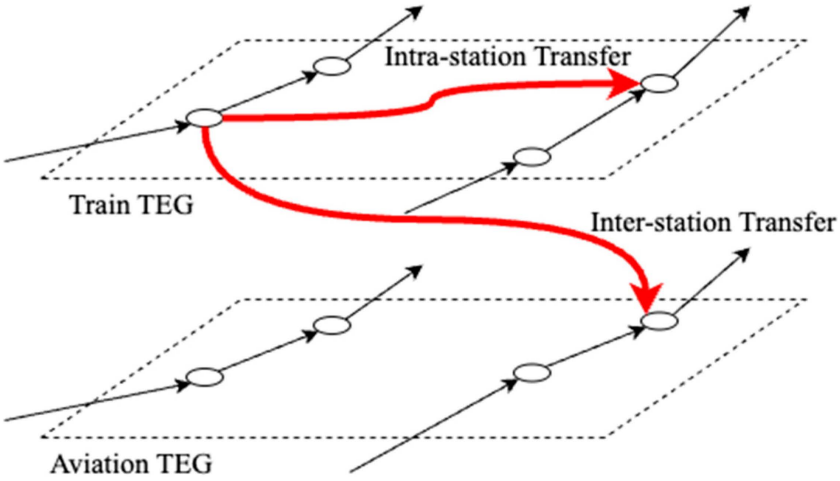

Figure 2. An example of a TEG graph with transfer edges.

The condition for intra-station transfers and inter-station transfers differ, and so do the strategies for establishing transfer edges. Different edge establishing strategies need to be selected based on actual conditions.

The intra-station transfer strategy is suitable for passengers who achieve transfers within the same stations through quick transfer corridors. In the transportation network, it is represented by the edge $(a_s^u, d_s^u)_t$, and the structure of the intra-station transfer is illustrated in Figure 3.

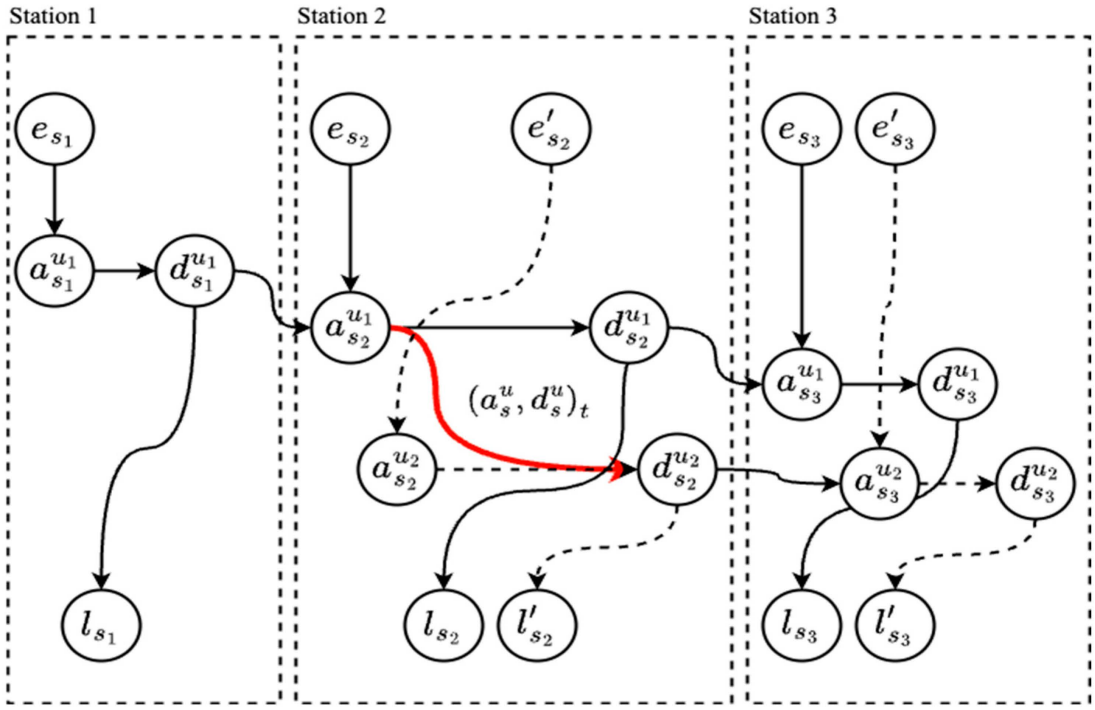

Figure 3. Schematic diagram of intra-station transfer mode.

Inter-station transfer is suitable for passengers transferring between different stations, or a transfer process in which procedures such as security checks are required within the same station. In the transportation network, it is represented by the edge (l_s, e_s), and the structure of the inter-station transfer is shown in Figure 4.

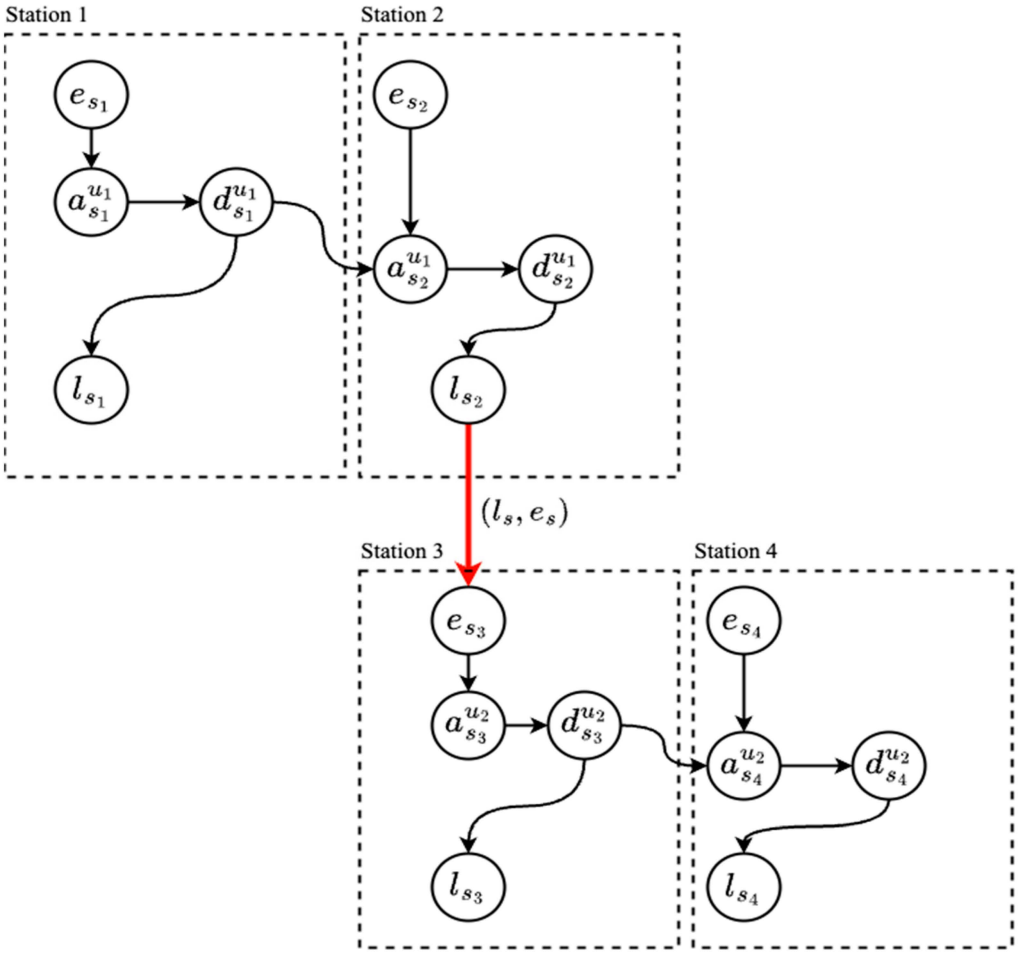

Figure 4. Schematic diagram of inter-station transfer mode.

Establishing transfer edges not only increases the number of available travel options but also significantly increases the number of nodes and edges in the existing network, however, it introduces a significant computational challenge, as it prolongs the time required for path planning calculations. Therefore, it is crucial to establish transfer edges reasonably to ensure the quality and speed of the path planning results. It is advisable to implement differentiated strategies for the generation of transfer edges, tailored to the specific modes of transport being interconnected.

Intermodal transfers between railway trains and airplanes support both intra-station and inter-station transfer modes. To keep the scale of the air–rail intermodal network within a reasonable range, the number of transfer paths established should be limited. By defining the maximum permissible transfer distance between stations and airports, and by setting the allowable transfer time intervals for trains and airplanes, we can control the number of transfer edges within a certain range, thereby controlling the overall scale of the air–rail intermodal network, ensuring its operational feasibility and efficiency.

Establishing intra-station transfer edges must meet the following constraints:

$$t\left(d_s^{u_2}\right) - t\left(a_s^{u_1}\right) \geq T \tag{1}$$

where

T represents the minimum transfer time interval between different modes of transportation within the station, with the unit being minutes (min).

$t\left(d_s^{u_2}\right)$ represents the time when the train u_2 arrives at the station s.

$t\left(a_s^{u_1}\right)$ represents the time when the train u_1 arrives at the station s.

Establishing inter-station transfer edges must meet the following constraints:

$$\begin{cases} d(s_i, s_j) \geq D \\ t\left(d_{s_j}^{u_2}\right) - t\left(d_{s_i}^{u_1}\right) \geq T, s_i = s_j, u_1 \neq u_2 \\ t\left(d_{s_j}^{u_2}\right) - t\left(d_{s_i}^{u_1}\right) \geq g(s_i, s_j)^* + f\left(a_{s_i}^{u_1}, l_{s_i}\right) + f\left(e_{s_i}, d_{s_j}^{u_2}\right), s_i \neq s_j, u_1 \neq u_2 \end{cases} \tag{2}$$

where

$d(s_i, s_j)$ represents the straight-line distance between station s_i and station s_j, with the unit being kilometers (km).

D is the maximum distance within which transfer paths can be established based on the types of preceding and succeeding stations, with the unit being kilometers (km).

$g(s_i, s_j)^*$ represents the estimated shortest navigation time from station s_i to station s_j, with the unit being minutes (min).

$f\left(a_{s_i}^{u_1}, l_{s_i}\right)$ represents the time taken for a passenger to travel from station s_i on transportation mode u_1 until they leave the station, with the unit being minutes (min).

$f\left(e_{s_i}, d_{s_j}^{u_2}\right)$ represents the time taken for a passenger to enter station s_j until they leave the station on transportation mode u_2, with the unit being minutes and seconds (min).

2.3. General Travel Cost

The function representing general travel cost for edge $w = (v_1, v_2)$ is shown as follows:

$$C(v_1, v_2) = \left[1 + \alpha_{(v_1, v_2)} + \beta_u^{(v_1, v_2)}\right] \times f(v_1, v_2) \tag{3}$$

where

$\alpha_{(v_1, v_2)}$ is the transfer cost coefficient. If (v_1, v_2) is not a transfer edge, then $\alpha_{(v_1, v_2)} = 0$. If (v_1, v_2) is a transfer edge and passengers prefer to minimize the number of transfers, then $\alpha_{(v_1, v_2)} > 1$.

$\beta_u^{(v_1, v_2)}$ is the cost coefficient of the transportation vehicle u on (v_1, v_2). If the type of transportation corresponding to (v_1, v_2) matches the passenger's preference, then let $\beta_u^{(v_1, v_2)} < 1$, otherwise let $\beta_u^{(v_1, v_2)} \geq 1$.

In summary, besides the basic travel time cost, there will be lower additional costs on the paths that match the passengers' travel preferences.

2.4. The Mathematical Model

In TEG, a passenger's journey commences at nodes symbolizing the event e_s and concludes at nodes indicating the event l_s. The path taken between nodes represents the travel plan of the passenger. The path is denoted as follows:

$$P = (v_1, v_2) + (v_3, v_4) + \cdots + (v_{m-1}, v_m) \tag{4}$$

The "+" symbol denotes the concatenation of transitions across various segments of the journey.

The objective is to identify paths between the e_s and the node l_s that minimize the aggregate cost. To sum up the generalized costs of each segment of the scheme P to represent the total cost of the route, denoted as $|P|$:

$$|P| = C(v_1, v_2) + C(v_3, v_4) + \cdots + C(v_{m-1}, v_m) \tag{5}$$

In summary, the general description of the air–rail intermodal path planning problem is as follows: Based on the given schedule data of trains and airplanes, construct the air–rail

intermodal network G in the form of TEG, set up the generalized cost function C, locate the e_s node and the l_s node corresponding to the user's search criteria, and try to find path P with lowest total cost $|P|$.

2.5. XGBoost-Based Heuristic Path Planning Algorithm

2.5.1. General Travel Cost Function with Heuristic Factor

XGBoost is the short name for 'Extreme gradient boosting' proposed by Chen and Guestrin and has a recognized impact in solving machine learning challenges in different application domains [20].

The XGB-HPPA proposed in this study utilizes the XGBoost model to forecast the most suitable stations or airports to serve as potential transfer stations, based on user inputs, with the ultimate goal of reaching the destination terminal. The heuristic factor of the transfer edges is set to quickly eliminate unreasonable edges in the path planning process, thereby accelerating the speed of calculation. The general travel cost function $C(w)$ for the edge $w = (v_1, v_2)$ is rewritten as follows:

$$C(v_1, v_2) = \left[1 + \alpha_{(v_1,v_2)} + \beta_u^{(v_1,v_2)}\right] \times f(v_1, v_2) \times \rho_{(v_1,v_2)} \tag{6}$$

where

$\rho_{(v_1,v_2)}$ is a heuristic factor which will be updated by the XGBoost model, with a default value of 1.

2.5.2. XGBoost Algorithm

Generally, XGBoost model is described as follows [21]:

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta) \tag{7}$$

where

$Obj(\Theta)$ is the objective function.

$L(\Theta)$ measures how well the model fit on training data.

$\Omega(\Theta)$ measures the complexity of the model.

The complexity is defined as:

$$\Omega(f) = \gamma T + \tfrac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \tag{8}$$

where

T is the number of leaf nodes.

γ is the penalty conefficient of the number of leaves.

λ is the penalty coefficient of regularization.

ω_j is tab1he score of leaf j.

In this study, we use an XGBoost multi-class classification model to predict potential transfer stations. It is defined as:

$$\hat{y}_i = F_k(x_i) = F_{k-1}(x_i) + f_k(x_i) \tag{9}$$

Let $f_k(x)$ represents the kth decision-tree based learner. The objective function of XGBoost is defined as follows:

$$Obj = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{k=1}^{K} \Omega(f_k) \tag{10}$$

The optimization goal of the model is to find the optimal $f(x_i)$ so that the objective function Obj is minimized. XGBoost adopts the method of objective function approximation. For this purpose, Equation (10) is rewritten as follows:

$$Obj^{(s)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(s-1)} + f_s(x_i)\right) + \Omega(f_s) \tag{11}$$

Equation (11) is further subjected to second-order Taylor expansion:

$$Obj^{(s)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(s-1)} + g_i f_s(x_i) + \frac{1}{2} h_i f_s^2(x_i)\right) + \Omega(f_s) \tag{12}$$

where

$g_i = \partial_{\hat{y}_i^{(s-1)}} l\left(y_i, \hat{y}_i^{(s-1)}\right)$, which is the first order partial derivative of the function.

$h_i = \partial^2_{\hat{y}_i^{(s-1)}} l\left(y_i, \hat{y}_i^{(s-1)}\right)$, which is the second order partial derivative of the function.

Since the constant term does not affect the optimization results of the model, Equation (12) is simplified as follows:

$$\begin{aligned} Obj^{(s)} &= \sum_{i=1}^{n} \left[g_i f_s(x_i) + \frac{1}{2} h_i f_s^2(x_i)\right] + \Omega(f_s) \\ &= \sum_{i=1}^{n} \left[g_i f_s(x_i) + \frac{1}{2} h_i f_s^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \\ &= \sum_{j-1}^{T} \left[\left(\sum_{i \in I_j} g_i\right)\omega_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right)\omega_j^2\right] + \gamma T \end{aligned} \tag{13}$$

where $I_j = \{i | q(x_i) = j\}$ is an instance set assigned to the jth leaf. We find Equation (13) is a unary quadratic function with independent variable ω_j and dependent variable $Obj^{(s)}$. The optimal ω_j^* of leaf node j is:

$$\omega_j^* = \frac{\sum_{i \in I_j} g_i}{-2 \times \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right)} = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{14}$$

The optimal objective function value of $Obj^{(s)}$ is:

$$Obj^{(s)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{15}$$

In Equations (14) and (15), G_j denotes $\sum_{i \in I_j} g_i$, and H_j denotes $\sum_{i \in I_j} h_i$, then:

$$\omega_j^* = -\frac{G_j}{H_j + \lambda} \tag{16}$$

$$Obj^{(s)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{{G_j}^2}{H_j + \lambda} + \gamma T \tag{17}$$

Equation (17) is the scoring function of a decision-tree-based learner with training. $Obj^{(s)}$ denotes the aggregate of all objective function values at the leaf nodes. For each iteration of the training process, the optimal decision tree model can be obtained by calculating the evaluation scores for all candidate decision tree models. To solve the problem in which a number of candidate decision trees is infinite, we use a greedy algorithm.

When there is only one node, the objective function becomes as follows:

$$Obj_j = -\frac{1}{2}\frac{{G_j}^2}{H_j+\lambda} + \gamma \tag{18}$$

The difference between the objective functions of two trees before and after a node splitting is that the splitting node splits into two new nodes, with the rest of the nodes remaining invariant. After the node is split into two child nodes, the objective function contribution of the two child nodes becomes:

$$Obj_s = -\frac{1}{2}\left(\frac{G_{jL}^2}{H_{jL}+\lambda} + \frac{G_{jR}^2}{H_{jR}+\lambda}\right) + 2\gamma \tag{19}$$

The objective function changes to:

$$Obj_{split}^{(j)} = Obj_j - Obj_s = \frac{1}{2}\left(\frac{G_{jL}^2}{H_{jL}+\lambda} + \frac{G_{jR}^2}{H_{jR}+\lambda} - \frac{G_j^2}{H_j+\lambda}\right) - \gamma \tag{20}$$

Finally, the objective function can be rewritten as follows:

$$Obj_{split} = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right) - \gamma \tag{21}$$

2.5.3. Evaluation Index Construction

For the multi-class classification problem, True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) are distinct combinations of the real category of data samples and the model prediction category [22]. It is usually necessary to calculate these metrics for each class individually and then aggregate them through methods such as micro-averaging or macro-averaging, according to the actual needs of the research problem. The confusion matrix is shown in Table 1.

Table 1. Confusion matrix.

	Positive	**Negative**
True	True Positive (TP)	True Negative (TN)
False	False Positive (FP)	False Negative (FN)

This study uses Accuracy, Recall, Precision, and F-Score to evaluate the quality of the model. The indicators of the model are defined as follows.

- Accuracy is the ratio of correct prediction. The formula is expressed as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{22}$$

- Recall is the possibility of finding all positive samples. The formula is expressed as:

$$Recall = \frac{TP}{TP+FN} \tag{23}$$

- Precision as the correlation of classifying a negative sample as a positive is measured. The formula is expressed as:

$$Precision = \frac{TP}{TP+FP} \tag{24}$$

- F-Score considers both precision and recall indicators and is expressed as:

$$F = \frac{\left(\alpha^2+1\right)P\times R}{\alpha^2(P+R)} \tag{25}$$

- When the parameter $\alpha = 1$, the most common F1 is the harmonic mean of precision and recall. The formula is expressed as:

$$F1 = \frac{2 \times P \times R}{P + R} \tag{26}$$

2.5.4. Heuristic Route Algorithm

In XGB-HPPA, prior to the search for the K shortest paths, an initial step involves training an XGBoost model leveraging users' historical travel data. This model is capable of recommending transfer stations by classifying based on the user's origin and destination station information.

Since the process of prediction transfer stations is a multi-class classification problem, the model's accuracy is usually not high when there are too many target classes. Therefore, we use the XGBoost model to predict transfer cities instead of stations first, thus limiting the number of target classes, and then all stations within the city are considered as recommended transfer stations. For these recommended stations, we regard all transfer edges originating from them as recommended, and we adjust ρ factors of these transfer edges. When all transfer edges' ρ factors are set, then we search for K shortest paths. The specific processes are shown in Figure 5.

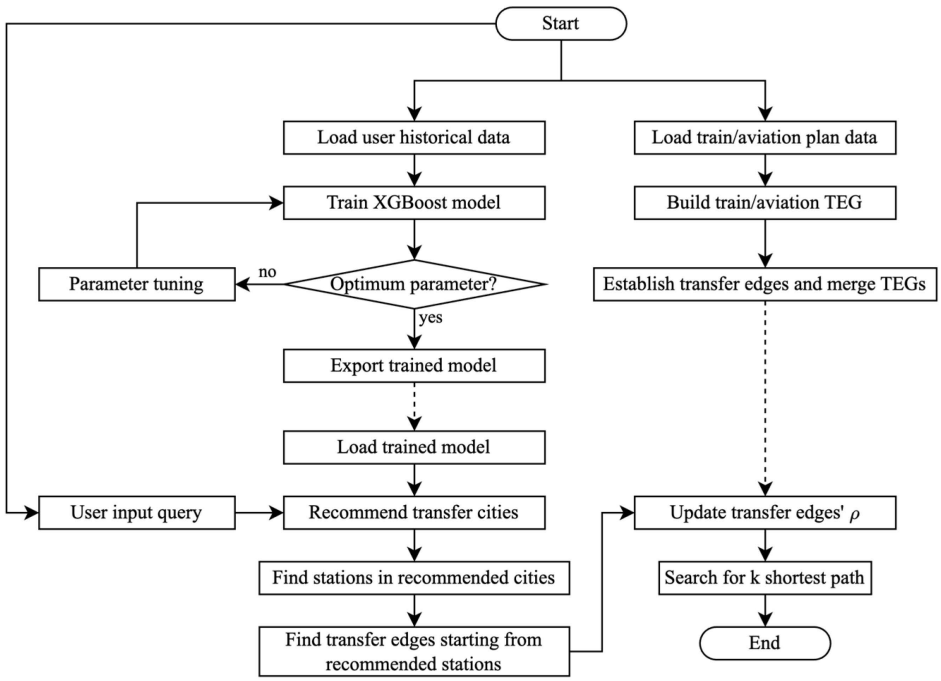

Figure 5. The XGB-HPPA algorithm flow chart.

1. Load railway and airplane operation plan data, generate TEG for train and airplane.
2. Establish transfer edges between train and airplane TEG to form an air–rail integrated TEG.
3. Load users' historical data.
4. Train the XGBoost model and perform parameter tuning.
5. Use the pre-trained XGBoost model to determine which transfer edges can be kept based on the user's query conditions.
6. Update ρ of the edges that will be kept to 1 and set the ρ of the other transfer edges to $+\infty$.
7. Run K shortest path algorithm, this paper utilizes the bidirectional Dijkstra for the K shortest path search process.

3. Numerical Experiments and Analysis

3.1. Data Prepareation

3.1.1. Air–Rail Intermodal Network Construction Dataset

We obtained all the train and civil aviation flight data from October to December 2023. Based on the rules described in this paper, we constructed an integrated air–rail multimodal network in TEG (Time-Expanded Graph) mode. The data includes 3174 railway stations, 259 airports, 23,748 train services, and 5800 civil aviation flights. The final integrated air–rail multimodal network consists of 2,386,214 nodes and 23,690,182 edges.

3.1.2. XGBoost Model Training Dataset

To train the XGBoost model, we first need to obtain the itinerary data of passengers using air–rail intermodal transport. We consider it as a single air–rail intermodal transport journey if the trip combination meets the following conditions:

1. The first segment and the second segment of the journey are by train and by airplane separately.
2. The arrival time of the first segment of the journey and the departure time of the second segment differ by less than 4 h.
3. The straight-line distance between the destination of the first segment of the journey and the origin of the second segment is within 50 km.

This study utilizes passenger air and train ticket purchase data sourced from China's online ticketing platform between October 2023 and December 2023, then processed according to the method mentioned before to form an air–rail intermodal transport dataset which contains 450,000 valid entries. The dataset includes the start and terminal station information as well as the information about transfer locations during intermodal travel.

In practical application contexts, passengers generally use the start and terminal station as search criteria. Accordingly, this study only focuses on information related to the starting and terminal station as model inputs. Table 2 below summarizes the basic information on the features used for this study.

Table 2. Summary of the basic information on the features used for this study.

Feature	Definition	Attribute
$Label_{start}$	The unique identification label of start station.	Categorical
$Label_{terminal}$	The unique identification label of terminal station.	Categorical
$City_{start}$	The unique identification label of start city.	Categorical
$City_{terminal}$	The unique identification label of terminal city.	Categorical
$Province_{start}$	The unique identification label of start province.	Categorical
$Province_{terminal}$	The unique identification label of terminal province.	Categorical
$Latitude_{start}$	The latitude of the start station.	Float
$Longitude_{start}$	The longitude of the start station.	Float
$Laatitude_{terminalt}$	The latitude of the terminal station.	Float
$Longitude_{terminal}$	The longitude of the terminal station.	Float
$Label_{transfer}$	The unique identification label of transfer station.	Categorical
$City_{transfer}$	The unique identification label of transfer city.	Categorical
$Province_{start}$	The unique identification label of transfer province.	Categorical

Among the above features, the names of the start, transfer and terminal stations, as well as the cities and provinces in which these stations are located, are represented in string format. To enable the XGBoost model to process these labels correctly, the unique identification labels of the feature are mapped to integers.

3.2. XGBoost Model Training

3.2.1. Feature Selection and Processing

To train the XGBoost model, we use the unique identification label and coordinate feature of start and terminal stations, the unique identification label of the cities and provinces where the stations are located as input parameters, and the unique identification label of cities where the transfer stations are located as an output value.

The XGBoost algorithm incorporates a feature importance function that quantitatively assesses and compares the significance of various features. Feature importance score illustrates the computation of relevance scores for feature variables subsequent to the completion of data ingestion. This process follows the experimental determination of feature importance scores, where metrics such as weight, gain, and cover yield congruent results.

Figure 6 illustrates the scores assigned to each feature, thereby reflecting their respective impacts to the prediction of transfer cities. The optimal feature subset is determined based on the evaluation index of the XGBoost model. In this study, the parameters of the XGBoost model were initialized using the learner's default settings, wherein we sequentially select different numbers of features to train the XGBoost model and calculate the corresponding evaluation indexes. Table 3 below shows the evaluation indexes for different numbers of selected features.

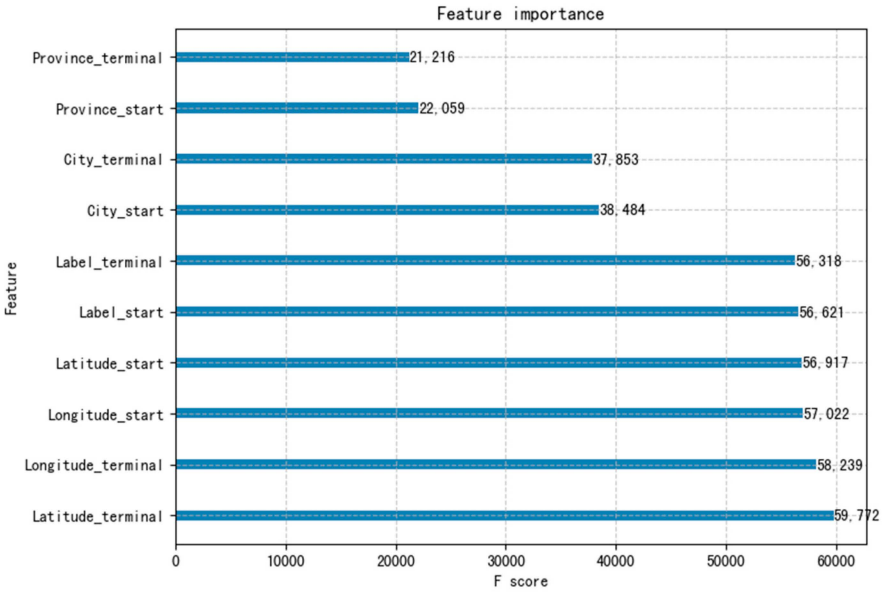

Figure 6. Feature importance score results.

Table 3. Evaluation indexes with different number of features.

Number of Features	Accuracy	Recall	Precision	F1-Score
1	0.3297	0.3297	0.3686	0.3061
2	0.6768	0.6768	0.6653	0.6664
3	0.7831	0.7831	0.7787	0.7767
4	0.8836	0.8836	0.8798	0.8796
5	0.8865	0.8865	0.8842	0.8827
6	0.8873	0.8873	0.8851	0.8837
7	0.8875	0.8875	0.8853	0.8838
8	0.8879	0.8879	0.8849	0.8842
9	0.8877	0.8877	0.8841	0.8841
10	0.8880	0.8880	0.8844	0.8843

Figure 7 illustrates that as the number of selected features increases, the evaluation indexes of the model exhibit significant improvement. However, when the number of selected features exceeds four, the improvement in the evaluation indexes becomes negligible. Therefore, we select the first four features, namely the latitude and longitude of the start and terminal stations, for subsequent model training.

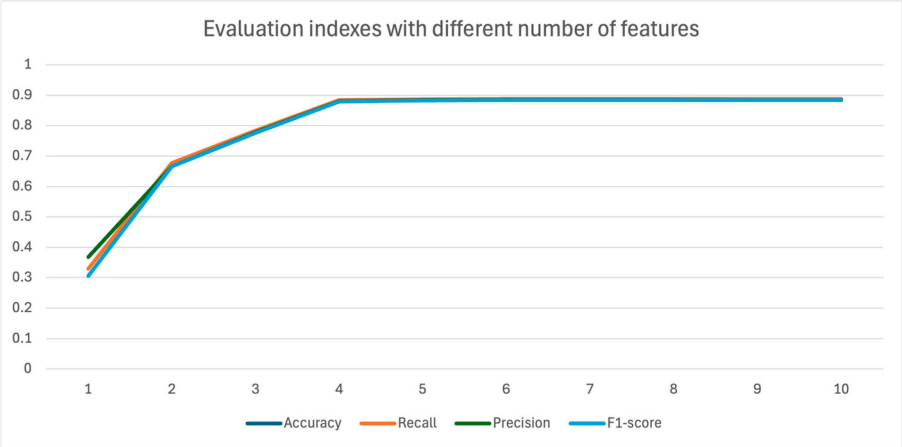

Figure 7. Evaluation indexes with different numbers of features.

3.2.2. Parameter Tuning

Parameter tuning of the model is an integral aspect of the modeling process; good parameter tuning can improve the performance of the model. The prior XGBoost model utilized the default parameters of the learner, as follows: $n\_estimaters = 150$, $max\_depth = 5$, $learning\_rate = 0.1$, $gamma = 0.1$. This section focuses on parameter tuning. Common approaches for tweaking model parameters include random parameter selection, grid search parameter selection and others.

After grid parameter adjustment of the XGBoost model, experimental results indicate that $n\_estimaters = 250$, $max\_depth = 7$, $learning\_rate = 0.1$ and $gamma = 0.1$ are the ideal parameter settings for the XGBoost model. It shows that a larger number of trees and a higher max_depth value can help to increase the accuracy level. Table 4 compares the XGBoost model before and after parameter adjustment.

Table 4. Comparison of evaluation indexes before and after parameter tuning of XGBoost model.

Evaluation Index	Before Parameter Adjustment	After Parameter Adjustment
Accuracy	0.8836	0.8903
Recall	0.8836	0.8904
Precision	0.8798	0.8877
F1-Score	0.8796	0.8876

3.2.3. Robust Validation

To assess the robustness of the trained XGBoost model, we conducted a k-fold cross-validation with k set to 5. The dataset was randomly divided into five equal folds, and in each iteration, one fold was used as the validation set while the remaining four folds were combined to form the training set.

The performance of the model was evaluated based on the accuracy metric. The cross-validation results are presented in Table 5, which shows the accuracy of the model on each fold as well as the mean and standard deviation of the accuracy across all folds.

Table 5. K-fold cross-validation results.

Fold	Accuracy
1	0.8848
2	0.8835
3	0.8846
4	0.8857
5	0.8855

As can be seen from the table, the model achieved a relatively high level of accuracy on each fold, with the accuracy ranging from 0.8835 to 0.8857. The mean accuracy across all folds was 0.8848 and the standard deviation was 0.00087. The result of the k-fold cross-validation indicates that the XGBoost model has good robustness.

3.3. XGBoost Acceleration Analysis

To evaluate the acceleration impact of the XGBoost model on the path planning algorithm, as well as to assess the quality of the path planning results, this study randomly selects 1000 OD pairs that are reachable in the air–rail intermodal network as samples. In the absence of XGBoost acceleration, the K shortest path planning algorithm would traverse all edges to achieve optimal results, which serves as a benchmark for comparison with the results using XGBoost. Path planning queries were conducted for these 1000 OD pairs under the following computational configurations:

1. Run K shortest path planning algorithm directly Without XGBoost.
2. Use XGB-HPPA.

The XGB-HPPA is written in Java and was compiled using JDK1.8. All experiments were executed on a 2.9 GHz Intel i7-10700 CPU PC with 16 cores and 16 GB of RAM running Windows 10. By conducting a detailed analysis of the network scale, calculation time and quality of path planning results, the XGBoost model's effect on the path planning process was analyzed.

3.3.1. Analysis of Algorithm Complexity and Scalability

The XGB-HPPA algorithm mainly consists of two major steps. The first step involves using the XGBoost model to predict transfer cities and setting the heuristic factor for the transfer edges. The second stop involves the K shortest path planning algorithm.

Since the training process of the XGBoost model is conducted offline, only the prediction calculation is utilized during user queries. Therefore, we focus solely on computational complexity in the prediction phase. The computational complexity in the prediction phase of XGBoost is relatively lower compared to the training phase. The prediction involves traversing each decision tree in the ensemble to obtain the predicted value for each sample. The time complexity for predicting a single sample is approximately O(K*log(n)), where K is the number of trees and n is the maximum depth of the trees. Since the number of trees (K) and depth of trees (n) are fixed, theoretically, the prediction time of XGBoost is not affected by changes in the scale of the network.

As for the K shortest path planning, the traditional Dijkstra algorithm has a time complexity of $O((V + E)\ log\ V)$ when using a priority queue, where V is the number of vertices and E is the number of edges in the graph. The time complexity of the bidirectional Dijkstra algorithm can be approximated as $O((V + E)\ log\ V/2)$, as the search is performed in both directions and the algorithm stops when the two searches meet. The actual performance improvement can vary depending on the specific graph structure and the distance between the start and goal nodes.

To analyze the effect of the XGBoost model on K shortest path planning, we collected statistics on the number of edge weights that need to be computed for path planning of 1000 OD pairs, both with and without XGBoost.

Figure 8 shows that, during the path planning calculation process for 1000 OD samples, the maximum number of edges that require weight computation is relatively large, and the distribution is concentrated in most cases. After acceleration with the XGBoost model, the maximum value has significantly decreased, the data distribution has become more concentrated, the box has noticeably shrunk, and the median has also decreased markedly.

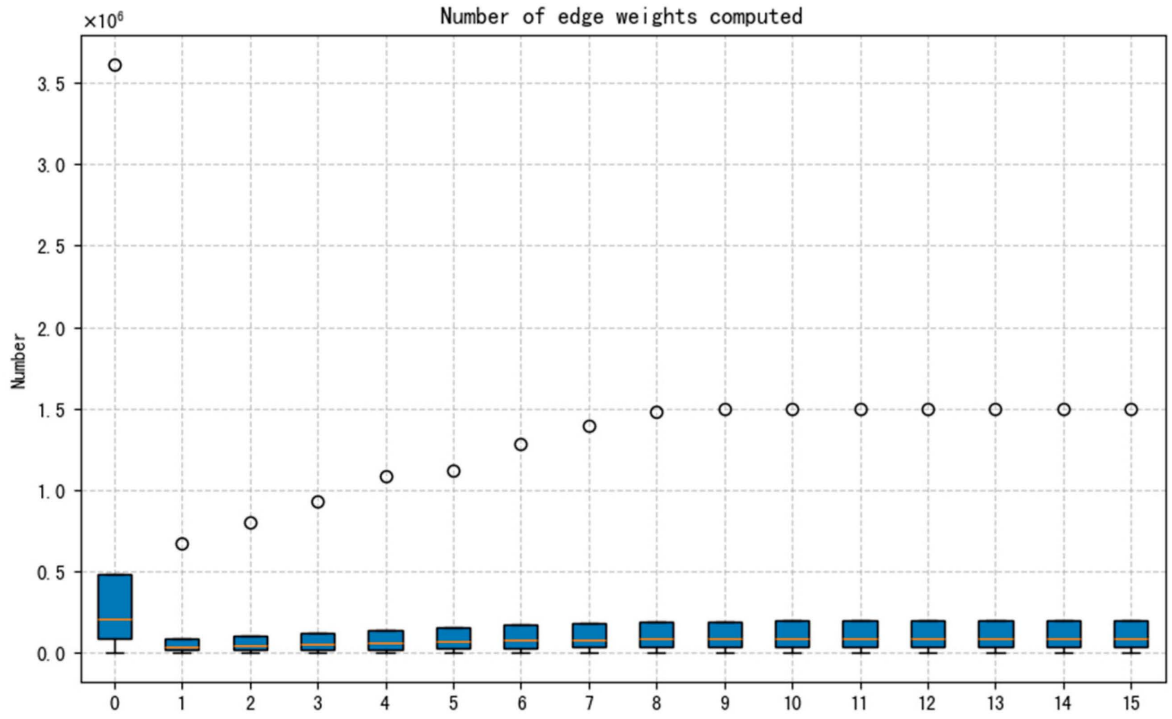

Figure 8. Number of edge weights computed.

In terms of algorithm scalability, the XGB-HPPA can independently handle each user query, thus it can be easily parallelized, with each search running on separate threads or processors. Moreover, further acceleration can be achieved by fine-tuning the XGBoost model.

3.3.2. Comparison of Calculation Time

We compared the average path planning calculation time, maximum calculation time and XGBoost calculation time for 1000 OD pairs, and the comparison results are as follows.

In Figure 9, the horizontal axis denotes the number of cities selected as transfer cities predicted by the XGBoost model, with 0 indicating no use of XGBoost. It can be observed that the average calculation time without using XGBoost is 673 ms. The implementation of XGBoost markedly diminishes this average time to 337 ms, which is approximately 50% of the original computation time. As the quantity of selected cities for transfers escalates, there is a corresponding rise in the average calculation time. Nevertheless, once the number of selected cities surpasses 10, the augmentation in average calculation time becomes negligible. This observation underscores the efficiency gains achieved through the strategic use of XGBoost in optimizing the path planning process, particularly when a moderate number of transfer cities are considered.

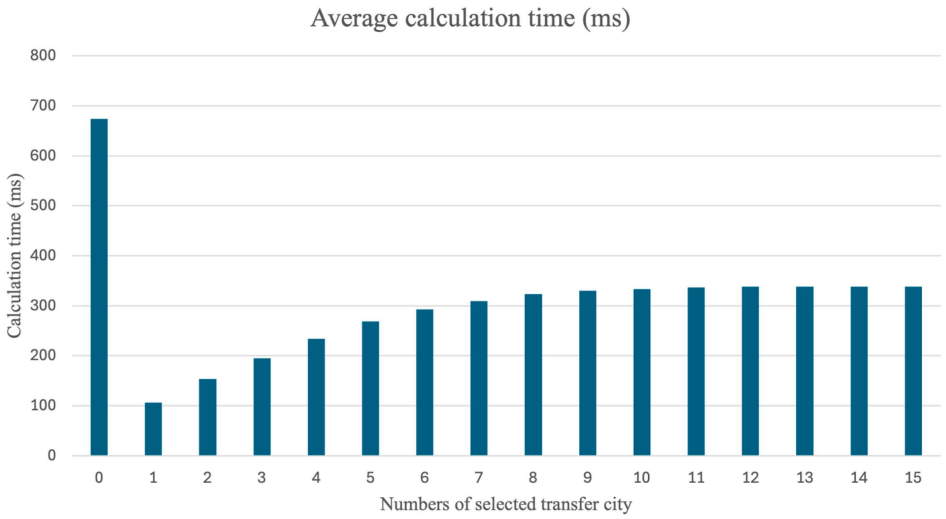

Figure 9. Average calculation time.

Figure 10 illustrates the maximum calculation time required among all 1000 OD pairs. It is evident that without the acceleration provided by XGBoost, certain OD queries may experience excessively prolonged path planning computation times, reaching up to 4772 ms. Upon the incorporation of XGBoost, this maximum calculation time is significantly reduced to 1657 ms, which is approximately 35% of the original computation time. In practical application settings, particularly when a website handles a high volume of concurrent queries, such lengthy query times could lead to network congestion and potentially hinder the website's ability to function properly. The acceleration facilitated by XGBoost notably diminishes the upper limit of query times, enhancing the overall efficiency and responsiveness of the system. Furthermore, it is observed that when the number of selected cities for transfers exceeds seven, the increase in maximum calculation time becomes insignificant, suggesting an optimal balance between the number of transfer options and computational efficiency.

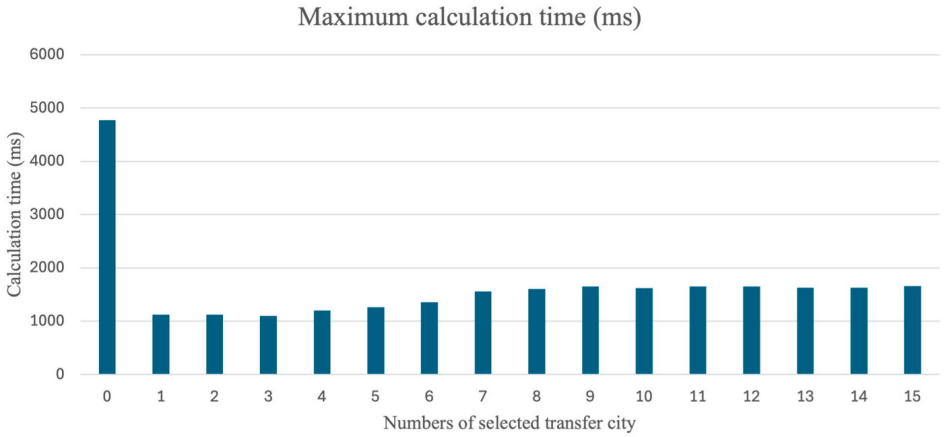

Figure 10. Maximum calculation time.

Figure 11 shows the maximum XGBoost prediction calculation time needed among all 1000 OD pairs. It is noteworthy that the time required to execute XGBoost predictions in the path planning calculations remains constant regardless of the number of cities selected for transfers. This observation underscores the efficiency of XGBoost in providing rapid predictions irrespective of the complexity introduced by the number of transfer cities.

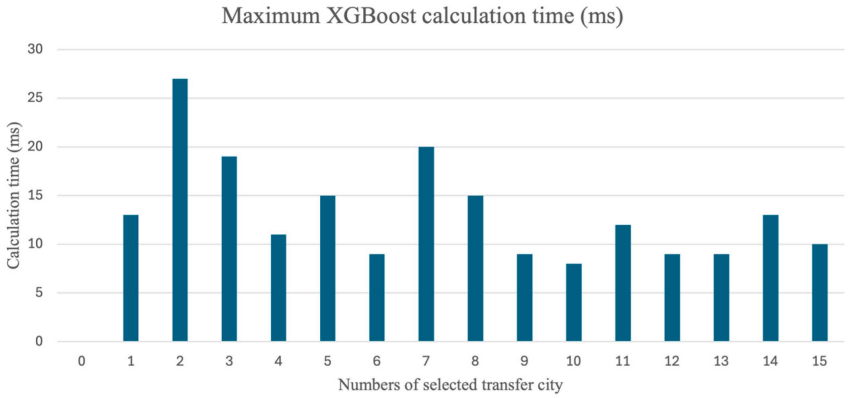

Figure 11. Maximum XGBoost calculation time.

The principle of acceleration based on XGBoost is to refine the original transportation network by excluding transfer nodes outside the model-recommended cities, thereby reducing the network scale and accelerating the speed of path planning calculations. The fewer selected cities, the smaller the refined network scale, and the faster the computation speed.

3.3.3. Comparison of Path Planning Results Quality

To assess the influence of XGBoost's network refinement on the quality of path planning outcomes, a comparative analysis was conducted. This analysis focused on following three aspects: distribution of travel time for optimal paths of OD samples, the average number of viable paths within the sample results and the proportion of resolvable OD pairs in relation to the original dataset. Through comparative analysis, insights were gleaned into how the refined network affects the diversity and feasibility of path planning options, as well as the algorithm's ability to provide solutions across a range of OD queries. This evaluation is crucial for understanding the trade-offs between computational efficiency and solution quality in the context of air–rail intermodal path planning.

Figure 12 indicates that when only one transfer city is selected using XGBoost, there is a noticeable increase in both the upper quartile and median of the travel time for the optimal paths of OD samples, suggesting a slight deterioration in the overall quality of the generated paths. However, when XGBoost selects two or more transfer cities, the distribution of the travel time for the optimal paths of OD samples is essentially the same as when the XGBoost model is not used.

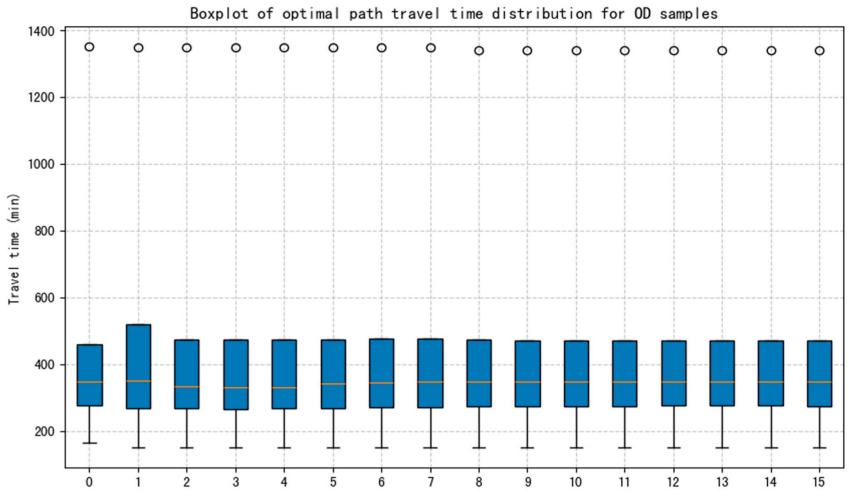

Figure 12. Boxplot of optimal path travel time distribution for OD samples.

Figure 13 shows the average number of available paths the algorithm can get when the number of OD pairs is 1000; it is evident that without XGBoost the algorithm can get about 10 paths. After using XGBoost, this average reduces to around eight paths. Furthermore, it is observed that as the number of selected cities for potential transfers decreases, so does the average number of available paths. Conversely, an increase in the number of selected cities leads to a corresponding increase in the number of viable paths. However, once the number of selected cities surpasses nine, the average number of available paths stabilizes, indicating that further increases in the number of cities do not significantly enhance the diversity of path planning options. This finding suggests an optimal balance between the number of transfer cities and the richness of the solutions, highlighting the importance of judiciously selecting the number of cities to include in the path planning strategy.

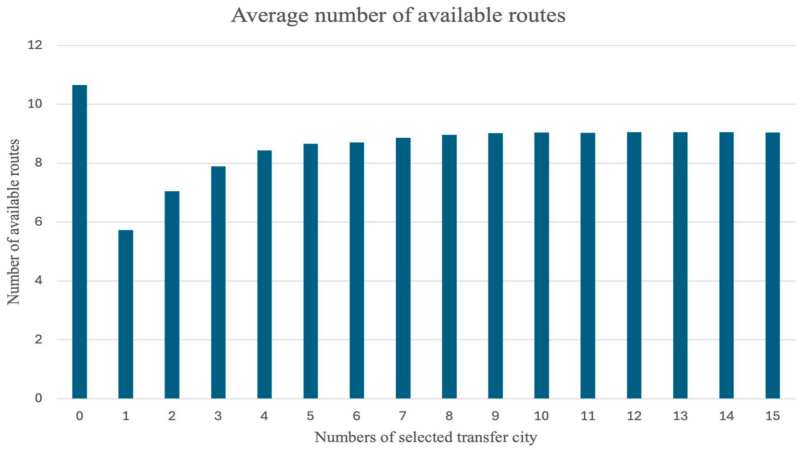

Figure 13. Average number of available routes.

Figure 14 demonstrates the impact of using XGBoost on the proportion of OD pairs that can still obtain optimal paths, it can be seen that as the number of selected cities increases, the proportion of solvable OD pairs also rises. When the number of selected cities exceeds nine, the proportion of solvable OD samples no longer increases significantly. When using XGBoost acceleration and selecting more than nine cities, approximately 98% of the samples can still obtain valid routes.

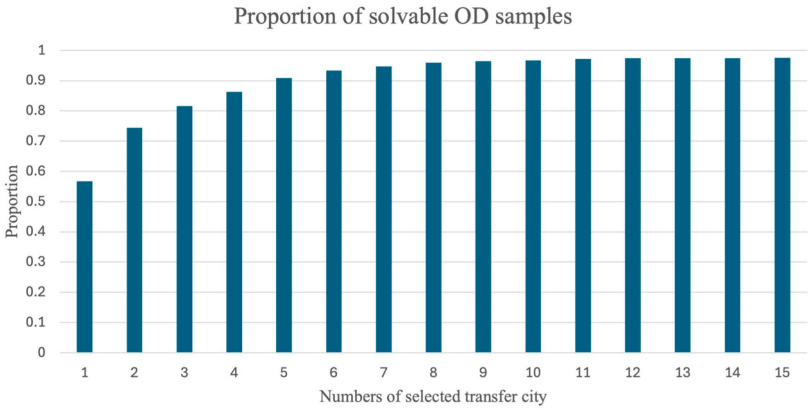

Figure 14. Proportion of solvable OD samples.

Taking the path planning results without using XGBoost as the optimal solution, it can be observed in Figure 15 that when merely one city is chosen for transfers, a mere 20% of the samples attain the optimal solution. As the number of selected cities for potential transfers grows, so does the percentage of samples that successfully reach the optimal solution. Once the number of selected cities surpasses ten, the proportion of samples achieving the optimal

solution does not increase significantly. At this juncture, approximately 83% of the samples are able to secure the optimal solution.

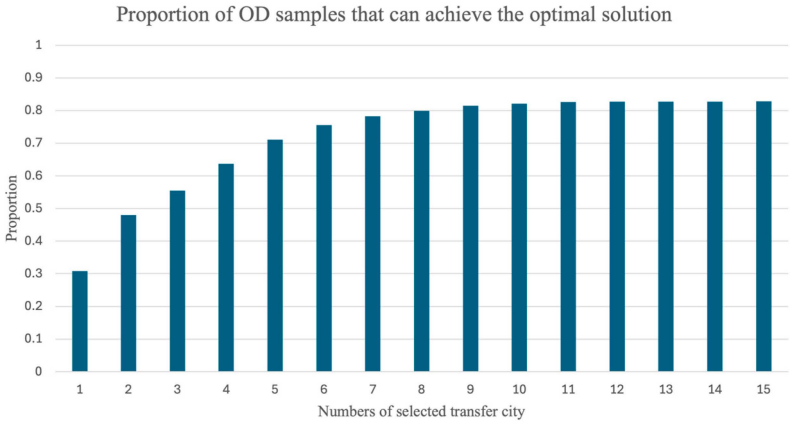

Figure 15. Proportion of OD samples that can achieve the optimal solution.

In conclusion, while XGBoost acceleration offers significant improvements in computational efficiency, it may introduce a trade-off with the quality of the results. The expansion of the number of selected cities for transfers can enhance the quality of the path planning outcomes up to a certain threshold. However, beyond this point, the increase in the number of cities also escalates the computational time required to resolve path planning queries. Thus, the selection of an optimal number of cities is pivotal in striking a balance between the speed of solution delivery and the preservation of result quality. Drawing from the empirical data presented, it is reasonable to infer that the selection of seven to ten cities provides a pragmatic approach, effectively harmonizing the dual objectives of swift computation and robust solution quality within the context of air–rail intermodal path planning.

4. Conclusions

This study developed a methodology to apply the XGBoost model to accelerate path planning algorithm. Real-world ticket purchase data and train and airplane schedule data are selected as the case study to examine the XGBoost model's acceleration rate and how the acceleration affect the result. According to the experimental results, the following conclusions can be drawn:

1. The coordinate features of starting and terminal stations are more influential than unique identification label features.
2. The XGBoost model can significantly enhance the computational speed of the path planning algorithm, reducing the average computation time by 50% in this paper's experiments, with the maximum time reduction reaching 31%.
3. The XGBoost model introduces a degree of compromise in path planning outcomes. This is observed through a modest decrease in the average number of viable paths for OD pairs. Additionally, there are instances where certain OD pairs may not readily reach the optimal routes. However, these effects are generally outweighed by the substantial gains in computational efficiency and the enhanced ability to manage complex path planning scenarios within acceptable time frames.
4. Selecting a greater number of cities for transfers using XGBoost can lead to improved results, although this also increases the time required for path planning. Optimal results are generally achieved when selecting between seven and ten cities for transfers.

Despite the significant improvement in computational efficiency demonstrated by the XGBoost model in accelerating path planning algorithms, its application is still subject

to certain limitations. Firstly, XGBoost requires meticulous parameter tuning to achieve optimal performance. The vast parameter space of XGBoost, including learning rate, tree depth, and subsampling ratio, necessitates sophisticated optimization techniques to avoid suboptimal configurations. This tuning process can be computationally expensive and time-consuming, especially for large datasets. Additionally, the interpretability of the XGBoost model is relatively weak, which can be a limitation in scenarios where model interpretability is required. Lastly, in the travel data of air–rail intermodal passengers, the passenger flow varies across different cities, which can easily lead to an imbalance in the sample data, XGBoost may not effectively handle minority class samples, which can lead to high accuracy for the majority class but poor performance for the minority class, thereby affecting overall prediction effectiveness.

Future research can explore several promising directions to address the limitations of the XGBoost model and enhance its performance. Combining XGBoost with other machine learning algorithms, such as deep learning models or ensemble methods, could leverage the strengths of multiple approaches. This could lead to more robust predictive models that outperform XGBoost alone in certain scenarios. Exploring ways to improve XGBoost's performance with imbalanced data, such as adjusting sample weights or using oversampling and undersampling techniques, can enhance the model's predictive capability for minority class samples, thereby improving overall prediction effectiveness.

Author Contributions: Conceptualization, S.W., N.Z. and X.S.; methodology, S.W. and G.B.; software, S.W. and J.W.; validation, X.S. and N.Z.; formal analysis, S.W., N.Z. and J.W.; investigation, S.W., N.Z. and G.B.; resources, X.S.; data curation, X.S. and S.W.; writing—original draft preparation, S.W. and G.B.; writing—review and editing, X.S.; visualization, S.W.; supervision, X.S. and N.Z.; project administration, X.S.; funding acquisition, S.W. and X.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China Academy of Railway Sciences Corporation Limited grant number 2023YJ132.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, K.; Jiang, C.; Ng, A.K.; Zhu, Z. Air and rail connectivity patterns of major city clusters in China. *Transp. Res. Part A Policy Pract.* **2020**, *139*, 35–53. [CrossRef]
2. Xu, T.; Ding, X.; Li, J. Research on the Connecting Path Search Algorithm for Air-Rail Integration. *J. Softw.* **2013**, *8*, 8. [CrossRef]
3. Xu, F.; Zhu, J.F.; Miao, J.J.; Ding, R.R. Simulation of Two Stages-Variant Growth Evolution Model of High-speed Railway and Civil Aviation Compound Network. *J. Syst. Simul.* **2018**, *30*, 1369–1375.
4. Xu, W.; He, S.; Song, R.; Chaudhry, S.S. Finding the K shortest paths in a schedule-based transit network. *Comput. Oper. Res.* **2012**, *39*, 1812–1826. [CrossRef]
5. Magzhan, K.; Jani, H.M. A review and evaluations of shortest path algorithms. *Int. J. Sci. Technol. Res* **2013**, *2*, 99–104.
6. Wang, S.X. The improved dijkstra's shortest path algorithm and its application. *Procedia Eng.* **2012**, *29*, 1186–1190.
7. Donald, G.; Hao, J.X.; Kai, S.R. Shortest path algorithms using dynamic breadth-first search. *Networks* **1991**, *21*, 29–50.
8. Aranski, A.W. Depth First Search Algorithm In Solving the Shortest Route Using the Concept of Generate and Test. *Int. J. Inf. Syst. Technol.* **2022**, *6*, 353–360.
9. Liu, L.; Wang, B.; Xu, H. Research on path-planning algorithm integrating optimization A-star algorithm and artificial potential field method. *Electronics* **2022**, *11*, 3600. [CrossRef]
10. Dirk, S.; Christian, T. Running time analysis of ant colony optimization for shortest path problems. *J. Discret. Algorithms* **2012**, *10*, 165–180.

11. Lin, L.; Gen, M. Priority-based genetic algorithm for shortest path routing problem in OSPF. In *Intelligent and Evolutionary Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 91–103.
12. Fu, L.; Sun, D.; Rilett, L.R. Heuristic shortest path algorithms for transportation applications: State of the art. *Comput. Oper. Res.* **2006**, *33*, 3324–3343. [CrossRef]
13. Xu, F.; Zhu, J.F.; Miao, J.J. Optimization Design of Air-rail Network Based on Algorithm of Traversal and Search. In 2016 International Conference on Service Science, Technology and Engineering (SSTE 2016). Available online: https://scholar.archive.org/work/zsoehcritvgnpfxfxdkjp3rvyi/access/wayback/http://dpi-proceedings.com/index.php/dtetr/article/download/6493/6089 (accessed on 10 November 2024).
14. Zheng, L.X.; Si, B.F. Hyper-graph based stochastic multimodal traffic network assignment problem. *Shandong Sci.* **2013**, *26*, 78–83.
15. Gendreau, M.; Ghiani, G.; Guerriero, E. Time-dependent routing problems: A review. *Comput. Oper. Res.* **2015**, *64*, 189–197. [CrossRef]
16. Nurhidayat, A.Y.; Widyastuti, H.; Sutikno; Upahita, D.P. Research on passengers' preferences and impact of high-speed rail on air transport demand. *Sustainability* **2023**, *15*, 3060. [CrossRef]
17. Fischer, F.; Helmberg, C. Dynamic graph generation for the shortest path problem in time expanded networks. *Math. Program.* **2014**, *143*, 257–297. [CrossRef]
18. Geisberger, R.; Sanders, P.; Schultes, D.; Vetter, C. Exact routing in large road networks using contraction hierarchies. *Transp. Sci.* **2012**, *46*, 388–404. [CrossRef]
19. Shi, H.; Cao, W.; Zhu, S.; Zhu, B. Applications of the Improved A* Algorithm for Route Planning. In Proceedings of the Second International Conference on Intelligent Computation Technology & Automation. IEEE Computer Society, Changsha, China, 10–11 October 2009.
20. Chen, T. Xgboost: Extreme Gradient Boosting, R Package Version 0.4-2. 2015; Volume 1, p. 464. 2015. Available online: https://cran.ms.unimelb.edu.au/web/packages/xgboost/vignettes/xgboost.pdf (accessed on 23 December 2024).
21. Chen, Z.; Fan, W. A freeway travel time prediction method based on an XGBoost model. *Sustainability* **2021**, *13*, 8577. [CrossRef]
22. Ren, Q.; Wang, J. Research on Enterprise Digital-Level Classification Based on XGBoost Model. *Sustainability* **2023**, *15*, 2699. [CrossRef]