applied sciences

MDPI

Article

Personalized Tour Itinerary Recommendation Algorithm Based on Tourist Comprehensive Satisfaction

Dingming Liu [1,*], Lizheng Wang [1], Yanling Zhong [1,*], Yi Dong [2,3] and Jinling Kong [1]

[1] School of Geological Engineering and Geomatics, Chang'an University, Xi'an 710054, China
[2] Aerial Photogrammetry and Remote Sensing Group Co., Ltd., Xi'an 710199, China
[3] Shaanxi Engineering Research Center of Geospatial Information, Xi'an 710199, China
* Correspondence: dmliu0401@gmail.com (D.L.); 2019026021@chd.edu.cn (Y.Z.)

Abstract: Personalized travel itinerary recommendation algorithms are the focus of research in smart tourism and tourism GIS. Aiming to address issues present in travel itinerary recommendations for the increasingly popular "self-drive tour" mode, this study proposes an algorithm based on comprehensive tourist satisfaction to mitigate problems such as the neglect of important relevant factors and low degree of personalization. First, we construct a model of tourist satisfaction for travel itineraries by comprehensively considering factors including time utilization, the attractiveness of attractions, itinerary feasibility, and the diversity of attraction types. Unlike previous studies, we consider dining and accommodation time during the itinerary, the physical condition of tourists, and the diversity of attraction types, and establish penalty functions to flexibly constrain deviations from the expected conditions in itinerary planning. Then, with the optimization of comprehensive tourist satisfaction as the objective, we design a new algorithm to address the itinerary recommendation problem, supporting tourists in selecting must-visit attractions, restaurants, and hotels, as well as personalized preferences such as the sightseeing sequence. The experimental results demonstrate that our proposed algorithm outperforms two baseline algorithms, providing higher comprehensive tourist satisfaction while also exhibiting greater feasibility in itinerary planning. The proposed algorithm effectively addresses the issue of personalized travel itinerary recommendation, presenting an efficient, feasible, and practical solution.

Keywords: tour itinerary recommendation; trip planning; personalization; approximation algorithm; combinatorial optimization problem

Citation: Liu, D.; Wang, L.; Zhong, Y.; Dong, Y.; Kong, J. Personalized Tour Itinerary Recommendation Algorithm Based on Tourist Comprehensive Satisfaction. *Appl. Sci.* **2024**, *14*, 5195. https://doi.org/10.3390/app14125195

Received: 1 May 2024
Revised: 3 June 2024
Accepted: 7 June 2024
Published: 14 June 2024

1. Introduction

In recent years, the "self-drive tour" tourism model has become increasingly popular. Simultaneously, under the concept of holistic tourism development [1], some regions are being developed as tourism destinations to facilitate tourists' visits to regional scenic spots through connected highways. This further promotes the development of self-drive tours. However, customizing personalized travel schedules remains a complicated and time-consuming problem for tourists. With the explosive growth of network information resources, problems such as information overload and resource misorientation have become increasingly serious [2]. Tourists need to spend a lot of time and energy searching for tourism information that truly meets their needs. Furthermore, while fixed tourist route recommendations cannot meet personalized demands, recommendations focusing solely on individual attractions fail to consider the overall travel experience. Tour itinerary recommendation has thus become a research hotspot in smart tourism and tourism GIS; both academia and industries have been constantly pursuing recommendation systems and optimization models that can accurately select attractions as well as create and recommend useful travel itineraries.

At present, there are several problems in the literature. In terms of factors considered, the relevant studies lack a comprehensive temporal dimension as they fail to consider the

scheduling of dining and accommodation time. This oversight could result in deviations from travelers' original plans during their journeys. Additionally, the recommended results often lack consideration for the diversity of scenic categories, which may lead to monotony and cause aesthetic fatigue during the journey. Furthermore, the majority of studies have not addressed the physical condition of travelers during their itinerary, potentially leading to fatigue during the journey.

At the algorithmic level, the related studies have not elucidated how to handle multiple time windows for nodes, which is crucial for multi-day itinerary planning and the scheduling of specific visit times for tourists. Moreover, personalization is generally weak, as most studies do not support the inclusion of selected tourist attractions, restaurants, and hotels or specific node visit sequences in recommendation requirements. Additionally, in most studies, recommendations for scenic spots primarily consider the intrinsic factors of the spots and user interests but fail to comprehensively assess their value within the broader context of user-defined factors and other potential scenic spots. This lack of comprehensive consideration hinders the generation of optimal itineraries.

To illustrate these issues, let us assume that John plans a three-day self-drive tour. He has arranged the departure and return times, as well as the starting and ending points of the trip. Additionally, he wishes to maintain a regular routine during the trip, including consistent meal and rest times. He plans to have lunch at Restaurant A on the second afternoon and has booked Hotel A for the first night. He hopes to visit Scenic Spot A in the morning and Scenic Spot C after that, while also visiting other attractions to meet the planned travel time. He hopes this trip will include some local historical and cultural attractions as well as rural scenic spots. Furthermore, he wants to avoid the fatigue caused by visiting multiple mountain attractions in one day and stay within a certain budget for the entrance fees of the attractions. This example illustrates a complex itinerary planning demand, highlighting the limitations of traditional recommendation systems in handling complex personalized needs.

This paper addresses the aforementioned issues by proposing a personalized tour itinerary recommendation algorithm based on tourists' comprehensive satisfaction, specifically tailored for the self-drive tourism mode. The algorithm is characterized by a comprehensive consideration of factors and rich personalization settings, while also balancing efficiency and optimization performance. The main contributions of this study can be summarized as follows.

Firstly, we constructed a comprehensive satisfaction score for the itinerary and formulated a problem model with this indicator as the objective function. We introduced a wide range of factors into the problem, including the type, grade, popularity, opening hours, recommended visiting time, accommodation and dining availability, visitor interest types, meal and rest time preferences, physical limitations, budget constraints, planned travel time, planned starting and ending points, other personalized settings, and the time cost of the itinerary. Based on these factors, we constructed indicators for the time utilization rate, attractiveness, feasibility, and diversity of attraction types to form the comprehensive satisfaction score for tourists. Additionally, in our model, some constraint conditions are flexible, and we used penalty functions to evaluate and constrain deviations between the itinerary plan and the expected conditions, making the itinerary planning more flexible and personalized.

Secondly, we proposed a novel algorithm based on a variant of the Monte Carlo Tree Search (MCTS) algorithm and the Lin–Kernighan Heuristic (LKH) algorithm to solve the proposed problem. This algorithm fully utilizes the search capabilities of the MCTS algorithm in large search spaces and the optimization capabilities of the LKH algorithm in the Traveling Salesman Problem, efficiently and effectively addressing the itinerary recommendation problem by decomposing it into local subproblems. In the algorithm, we designed a time planning algorithm based on the minimum penalty principle and redesigned some functions in the MCTS and LKH algorithms. Furthermore, we improved the algorithm's efficiency through the use of hash tables. The proposed algorithm supports nodes with

multiple time windows, allowing tourists to select must-visit attractions, restaurants, and hotels and choose specific node visitation sequences and other personalized preferences. The experimental results using real data demonstrate that the proposed algorithm can provide itinerary recommendations with higher comprehensive satisfaction and greater feasibility in itinerary planning.

Thirdly, we proposed a system framework outlining the service approach of the algorithm and the supported personalized settings. In the designed system, tourists can input their requirements and preferences through basic settings and extensive optional settings and can then obtain detailed references for their travel itineraries in terms of both time and route.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the problem definition, proposed system framework, and constructed mathematical model. Section 4 presents the proposed algorithm. Section 5 describes the conducted experiments and analyzes the results. Finally, Section 6 concludes the work and proposes future directions for improvement.

2. Literature Review

In this section, we review existing research relevant to our study from two perspectives: tourism itinerary recommendation and tourism itinerary planning. Itinerary planning refers to the process of planning when a specific set of locations is determined, which is typically considered the basis for itinerary recommendation. In recent years, itinerary planning and recommendation have attracted widespread attention from fields such as operations research, computer science and applications, graph theory, and mathematics.

2.1. Research on Itinerary Recommendation Algorithms

In the realm of trip recommendation, a significant amount of research has been conducted focusing on orientation problems, combinatorial optimization problems, or similar paradigms. The primary aim across these studies is to maximize global rewards within user-defined time budgets [3]. Sajal Halder [4], for example, leveraged a variant of the Monte Carlo Tree Search to devise an amusement park itinerary recommendation method, which takes into account factors such as the popularity of scenic spots, user interest levels, and queuing times. Similarly, Chen Jing [5] analyzed the impact of weather factors on people's travel and proposed a model of Dynamic Personalized POI Sequence Recommendation with fine-grained contexts, and solved it based on the MCTS algorithm. Kendall Taylor [6] formulated an integer linear programming model for trip recommendations, addressing selected scenic spots and solving it as an integer linear programming problem. Furthermore, Narisara Khamsing [7] devised an itinerary recommendation method for family travel, utilizing an improved large-domain adaptive search algorithm to maximize overall family satisfaction under constraints. Joy Lal Sarkar [8] likewise considered group-oriented tourism route recommendations, aiming to satisfy the preferences of each member in the group as much as possible.

Fu Chengyao [9] integrated factors like scenic spot popularity, distances, and user demand constraints to derive recommended itineraries by sequentially adding optimal scenic spots. Zhang Chenyi [10] proposed a method based on a state expansion approach to solve recommended itineraries, factoring in user interests, travel time constraints, and scenic spot time windows. In addition, Rizwan Abbas [11] introduced a route recommendation method supplemented with random recommendations based on neural network algorithms, aiming to inject novelty into scenic spot recommendations. Chen Lei [12] devised an itinerary recommendation approach grounded in graph representation learning and reinforcement learning, taking into account scenic spot popularity and user interests. Zhou Xiao [13] developed an intelligent travel route planning algorithm that employs an iterative search of clustering central motivation. In this method, clustering algorithms are employed to mine tourist attraction data based on tourists' interests, thereby obtaining optimal scenic spots and recommended itineraries. Additionally, Zhou Xiao [14] proposed

a travel recommendation algorithm based on text mining and the MP neural cell model of multiple transportation modes, achieving recommended itineraries through optimal tourism landscape matching algorithms and tourism route chain algorithms.

Zhang Yanmei [15] presented a route planning method that comprehensively considered factors such as station distances, initial travel locations, departure times, travel durations, costs, station scores, and popularity, ultimately identifying optimal routes via genetic algorithms. Phatpicha Yochum [16] introduced an adaptive genetic algorithm for personalized travel itinerary planning, incorporating attractions near starting and destination points as well as potential tourist preference factors into multi-objective optimization frameworks. Etsushi Saeki [17] proposed a multi-objective travel itinerary planning method based on ant colony algorithms, aiming to optimize travel costs and user satisfaction. A distinctive aspect of this approach involves integrating tourists' past travel records to adjust pheromones, thereby reflecting their general preferences. Meanwhile, Ding Yi [18] addressed travel itinerary planning under time constraints and uncertain traffic conditions, using a two-stage stochastic optimization model with chance constraints and the sample average approximation (SAA) method. Additionally, Ines Gasmi [19] proposed the use of multi-objective evolutionary algorithms to balance the popularity of POIs and user interest in itinerary recommendations. By studying and comparing several multi-objective evolutionary algorithms from multiple perspectives, it was demonstrated that NSGA-II is the most effective in generating personalized itinerary recommendation rules for tourists unfamiliar with a city.

2.2. Research on Itinerary Planning Algorithms

In terms of itinerary planning, Zhou Xiao [20] proposed a tourism route planning method that comprehensively considers distance, the number of transfers between public transportation systems, and average road conditions based on a dynamic programming algorithm. Xu Ke [21] proposed a tourism route planning algorithm based on a reinforcement learning algorithm to avoid areas with frequent traffic accidents and traffic congestion. Huang Zebin [22] constrained tourist travel time by introducing time window coefficients into the ant colony algorithm heuristic function. Lu Baichuan [23] comprehensively analyzed factors such as travel distance, costs, time, operating hours, and fees of tourist attractions to propose a city suburban self-driving tour route planning method focused on enhancing the overall tourist experience. Wu Xiongbin [24] proposed a tourism experience utility objective function comprising two parts: tourism activity utility and tourism travel utility, considering various optional transportation modes and solving it based on the ant colony algorithm. Additionally, some works focus on recommending the top-K or next POI (point of interest), mainly based on collaborative filtering (CF) or matrix factorization methods, recommending the top-ranked or best tourist attractions to tourists. Heitor Werneck [25] provided an overview of this aspect of work.

The above studies have considered multiple factors for tourist itinerary planning and recommendation, implementing various algorithms. Overall, existing methods contribute to enhancing tourists' travel experiences. However, tourists can further benefit from a more sophisticated model and algorithms with higher levels of personalization. This paper proposes a tourist itinerary recommendation algorithm that considers a wide range of factors and supports various personalized needs.

3. Problem Definition and Modeling

In this section, we present an overview of the travel itinerary recommendation problem addressed in our study and introduce the service mode of our algorithm. Subsequently, in Section 3.2, we mathematically model tourist comprehensive satisfaction, while Section 3.3 establishes the mathematical model for the travel itinerary recommendation problem with the objective of maximizing tourist comprehensive satisfaction.

3.1. Problem Definition

We define the tourism itinerary recommendation problem as follows: considering practical circumstances and expected demands in travel arrangements, we provide itinerary planning containing recommended attractions. The primary objective is to maximize the popularity and user interest in attractions, as well as the duration of attraction visits, while minimizing the deviation between estimated actual circumstances and expected conditions. In the next section, we formalize these factors and integrate them into a comprehensive satisfaction score. Additionally, the problem definition includes supporting nodes with multiple time windows, as well as various personalized settings.

This study focuses on formulating comprehensive rules by considering multiple factors and designing optimization methods to solve the complex mathematical problems proposed. It aims to provide itinerary planning references for tourists by solving the recommended scenic spots, the visiting order of nodes, and the time planning of trips, combined with path planning data between two points. The proposed algorithm can be applied independently or integrated as a complementary rule component with data-driven methods to enhance their flexibility and applicability.

The system framework illustrated in Figure 1 elucidates the service implementation approach of the algorithm. Tourists set their preferences and requirements through an interactive interface based on provided tourism information and their own needs. The system middleware converts the data into a format supported by the algorithm, and the resulting itinerary recommendations are presented to tourists via the algorithm. Figure 2 demonstrates the style of our results, where route information is derived from the sequence of node visits and corresponding route planning strategies, and time planning includes key time points during the journey and arrangements for dining and accommodation, providing tourists with time references closest to their expected conditions. Tourists can modify their requirements based on the results, and the system continuously optimizes the results based on user feedback.

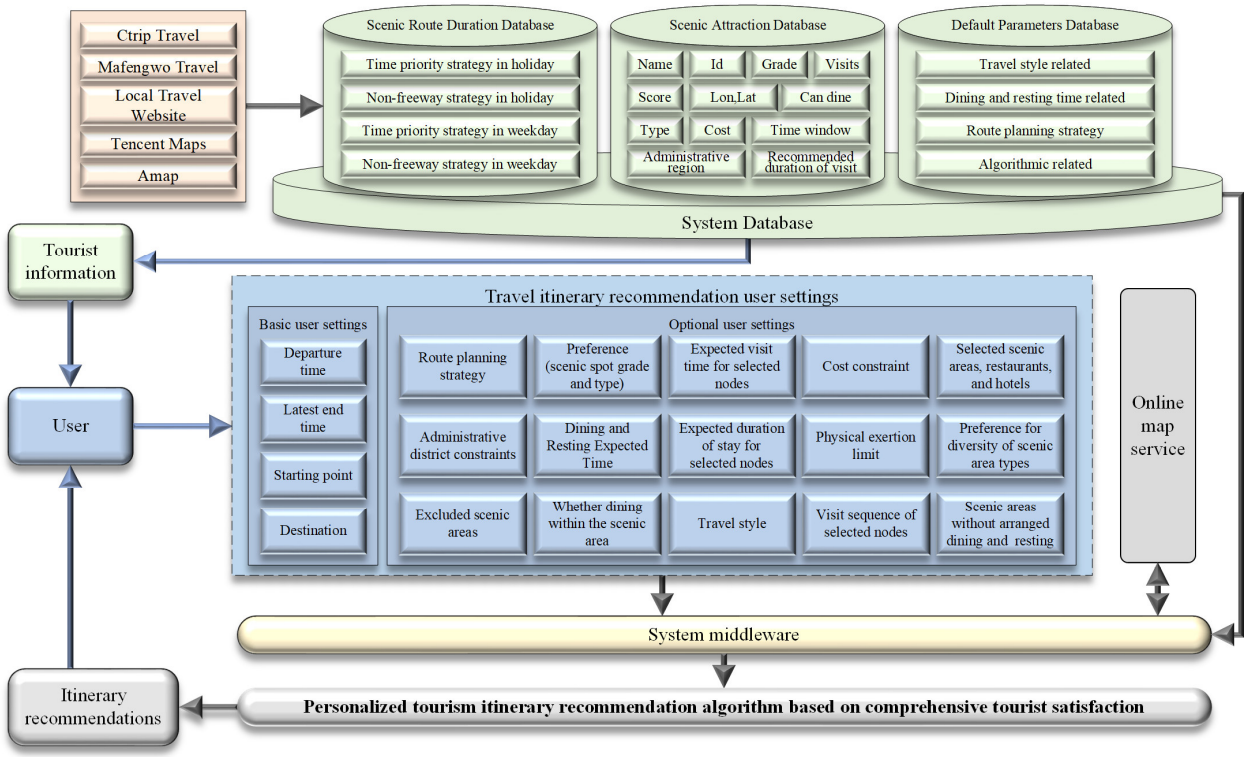
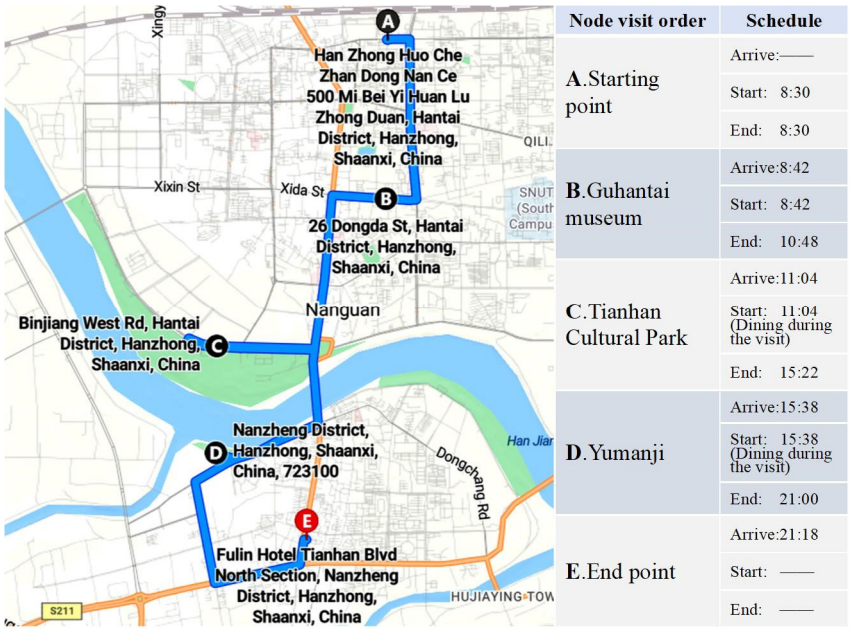


Figure 1. Proposed system framework.



Node visit order	Schedule
A.Starting point	Arrive:——
	Start: 8:30
	End: 8:30
B.Guhantai museum	Arrive:8:42
	Start: 8:42
	End: 10:48
C.Tianhan Cultural Park	Arrive:11:04
	Start: 11:04 (Dining during the visit)
	End: 15:22
D.Yumanji	Arrive:15:38
	Start: 15:38 (Dining during the visit)
	End: 21:00
E.End point	Arrive:21:18
	Start: ——
	End: ——

Figure 2. Result example.

The system database stores attraction attribute data extracted from tourism websites and map services, as well as travel time data between attractions under different time and route planning strategies, providing data support for tourism information display and algorithm operation. The system middleware integrates user settings with database data, obtaining travel time data for nodes outside the database by calling path planning services. It also converts user settings and database data into the format required by the algorithm through basic data calculations, including node time window calculations, user interest level calculations for nodes, and input data validation.

In addition to essential planned travel time and journey start and end points, the designed system offers a range of optional settings. Tourists can make selections based on their needs and preferences, as detailed in Figure 2. These personalized settings are achieved in two aspects. On the one hand, they are implemented through corresponding processing methods designed within the algorithm, such as selected attractions, hotels, and restaurants; the consideration of diversity in attraction types; specified node visit sequences; dining within attractions; and the exclusion of candidate attractions. On the other hand, these personalized settings benefit from the introduction of a wide range of factors in the problem; modifications to default parameters enable the fulfillment of corresponding personalized requirements, such as path planning strategies, travel modes, dining and sleeping times, expected stay and visit times for nodes (achieved by modifying node time windows), types of attractions of interest, budget constraints for attraction visits, regional restrictions for recommended attractions, and excluded candidate attractions. These data modifications can be completed within the middleware.

Appendix A introduces the basic concepts involved in this research, including nodes, meal and rest accommodation time parameters, node time windows, the node time period, scenic spot physical exertion, the scenic spot interest level, travel style, expectation, and penalty. The significance, specific forms, and quantification methods of these concepts are explained to provide readers with a clear understanding of the key terms and parameters used in the research. For detailed information, please refer to Appendix A.

3.2. Comprehensive Satisfaction of Tour Itinerary

To recommend a tour itinerary that meets the needs of tourists, we consider multiple factors. This section describes how to integrate these factors to construct a comprehensive satisfaction evaluation method for tour itinerary planning.

3.2.1. Scoring Based on the Time Utilization Ratio

People typically prefer to allocate more of their limited travel time to visiting scenic spots rather than spending it on travel or waiting. We define the Time Utilization Score (TUS) to evaluate the proportion of actual time spent on visiting scenic spots in the tour itinerary I. It is described using Equation (1):

$$TUS(I) = \frac{\sum_{a_i \in I} AVDur_{a_i}}{AvailableTime} \tag{1}$$

where $AVDur_{a_i}$ represents the actual visit duration of scenic spot a_i in the itinerary plan. The duration of visiting each scenic spot will not exceed the expected time. $AvailableTime$ represents the available time, which equals the planned journey duration minus the total time for meals and rest expected during the journey.

3.2.2. Scoring Based on the Attractiveness of Recommended Scenic Spots

Furthermore, people typically prefer to visit scenic spots that are more popular and interesting. Firstly, we define the scenic spot attractiveness score (SAS) of scenic spots a_i as follows:

$$SAS_{a_i} = Pop_{a_i} * Int_{a_i} \tag{2}$$

where Pop_{a_i} represents the popularity of scenic spot a_i and Int_{a_i} represents the level of interest tourists have in scenic spot a_i. We then express the scenic spot attractiveness score of the itinerary ($ISAS$) based on the average SAS of the scenic spots recommended in itinerary I, as defined using Equation (3):

$$ISAS(I) = \frac{\sum_{a_i \in I, a_i \in R} SAS_{a_i}}{RecNum} \tag{3}$$

where R represents the set of recommended scenic spots, and $RecNum$ represents their total number.

3.2.3. Scoring Based on Itinerary Feasibility

The itinerary penalty can quantitatively evaluate the deviation between the planned itinerary and the expected situation. Please refer to Appendix B for its calculation method. Because the penalty function is time-dependent, we define the itinerary feasibility score (FS) to assess the relative impact of this deviation on tour itinerary I, described as follows:

$$FS(I) = 1 - \frac{P(I)}{ItinDur} \tag{4}$$

where $ItinDur$ represents the duration of the tour itinerary.

3.2.4. Reward Based on Scenic Category Diversity

The monotonous categories of scenic spots in an itinerary may lead to tourist aesthetic fatigue. Considering that tourists may have preferences for the categories of scenic spots included in their itinerary, we have designed two types of diversity rewards (R_{div}) for scenic categories. They are described as follows:

$$R_{div1}(I) = |IS_{cat} \cap Cat(I)| * \gamma_1 + (|IS_{cat}| - |IS_{cat} \cap Cat(I)|) * \gamma_2 \tag{5}$$

$$R_{div2}(I) = R_{div1} + (|Cat(I)| - |IS_{cat} \cap Cat(I)|) * \gamma_3 \tag{6}$$

where IS_{cat} represents the set of user interest scenic spot categories and $Cat(I)$ represents the set of scenic spot categories included in the itinerary. The first diversity reward R_{div1} consists of two parts: a bonus for including the interest categories of scenic spots and a negative reward for missing interest categories in the itinerary, applicable when the user has selected interest categories. The second diversity reward R_{div2} consists of three parts:

it builds upon R_{div1} by adding a bonus for the diversity of categories beyond the selected interest categories, and it applies even when the user has not selected interest categories. In this paper, $\gamma_1 = 1, \gamma_2 = -1$ and $\gamma_3 = 0.5$.

3.2.5. Comprehensive Satisfaction Score

After considering the aforementioned factors, we define the comprehensive satisfaction score (CSS) for the tour itinerary, represented by the following equation:

$$CSS(I) = TUS(I) * ISAS(I) * FS(I) + R_{div1}(I) * \alpha_1 + R_{div2}(I) * \alpha_2 \quad (7)$$

Comprehensive satisfaction considers the time utilization rate, scenic spot attractiveness, itinerary feasibility, and diversity of scenic categories. Considering that not every user has needs related to the categories of scenic spots, we add the diversity reward of the scenic spot categories as additional terms to the CSS. Decision variables α_1 and α_2 are initially set to 0 by default but can be set to 0 or 1 based on user selection, but both cannot be set to 1 simultaneously.

3.3. Modeling of the Tour Itinerary Recommendation Problem Based on Comprehensive Satisfaction

We employ a mathematical model to represent the actual tour itinerary recommendation problem, as indicated in (8)–(13). Equation (8) represents the objective function, where our aim is to find a tour itinerary with routes and time arrangements that maximize the comprehensive satisfaction score. Simultaneously, it adheres to the constraints represented by (9)–(13), where $x_{ij} = 1$ when the itinerary includes the visit from a_i to a_j in order; otherwise, $x_{ij} = 0$.

$$Max\ CSS(I) \quad (8)$$

$$\sum_{j \in I} x_{ij} = 1 \quad (9)$$

$$\sum_{i \in I} x_{ij} = 1 \quad (10)$$

$$x_{ij} \times x_{ji} = 0 \quad (11)$$

$$DepartureT + ItinDur \leq LatestEndT \quad (12)$$

$$\sum_{a_i \in I} Cost_{a_i} \leq CostConstraint \quad (13)$$

Equations (9)–(11) signify that each node is visited exactly once, and the itinerary comprises no fewer than two nodes. Equation (12) ensures that the ending time of the itinerary does not exceed the expected ending time, while Equation (13) guarantees that the total cost of visiting scenic spots during the itinerary does not surpass the specified cost constraint. Additionally, the itinerary must start at the starting point and end at the destination, while adhering to the user's personalized requirements.

Our tour itinerary recommendation problem is modeled on a variant of the combinatorial optimization problem, which has been proven to be NP-hard [26,27]. Moreover, this problem involves penalty functions related to time, further complicating its complexity. Due to its NP-hard nature, finding an optimal solution is infeasible. To address these challenges, we propose the MCTSLKH_TR algorithm, which is based on the LKH algorithm and a variant of the MCTS algorithm. These algorithms heuristically search for optimal solutions within a more promising solution space, ensuring real-time performance and effectiveness. In Section 3, we will present the implementation of this algorithm.

4. Method

This section describes our proposed algorithm. Essentially, the tour itinerary recommendation problem can be boiled down to solving two key issues: recommending which scenic spots to visit (determining the node combination of the tour itinerary) and how to

plan the itinerary (including routes and timing). These two factors collectively influence the quality of the tour itinerary recommendation results. In the objective function, node combination directly determines $ISAS$ and R_{div} and sets upper limits for TUS and FS, while itinerary planning directly determines TUS and FS. Therefore, the main idea of the recommendation algorithm is to explore different node combinations and optimize the objective function of the itinerary composed of these node combinations through itinerary planning, aiming to return the optimal result among all attempts. During each iteration, the selection of candidate attractions is influenced by their performance in participating in the itinerary. We implemented this main process based on a variant of the MCTS algorithm.

As a crucial step in the recommendation algorithm, tour itinerary planning aims to find the optimal itinerary for the given set of nodes by maximizing $TUS * FS$. We define $TUS * FS$ as the tour itinerary planning satisfaction score ($TPSS$). The tour itinerary planning problem also involves two key issues: route planning (represented as the order of node visits in the algorithm) and time planning. Both factors collectively affect the quality of the itinerary planning results. The order of node visits determines the upper limit of the $TPSS$, while the time scheduling on this node visit order determines the specific $TPSS$. Therefore, our main approach to itinerary planning is to heuristically explore different node visit sequences and optimize the TPSS under the current sequence through timing planning, aiming to return the optimal itinerary arrangement among all attempts. Itinerary planning is modeled based on the Traveling Salesman Problem, which was implemented using the LKH algorithm.

As a key step in itinerary planning, timing planning comprehensively considers the time costs of food, lodging, transportation, and sightseeing for the determined node combination and visit sequence. It primarily maximizes the $TPSS$ through the selection of node time windows and the arrangement of food and lodging. We implemented this using a local optimal algorithm. Figure 3 illustrates the hierarchical relationship of the recommendation algorithm, itinerary planning algorithm, and timing planning algorithm, as well as the main processes of the algorithms. Next, we will provide detailed descriptions of the timing planning, itinerary planning modules, and overall algorithm implementation.

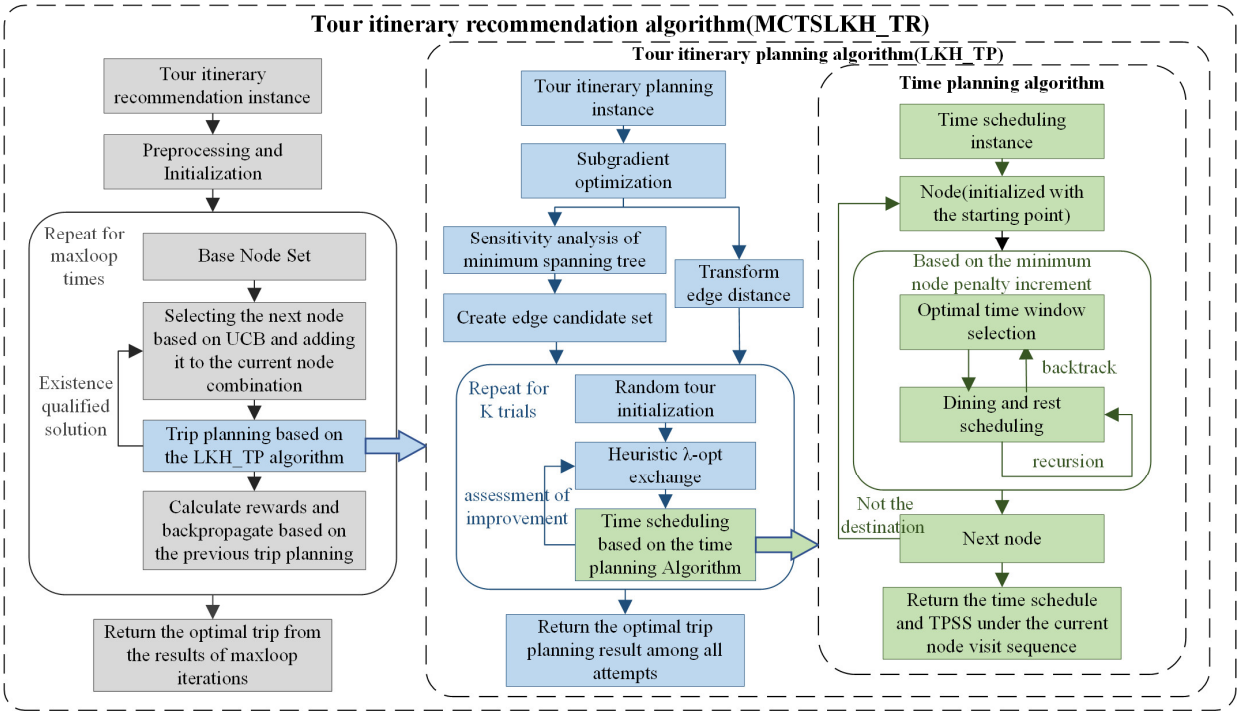


Figure 3. Framework and workflow of the tour itinerary recommendation algorithm.

4.1. Time Planning Algorithm Based on Tour Itinerary Planning Satisfaction

Since the time planning algorithm is frequently called in the program, we opted for simplifying calculations to enhance efficiency. Specifically, we set the objective as minimizing penalties under the premise of prioritizing the time spent on visiting scenic spots to maximize the $TPSS$. We implemented the time planning algorithm based on a local optimal approach. Algorithm 1 outlines our time planning function. The main process is as follows: for a given itinerary sequence, starting from the starting point, each node is processed sequentially. The algorithm selects time windows and arrangements for meals and accommodations based on minimizing the penalty for each node until the end of the itinerary, returning the time planning and the value of the itinerary objective function. If it is discovered during the process that the itinerary has exceeded the latest ending time or there are nodes that cannot be visited, the itinerary is deemed ineligible, and the algorithm returns to save unnecessary computations.

Algorithm 1 : TimePlanning()—Overview of Algorithm

Input : $I = Node\ sequence; Node\ Related\ attribute; MRTimePar;\ AvailableTime;\ Physical\ exertion\ limit;$
Output : $TPSS\ and\ Itinerary\ schedule;$

$TimePlan()$
  $N \leftarrow StartingPoint;$
  $CurrT \leftarrow DepartureTime;$
  $/* CurrT\ represents\ the\ current\ planned\ latest\ time, Updated\ as\ the\ planning\ progresses\ */$
  while$((N \leftarrow NextN)! = StartingPoint)$**do**
    $N.ArrT \leftarrow CurrT;$
    $NextN \leftarrow NextNodeInItinerary(N, I);$
    $OptimalTimeWindowSelection(N.ArrT, N);$
    $MealAndRestArrangement(N, Externalcall);$
    $P \leftarrow P + N.P, AVDur = AVDur + N.AVDur;$
    if $CurrentTime > LatestEndTime$ **or** $P > M$ **then**
      $return\ 0;$
    $TPSS \leftarrow \frac{AVDur}{AvailableTime}\left(1 - \frac{P}{CurrT - DepartureTime}\right)$
  $return\ TPSS;$

The optimal selection algorithm for time windows involves attempting all time windows for the node at the current time, calculating the corresponding waiting penalty or late penalty, and selecting the most suitable time window based on the principle of the minimal penalty. The current time is determined by the sum of the itinerary departure time and the already planned time. After determining the time window, the node will obtain a complete time period, status (early or late arrival), and actual duration of visitation.

The dining and accommodation arrangement algorithm encompasses handling for a hotel or restaurant designated by the user, as well as standard dining and accommodation scheduling. It employs recursion to handle cases where multiple dining and accommodation arrangements are required within a node's time period. In instances of conflicting expected dining and accommodation scheduling and sightseeing times, it selects from alternative options based on the principle of minimizing penalties. These alternatives consider various scenarios, including reducing expected duration, deviating from expected schedules, and backtracking within nodes. Please refer to Appendix C for the detailed process of the dining and accommodation arrangement algorithm.

Additionally, the algorithm introduces a dynamically adjusted expected dinnertime and physical exertion limit per day. When the lunchtime shifts, we accordingly adjust the expected dinnertime to ensure a consistent interval between the two meals. If there is additional physical exertion from the previous day, the physical exertion limit for the current day will be reduced accordingly, with the reduction amount proportional to the extra physical exertion from the previous day. These adjustments better meet the needs of tourists, making the itinerary planning more aligned with real-life situations.

Based on the above rules, we determine the time period of the nodes through the selection of time windows and the arrangement of dining and accommodation time, thus achieving acceptable time planning for the itinerary.

4.2. Tour Itinerary Planning Algorithm Based on Tour Itinerary Planning Satisfaction

We have implemented the tour itinerary planning algorithm based on tour itinerary planning satisfaction (named LKH_TP), using the LKH algorithm, which is one of the best algorithms for solving the Traveling Salesman Problem [28].

The LKH algorithm finds the optimal solution by performing heuristic edge exchanges on the initial solution. The main framework of the LKH_TP algorithm is similar to that of the LKH-3 algorithm.

Before running the algorithm, we perform preprocessing tasks, including creating a node set, establishing fixed edges from the endpoint to the starting point (with zero cost) for trips with different start and end points (to form the TSP problem), and disabling node visit sequences that would render nodes inaccessible.

As shown in Algorithm 2, in line 2, the candidate edge function is created to optimize the lower bound of the optimal itinerary using subgradient optimization. It calculates the likelihood of each edge being part of the optimal itinerary and associates a certain number of candidate edges with nodes. Limiting the search scope by creating a set of candidate edges, the search direction is guided based on the α measure.

Lines 4–9 use the LKH algorithm to find local optimal solutions for a specified number of runs (Runs) and return the best solution among them.

Algorithm 2 : LKH_TP—Overview of Algorithm

```
    Input : Node : Related attribute; MRTimePar; AvailableTime; Physical exertion limit;
    Output : TPSS and Itinerary schedule;
1   LKH_TP()
2    CreateCandidateSet();
3    BestTPSS ← 0;
4    for Run ← 1 to Runs do
5     TPSS_FT ← FindTour();
6     if TPSS_FT > BestTPSS then
7      BestTPSS ← TPSS_FT;
8      RecordBestTour();
9    return BestTour;
10  FindTour()
11   BetterTPSS ← 0;
12   for Trial ← 1 to Trials do
13    ChooseInitialTour();
14    TPSS_LK ← LinKernighan();
15    if TPSS_LK > BetterTPSS then
16     RecordBetterTour();
17     BetterTPSS ← TPSS_LK;
18     AdjustCandidateSet();
19   ResetCandidateSet();
20   return BetterTPSS;
```

Lines 11–20 explain the *FindTour* function, which conducts trial experiments. In each experiment, it first generates initial tour sequences using a combination of random and heuristic methods, ensuring the sufficient diversity of initial solutions while improving the efficiency. Then, the Lin–Kernighan function attempts to improve the selected initial tour through heuristic $\lambda - opt$ edge exchanges. For each tour sequence, the objective function value is obtained by calling the time planning algorithm, and this value is used to determine whether an improvement has occurred. Whenever a better tour is found, it is recorded, and the priority of candidate edges is re-ranked. Finally, the best tour among these experiments

is returned, and the original candidate edge set is re-established at the exit from *FindTour*. For a detailed explanation of the LKH algorithm, please refer to references [28,29]. The specified node visit order is achieved by adding constraints during the generation of the initial solution and edge exchanges.

4.3. Tour Itinerary Recommendation Algorithm Based on Tourist Comprehensive Satisfaction

We implemented the itinerary recommendation algorithm by combining the algorithm described in Section 4.2 with a variant of the Monte Carlo Tree Search (MCTS) algorithm. The MCTS algorithm is a search algorithm that has been successfully applied not only in artificial intelligence (AI) games such as chess and Go [30] but also in personalized itinerary recommendation [4,5,31]. The MCTS algorithm enables us to explore smaller, more promising regions of the solution space without traversing the entire solution space, thereby improving operational efficiency.

Algorithm 3 outlines the implementation process of the MCTSLKH_TR algorithm. In each iteration, we start from the user-selected base node combination (line 6), select nodes to add to the current combination, and perform itinerary planning until inserting a scenic spot would result in exceeding the cost limit, total time limit, or inaccessible scenic spots (lines 7–15). Then, the successful itinerary from the previous iteration is recorded as the recommendation for the current iteration, and itinerary-related parameters are calculated and propagated back to relevant candidate nodes (lines 16–27). Finally, the recommendation with the maximum objective function among these itinerary recommendations is returned (line 28).

Algorithm 3 : MCTSLKH_TR—Overview of Algorithm

Input : *Node* : *Related attribute*; *MRTimePar*; *Physical exertion limit*; *Travel style*; *Cost constraint*; R_{div} *type*;
Output : *I* : *Itinerary schedule*;

1 $MCTSLKH\_TourRec()$
2 $Preprocessing()$;
3 Initialize $NodeRecordR$, $NodeRecordV$, $NodeRecordTE$;
/ ∗ *respectively record the cumulative Reward, visit count, and TimeEfficiency of each node.* ∗ /
4 $TotalSeleNum \leftarrow 0$;
5 **for** $Iterations \leftarrow 1$ **to** $maxLoop$ **do**
6 $IsPass \leftarrow True$ $NodeSet \leftarrow BaseNodeSet$;
7 **while**$(IsPass = True)$ **do**
8 $PreviousNodeSet \leftarrow NodeSet$;
9 $N \leftarrow SelectNextNode()$;
10 $TotalSeleNum \leftarrow TotalSeleNum + 1$;
11 $NodeSet \leftarrow UpdateNodeSet(NodeSet, N)$;
12 $IsFound, IsPass \leftarrow HashSearch(HTable, Hash(NodeSet))$;
13 **if** $IsFound \neq True$ **then**
14 $IsPass, I_{tenp} \leftarrow LKH\_TP(NodeSet, abridged)$;
15 $HashInsert(HTable, Hash(NodeSet), IsPass)$;
16 $IsFound, TPSS, I_{tenp} \leftarrow HashSearch2(HTable2, Hash(PreviousNodeSet))$;
17 **if** $IsFound \neq True$ **then**
18 $TPSS, I_{tenp} \leftarrow LKH\_TourPlan(PreviousNodeSet, unabridged)$;
19 $HashInsert2\left(HTable2, Hash(PreviousNodeSet), TPSS, I_{tenp}\right)$;
20 $Reward_{I_{tenp}} \leftarrow CalculateRaward(PreviousNodeSet, TPSS)$;
21 $TimeEfficiency_{I_{tenp}}[] \leftarrow CalculateTimeEfficiency\left(PreviousNodeSet, I_{tenp}\right)$;
22 $BackPropagationR\left(NodeRecordR, I_{tenp}, Reward_{I_{tenp}}\right)$;
23 $BackPropagationV\left(NodeRecordV, I_{tenp}\right)$;
24 $BackPropagationTE\left(NodeRecordTE, I_{tenp}, TimeEfficiency_{I_{tenp}}[]\right)$;
25 $CategoryR_{I_{tenp}} \leftarrow CalculateCategoryR(PreviousNodeSet, Rtype)$;
26 $CSS_{I_{tenp}} \leftarrow Reward_{I_{tenp}} + CategoryR_{I_{tenp}}$
27 $I_{list} \leftarrow I_{list} \cup I_{tenp}$
28 *Return itinerary I with*maxCSS *from* I_{list};

In each iteration, we decompose the itinerary recommendation task into two parts: node selection and itinerary planning, which are completed using the MCTS algorithm and the itinerary planning algorithm, respectively. The MCTS algorithm is responsible for selecting unordered node combinations, while the itinerary planning algorithm generates complete itineraries. Subsequently, the optimal itinerary recommendation result is selected based on overall satisfaction. This process differs from conventional MCTS algorithms, which gradually determine the sequence of actions, but it is better suited to our itinerary recommendation problem. It ensures the quality of itinerary planning and makes itinerary rewards more valuable, thereby promoting the selection of subsequent high-quality nodes. Additionally, this approach can reduce the computational complexity and provide more powerful personalized services through independent itinerary planning optimization algorithms.

The recommended itinerary provided in each iteration gives feedback to the relevant candidate nodes through a reward function, where the reward function is represented by Equation (14) and includes all parts of the objective function except for the reward for diversity in attraction types. This is because the diversity in the attraction types of the itinerary is independent of the quality of the nodes, and the related requirements are optional.

$$Reward = TUS(I) * ISAS(I) * FS(I) \tag{14}$$

We simultaneously calculate both the time efficiency of nodes and the diversity of scenic categories within the itinerary. The time efficiency of a node within an itinerary is defined using Equation (15), where a_{last} represents the previous node a_i in the itinerary, a_{next} represents the next node a_i in the itinerary, and $TDur$ represents the travel time between these two points. The time efficiency of nodes is also propagated to every recommended node within the itinerary. This indicator is highly correlated with the time utilization rate of the itinerary and can highlight the inherent properties of the nodes themselves. Its average value can indirectly evaluate the positional attributes of the nodes, which we use as a heuristic factor considered in the selection node function.

$$TimeEfficiency_{a_i} = \frac{EVDur_{a_i}}{EVDur_{a_i} + TDur(a_{last}, a_i) + TDur(a_i, a_{next}) - TDur(a_{last}, a_{next})} \tag{15}$$

The diversity of scenic categories is calculated based on user preferences using Equation (5) or (6). The scenic category diversity reward, combined with the reward function, constitutes the objective function value of the itinerary.

In the process of incorporating scenic spots into the itinerary, the itinerary planning algorithm is used to determine whether it is still possible to insert additional spots. In such cases, optimizing the itinerary planning results is meaningless. Therefore, the itinerary planning algorithm is designed to exit as soon as an eligible itinerary planning is found (Lines 14), while a complete itinerary planning is performed when computing the final results in each iteration (Lines 18). Additionally, because the same node combination will yield the same result through itinerary planning, we utilize two hash tables to store obtained itinerary planning results, thus avoiding duplicate calculations (Lines 12–19). We enhance the efficiency of the algorithm through these measures.

Algorithm 4 outlines how to select the next node to add to the itinerary. We traverse the candidate scenic spots outside the current itinerary (Line 3), first obtaining the number of times each node has been selected. If a node has never been selected before, we directly choose that node (Lines 4–5). Next, we avoid adding nodes known to make the current itinerary infeasible through hash table queries (Lines 6–8), thereby improving the effectiveness of node additions and the diversity of node combinations. Then, we utilize the Upper Confidence Bound (UCB) algorithm to select the node with the highest UCB index from the remaining candidate nodes to add to the current itinerary (Lines 9–19).

Algorithm 4 : MCTSLKH_TR—SelectNextNode()

Input : $NodeRecordR$; $NodeRecordV$; $NodeRecordTE$; $TotalSeleNum$; $CandiNodeSet$; $CurrentNodeSet$;
Output : N : *Next node add to the itinerary*;
SelectNextNode()
  $NSet \leftarrow Null$;
  for $N \in CandiNodeSet$ *and* $N \notin CurrentNodeSet$ **do**
    $VisitCount_N \leftarrow GetVisitCount(N, NodeRecordV)$;
    if $VisitCount_N = 0$ **then** *return* N;
    $IsFound, IsPass \leftarrow HashSearch(HTable, Hash(NodeSet \cup N))$;
    if $IsFound = True$ *and* $IsPass = False$ **then**
      continue;
    $Reward_N \leftarrow GetReward(N, NodeRecordR)$;
    $TimeEfficiency_N \leftarrow GetTE(N, NodeRecordTE)$;
    $Exploit_N \leftarrow Reward_N / VisitCount_N$;
    $Explore_N \leftarrow C_p\sqrt{2lnTotalSeleNum / VisitCount_N}$;
    $Heur_N \leftarrow TimeEfficiency_N / VisitCount_N * SAS_N$;
    $NSet \leftarrow Record(NSet, N)$;
  for n *in* $NSet$ **do**
    $Normalization()$;
    $UCB_n \leftarrow \alpha_H * Heuristic_n + Exploit_n + Explore_n$;
  $N \leftarrow GetNwithMaxUCB()$
  return N;

The UCB algorithm applied to tree structures is commonly referred to as the UCT algorithm [32], typically with the objective of maximizing (16). The first part represents the exploitation value of the node, while the second part represents the exploration value of the node. Here, the parameter C_p determines the emphasis on exploring nodes. We set C_p to $1/\sqrt{2}$ to balance exploitation and exploration.

$$UCT_n = \frac{TotalReward_N}{VisitCount_N} + 2C_p\sqrt{\frac{2\ln VisitCount_{CurrN}}{VisitCount_N}} \tag{16}$$

$$UCB_n = \alpha_H * \frac{TotalTimeEfficiency_N * SAS_N}{VisitCount_N} + \frac{TotalReward_N}{VisitCount_N} + 2C_p\sqrt{\frac{2\ln TotalSeleNum}{VisitCount_N}} \tag{17}$$

Our UCB formula (as outlined in Equation (17)) significantly improves upon the original UCT algorithm. Because the candidate nodes have different intrinsic properties, we introduce a heuristic term $TotalTimeEfficiency_N / VisitCount_N * SAS_N$ that considers the average time efficiency and attractiveness of the nodes. It is beneficial for selecting nodes that maximize the rewards of the itinerary.

Moreover, since our node visit order depends on the itinerary planning algorithm, the choice of the next node is independent of the currently expanded node. We replace the selection count of the current node in the exploration term with the total count of selections for all nodes.

To control the influence weights of both components, we normalize both the exploitation and heuristic parts, with α_H denoting the weighting coefficient for the heuristic term.

5. Experiment and Example Analysis

This section introduces four sets of experiments related to the algorithm and provides an in-depth analysis of the experimental results. Specifically, Section 5.2 describes in detail the process of setting the number of iterations and heuristic weights in the algorithm. Section 5.3 analyzes the effectiveness of heuristics in the itinerary planning algorithm. Section 5.4 analyzes the effectiveness of the selection function in the itinerary planning algorithm. Section 5.5 explores the importance of considering the dining and accommoda-

tion times in itinerary planning by comparing them with a baseline algorithm that does not consider the dining and accommodation times. Finally, Section 5.6 evaluates the optimization performance of the proposed algorithm by comparing it with other algorithms. The algorithm proposed in this paper and all baseline algorithms are developed using the C language. All experiments were conducted on a computer with the following specifications: CPU: Intel Core i7-12700H 2.30 GHz, RAM: 16 GB.

5.1. Experimental Data

The experiment took the entire area of Hanzhong City as the research area, with a total area of approximately 27,200 square kilometers. Fifty key scenic spots in Hanzhong City were selected for the study. The distribution of the study area and selected scenic spots are shown in Figure 4. One hotel was selected to simulate the user's chosen starting point and destination. The travel time data between scenic spots was obtained from Tencent Maps' route planning service, which provides estimated travel times based on both distance and traffic conditions derived from big data. Information such as the popularity, opening hours, ticket prices, and categories of the scenic spots was sourced from Amap and Ctrip. The evaluation of the scenic spots is based on ratings from two websites and recent visitation numbers, comprising three factors in total. Among them, the ratings of the scenic spots are generated from user data. The recommended duration for visiting each scenic spot was sourced from Mafengwo and local tourism information websites.

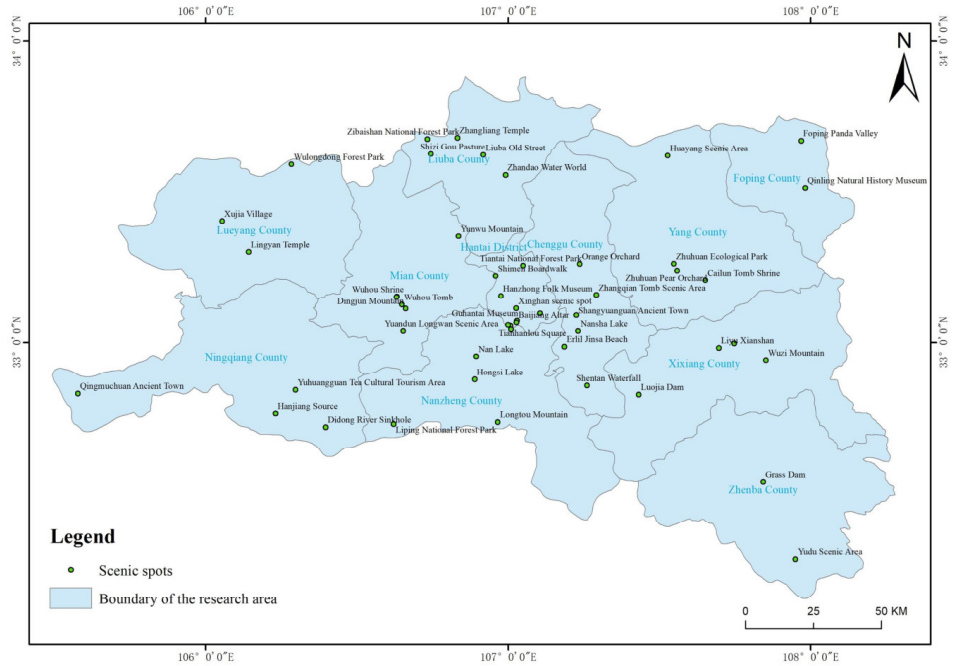


Figure 4. Study area and distribution overview of the selected scenic spots.

5.2. Parameter Debugging of Tour Itinerary Recommendation Algorithm

We conducted experiments to determine how to set the iteration count ($maxLoop$) and heuristic weight (α_H) in the MCTSLKH_TR algorithm. We performed a grid search across three cases, as shown in Table 1, to find the optimal values for these two parameters that balance efficiency and effectiveness effectively.

Table 1. Experimental cases.

Case	Planned Travel Time	Starting Point	Destination	Remaining Settings
1	One-day tour (day1 8:30–day1 21:30)	Hotel1	Hotel1	default
2	Three-day tour (day1 8:30–day3 21:30)	Hotel1	Hotel1	default
3	Five-day tour (day1 8:30–day5 21:30)	Hotel1	Hotel1	default

According to Figure 5, as the iteration count *maxLoop* increases, the number of new tour itinerary planning instances and number of duplicate tour itinerary planning instances both increase gradually when the number of tour itinerary planning requirements increases linearly. Moreover, the growth rate of new itinerary planning instances gradually slows down, while the growth rate of duplicate itinerary planning instances intensifies. This is because as the iteration count *maxLoop* increases, the solution space gradually converges, meaning that a greater proportion of itinerary planning instances has been calculated. When the growth rate of duplicate itinerary planning instances approaches the growth rate of itinerary planning requirements, it suggests that the algorithm hardly produces new solutions in the increasing iteration rounds. At this stage, the computational cost no longer offers a good balance between efficiency and value. Additionally, according to Table 2, after the growth rate of duplicate itinerary planning instances increases significantly, no better solutions are generated. Balancing algorithm efficiency and effectiveness, considering the scenarios of the three cases, we set the iteration count *maxLoop* to 500.

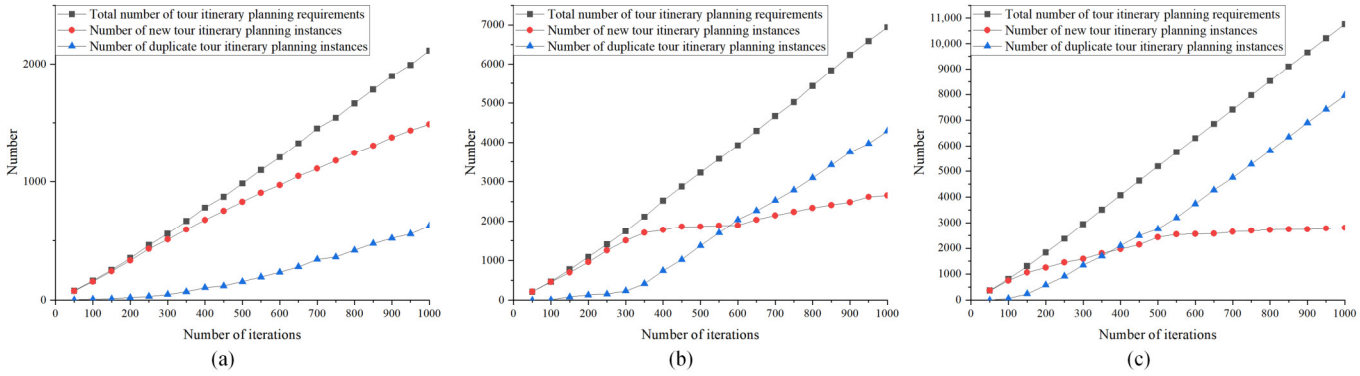


Figure 5. The change trend of the number of tour itinerary planning requirements, new tour itinerary planning instances, and duplicate tour itinerary planning instances with the number of iterations *maxLoop* (Figure (**a**), (**b**), and (**c**) correspond to cases 1, 2, and 3, respectively).

Table 2. The round in which the best result is found for different iterations in the three cases.

Number of Iterations	50	100	150	200	250	300	350	400	450	500
Case 1	3	88	88	88	88	88	88	88	88	88
Case 2	24	97	101	197	197	197	197	399	402	402
Case 3	17	66	66	66	215	215	215	215	438	438
Number of Iterations	**550**	**600**	**650**	**700**	**750**	**800**	**850**	**900**	**950**	**1000**
Case 1	88	88	88	88	88	88	88	88	88	88
Case 2	402	402	402	402	402	402	402	402	402	402
Case 3	438	438	438	438	438	438	438	438	438	438

According to Figure 6, as the heuristic weight α_H increases, the number of demands for itinerary planning and the quantity of duplicate itinerary planning instances gradually increase, while the number of new itinerary planning instances first increases and then decreases. The increase in these event quantities is due to the inclusion of nodes with higher time efficiency in the heuristic guidance, reducing the time cost of itinerary distance and allowing more scenic spots to be included in the itinerary within a limited time. The decrease in the number of new itinerary planning instances is caused by the increase in the heuristic weight α_H, which leads to an excessive reliance on factors contained in the heuristic, thereby narrowing the solution space and generating numerous duplicate itinerary planning instances. Since our itinerary recommendation aims to find the optimal solution among all results, an excessive reliance on the heuristic may reduce our exploration space, potentially missing better solutions. Therefore, we set the heuristic weight to the value that maximizes the number of new itinerary planning instances. Considering the situations of the three cases mentioned above, we set the heuristic weight α_H to 0.6.

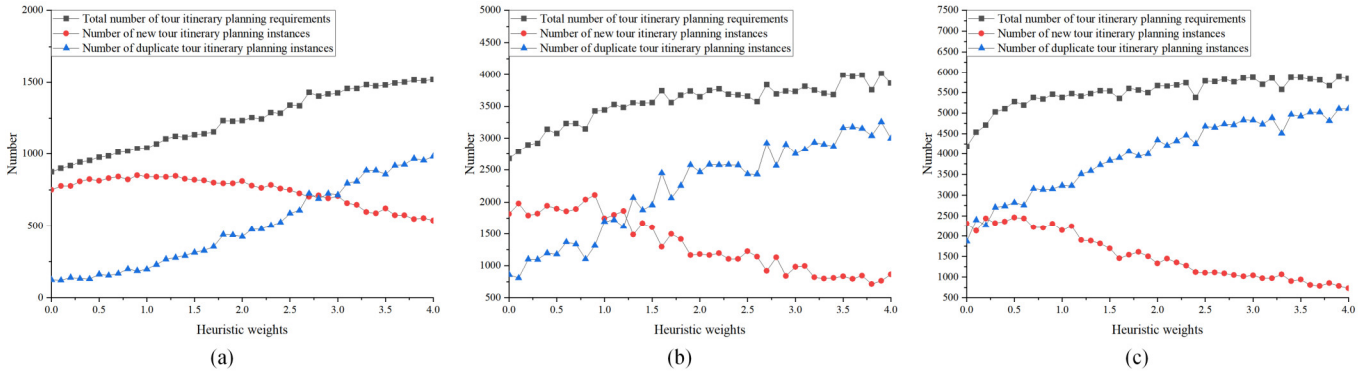


Figure 6. The change trend of the number of tour itinerary planning requirements, new tour itinerary planning instances, and duplicate tour itinerary planning instances with the heuristic weight α_H (Figure (**a**), (**b**), and (**c**) correspond to cases 1, 2, and 3, respectively).

It can be observed that as the planned tour duration increases across the scenarios, the intersection of the number of new itinerary planning instances and the number of duplicate itinerary planning instances occurs earlier in both Figures 5 and 6. This is because, with a constant number of candidate nodes, for cases where more nodes can be added, the itinerary result set will contain a larger proportion of common nodes. This leads to an increase in the proportion of duplicate itinerary planning instances.

5.3. Tour Itinerary Planning Algorithm Heuristic Effectiveness Analysis

Because time utilization and itinerary feasibility are not directly related to known factors, it is difficult to obtain heuristic information directly tailored to the objective function through preprocessing. We adopted the heuristic method from the original LKH algorithm, which is designed to optimize route costs. Taking the nodes in the recommended itinerary and corresponding conditions from Case 3 in Table 1 as an example, we conducted experiments to analyze the effectiveness of the route cost heuristic for our objective optimization.

Firstly, we compared random sequences with sequences generated using the LKH_TP algorithm and evaluated whether these sequences could form an eligible itinerary using a time planning algorithm. Figure 7 illustrates the distribution of route costs for random sequences, eligible solutions in random sequences, sequences generated using the heuristic, and eligible solutions in sequences generated using the heuristic. Table 3 indicates the number of occurrences for each event. The results show that the validity rate of random sequences is 0.18%, while the validity rate of sequences generated using the heuristic is 25.74%.

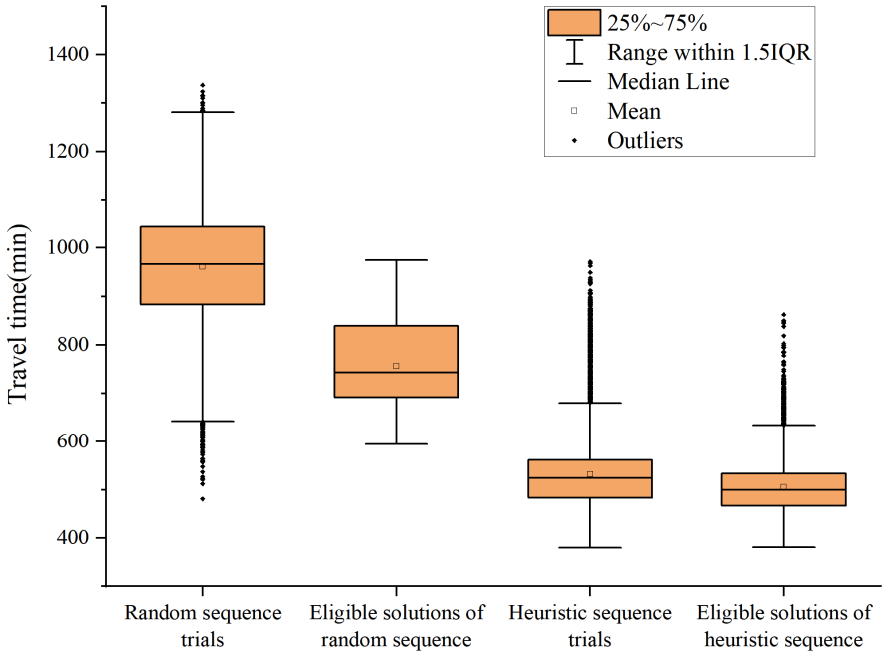


Figure 7. Route cost distribution for random and heuristic trial sequences and the eligible sequences therein.

Table 3. The number of random and heuristic sequence trials and their eligible sequences.

Event	Trials	Eligible Solutions	Pass Rate
Random sequence	1,000,000	1823	0.18%
Heuristic sequence	113,823	29,296	25.74%

We can observe that the route heuristic aids in exploring sequences with lower route costs, and eligible solutions tend to have sequences with lower route costs. The use of heuristics significantly increases the qualification rate of experimental sequences, facilitating the discovery of eligible solutions and itinerary optimization.

Furthermore, we analyzed the impact of route costs on tour itinerary planning satisfaction. Due to the predetermined time constraints, experimental sequences often resulted in an ineligible itinerary. To address this, we conducted the analysis without imposing a total travel time limit to ensure the integrity of the itinerary.

Figure 8 illustrates the relationship between route costs and their corresponding average minimum required available time and average tour itinerary planning satisfaction. Due to the influence of time windows for tourist attractions as well as meal and accommodation times, trips with the same route costs exhibit significant differences in the minimum required available time and itinerary satisfaction. To mitigate the impact of these factors, we use averages to highlight the overall trends of the minimum required available time and itinerary satisfaction with respect to route costs. The correlation coefficients between them were found to be 0.901 and -0.914, respectively. These data were derived from the time planning results of one million random sequences without a total time limit.

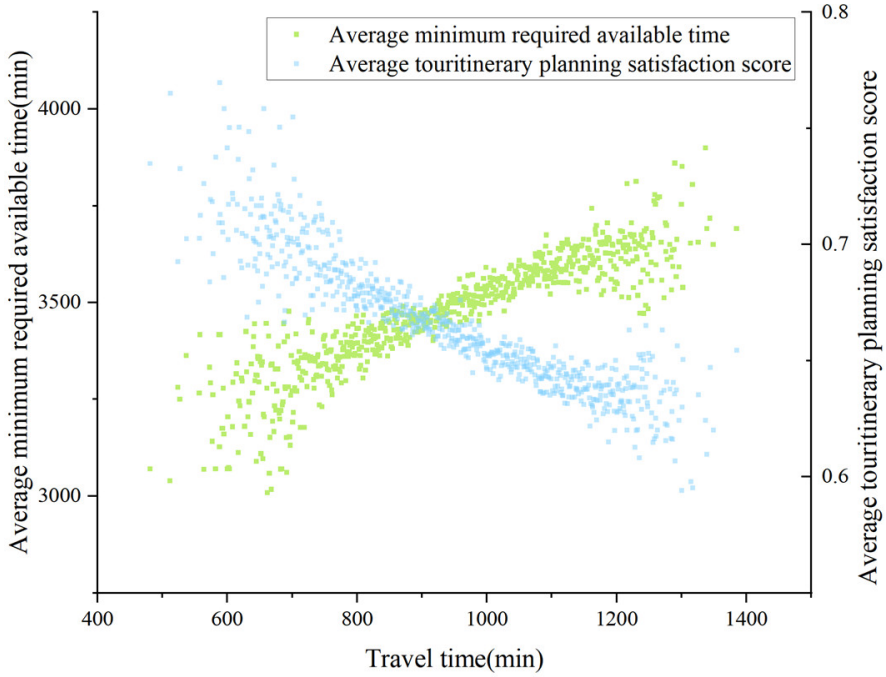


Figure 8. Relationship between route cost and average minimum required available time and average tour itinerary planning satisfaction score.

It can be predicted that under predetermined time constraints, lower route costs lead to a higher probability of forming an eligible itinerary and a reduced likelihood of time loss for visiting tourist attractions within the itinerary, owing to the competition between travel time and time spent at tourist attractions. This also increases the chances of obtaining a more satisfactory itinerary. Therefore, by emphasizing attempts with lower route costs through heuristics, we can protect the visiting time and discover a better itinerary.

In conclusion, the heuristic approach to optimizing route costs is well-suited for our itinerary planning needs.

5.4. Tour Itinerary Recommendation Algorithm Selection Function Effectiveness Analysis

We also analyzed the effectiveness of the itinerary recommendation algorithm selection function through the three cases in Table 1. Firstly, we examined the effectiveness of the added heuristic factors. Figure 9 illustrates the relationship between the heuristic factors of each candidate node and the corresponding factors in the reward for the three cases. The average correlations are 0.87, 0.76, and 0.86, respectively. It can be observed that there is a high positive correlation between the heuristic and the reward. Additionally, the heuristic of the nodes exhibits greater variability compared to the reward, especially in itineraries containing more recommended attractions. This is because the heuristic removes the influence of other nodes in the reward. The inclusion of the heuristic highlights the inherent value of candidate attractions, facilitating the selection of superior attractions among those with similar rewards, which positively contributes to discovering itineraries with higher rewards. Moreover, it can be observed that in cases with longer travel durations, attractions generally exhibit a higher time efficiency, as more nodes in the itinerary make it easier to include a larger proportion of attractions along the route.

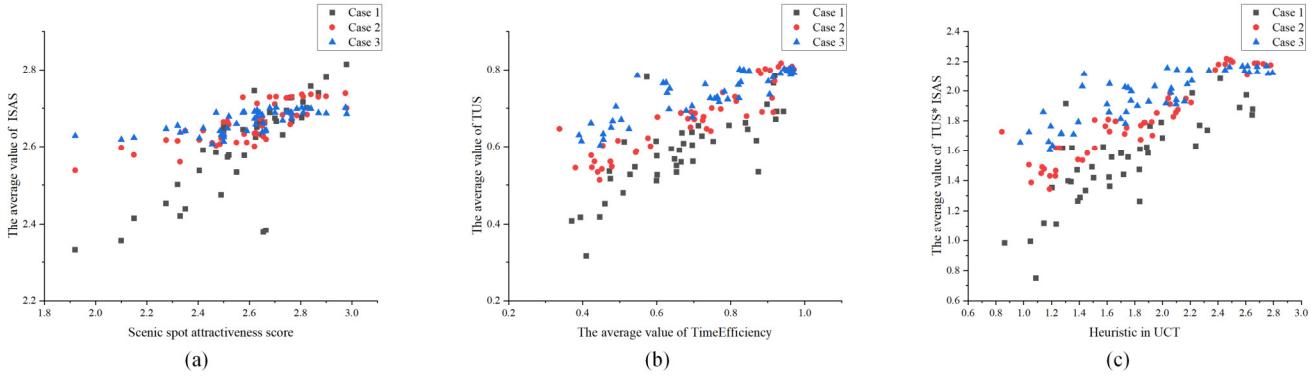


Figure 9. The relationship between the heuristic factors of candidate nodes and their corresponding factors in the rewards: (**a**) Scenic spot attractiveness score and the average value of ISAS; (**b**) The average value of TimeEfficiency and the average value of TUS; (**c**) Heuristic in UCT and the average value of TUS * ISAS.

Then, we further analyzed the effectiveness of the selection function by examining the average rankings of the recommended nodes in the optimal itinerary in terms of their selected number and various dimensions in the selection function (among the 50 candidate nodes). We also assessed the distribution of nodes selected in each iteration during the algorithm process. The reference optimal values in Table 4 are based on the theoretical optimal average ranks derived from the number of recommended nodes. It can be observed that our actual values are close to or equal to the optimal values, significantly outperforming the average rank of 25 obtained randomly. This indicates that the utilization dimensions in the selection function effectively guide the selection of nodes. Additionally, from Figure 10, it can be seen that our selection function balances exploration and exploitation, gradually converging during exploration. This not only facilitates the discovery of a high-quality itinerary but also ensures a sufficient number of attempts. Overall, our selection function demonstrates excellent effectiveness in itinerary recommendation.

Table 4. The average rankings of the recommended nodes in the optimal itinerary.

Case	Optimal Value Reference	$VisitCount_N$	$\frac{Reward}{VisitCount_N}$	$\frac{TimeEfficiency}{VisitCount_N}$	SAS	UCB Excluding Exploration Part
1	2	2	2	3.667	5.667	2
3	5	6.111	7.556	7.111	13.889	6.333
5	8	8	9	9.125	15.933	8

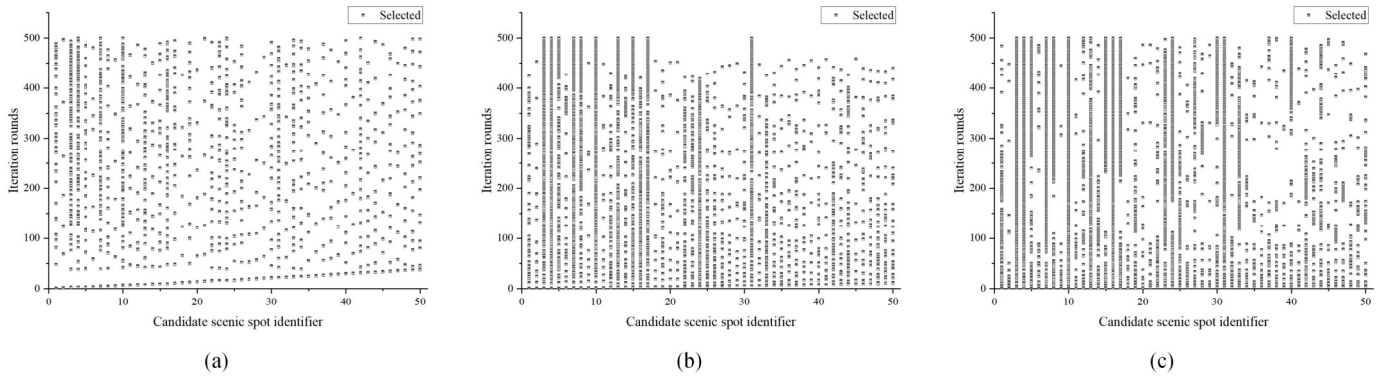


Figure 10. Selection of scenic spots in each iteration (Figure (**a**), (**b**), and (**c**) correspond to cases 1, 2, and 3, respectively).

5.5. Analysis of Tour Itinerary Planning Results

As a key component of the algorithm, the results of itinerary planning directly impact the itinerary arrangements in the recommendation results. We conducted an analysis of the itinerary planning results through experiments. Among the fundamental elements, travel, eating, lodging, transportation, and sightseeing are all crucial. However, past research has often overlooked the importance of properly arranging dining and accommodation times. Therefore, we primarily focused on the significance of considering dining and lodging times in the temporal dimension. In our experiments, we used the recommended nodes and their corresponding conditions from the three cases in Table 1 for itinerary planning. We also selected the following two baseline algorithms, both based on the LKH algorithm:

(1) LKH: This algorithm plans an itinerary based on minimizing route costs, which is the most basic and common method of itinerary planning. It does not consider the temporal feasibility of itinerary arrangements.
(2) LKH-TW: This algorithm plans an itinerary based on the optimal $TPSS$, employing the same objective function as presented in this paper. It considers the time windows of attractions but does not consider dining and accommodation times during the itinerary. The algorithm introduces the optimal time window selection algorithm proposed in this paper to accommodate multiple time windows of attractions in multi-day trips. Additionally, to avoid unreasonable overnight travel, any intersection between attraction time windows and resting time is removed, implying a rigid arrangement for accommodation time. This algorithm represents the current level of time constraints in itinerary planning problems in the related literature.

We utilized the following seven evaluation metrics to measure the provided itinerary solutions:

(1) The travel time ($TravelT$).
(2) Whether the itinerary exceeds the planned time ($Timeout$) (indicated by 0 or 1).
(3) Whether there are attractions that cannot be visited ($UnableVisit$). An attraction is considered unable to be visited if, upon arrival at the node, it is already past the last entry time for that attraction within the planned travel time.
(4) The actual duration of attraction visits planned in the itinerary ($AVisDur$).
(5) The ratio of $AVisDur$ to expected visit duration ($AEVDurRatio$). The difference between the two values stems from late arrivals at attractions.
(6) The itinerary penalty ($Penalty$).
(7) Tour itinerary planning satisfaction ($TPSS$).

Due to the inability of the LKH algorithm to produce evaluation metrics beyond the route cost and the evaluation metrics of the LKH_TW algorithm results not encompassing the dining and accommodation time of tourists, for the sake of uniform evaluation criteria, the solution proposed in this paper was adopted as the standard. For the itinerary planning results obtained using the two baseline algorithms, evaluation metrics other than the route cost were calculated using the time planning algorithm proposed in this paper, according to their visiting order. The results of the evaluation metrics are shown in Table 5, and Table 6 analyzes the sources of penalties for each algorithm in the cases (the values in parentheses represent the original results of $AVisDur$, $Penalty$, and $TPSS$).

Table 5. The evaluation metrics results of tour itinerary planning experiments.

Case	Algorithm	*TravelT* (min)	*Timeout*	UnableVisit	*AVisDur* (min)	*AEVDurRatio*	*Penalty*	*TPSS*
1	LKH	46	0	0	420	100.00%	9	0.727
	LKH_TW	66	0	0	420(420)	100.00%	13(0)	0.723(0.737)
	LKH_TP	46	0	0	420	100.00%	9	0.727
2	LKH	238	0	1	1246	85.58%	88 + M	0
	LKH_TW	260	1	0	1265(1412)	86.88%	344 + M [1](29)	0(0.821)
	LKH_TP	264	0	0	1424	97.80%	68	0.817
3	LKH	426	0	1	2128	83.98%	231 + 2M	0
	LKH_TW	465	0	1	2062(2504)	81.37%	268 + 2M(124)	0(0.856)
	LKH_TP	473	0	0	2489	98.22%	189	0.848

[1] For the purpose of analysis, the penalty M is represented separately.

Table 6. Analysis of penalty sources in itinerary planning results.

Case	Algorithm	*Penalty*	Source of Penalty					
			WaitP	*LateArrP*	*MealDeviaP*	*MealReduP*	*RestDelaP*	*PhysFatiP*
1	LKH	9	9					
	LKH_TW	13			8	5		
	LKH_TP	9	9					
2	LKH	88 + M	71	M	8		9	
	LKH_TW	344 + M	112	217 + M	15			
	LKH_TP	68	2	32	25		9	
3	LKH	231 + 2M	119	2M	59		53	
	LKH_TW	268 + 2M	158	66 + 2M		44		
	LKH_TP	189		45	15	120	9	

We analyze the results as follows:

(1) *TravelT*: The LKH algorithm achieves the lowest value in all three cases. The LKH_TW algorithm and the proposed method are roughly comparable, but there is no significant enhancement compared to the LKH algorithm. This is because although our objective function does not directly consider route costs, the extent to which routes can be sacrificed to improve the time spent at attractions and itinerary feasibility is limited. In the optimization scenario of the algorithm, time constraints will restrict unreasonable routes.

(2) *Timeout* and *UnableVisit*: Except for Case 1 where all three algorithms provide eligible results, only the proposed method offers eligible results in the remaining cases. In Case 2, the LKH algorithm misses one attraction, while the LKH-TW algorithm exceeds the planned travel time. In Case 3, both the LKH and LKH-TW algorithms miss two attractions. This indicates that the itinerary planning results provided by the two baseline algorithms are likely to be unreasonable in reality. Moreover, as the travel duration increases and the number of itinerary nodes grows, the probability and severity of unreasonable results tend to escalate.

(3) *AVisDur*, *AEVDurRatio*, *Penalty*, and *TPSS*: It can be observed that, due to there being fewer constraints, the original *TPSS* of the results from the LKH_TW algorithm is superior to that of our proposed algorithm, indicating a higher attraction visit time or fewer penalties. However, under unified time planning, its results are comparable to those of the LKH algorithm, resulting in significant losses in attraction visit time and penalties for waiting and late arrivals. This is because it overlooks the necessary dining time expenditure for tourists during travel and does not consider rest and accommodation time in the itinerary based on specific circumstances. This can lead to a disconnect between visitors' actual experiences during travel and the planned

time arrangements. It is evident that dining and accommodation time have a significant impact on itinerary planning. Meanwhile, our algorithm yields optimal results in these four indicators, with an average $AEVDurRatio$ reaching 98.67%, which is 9.82% higher than that of the LKH algorithm and 10.34% higher than that of the LKH-TW algorithm.

Through the above analysis, it can be observed that considering the arrangement of dining and accommodation time is of significant importance for detailed itinerary planning under the self-drive tour mode. Additionally, the results of the itinerary planning algorithm proposed in this paper approach the expected conditions across various indicators, ensuring a relatively complete expected visitation time and regular meal and rest schedules. The provided itinerary planning results are highly feasible.

5.6. Analysis of Tour Itinerary Recommendation Results

We conducted an analysis of the tour itinerary recommendation results through experiments. To evaluate the optimization performance of the algorithm under the same rules, we developed two algorithms as baseline algorithms. These algorithms integrate our time planning algorithm with advanced related algorithms and utilize the same objective function as our algorithm. The two baseline algorithms are as follows:

a. GreedyLKH_TR: This algorithm combines our tour itinerary planning approach with the sequential attraction insertion method proposed in [9,10]. It begins with the basic node combination and attempts to add each candidate attraction to the current node combination individually. Then, it applies the LKH_TP algorithm to plan the itinerary for each combination and selects the node that maximizes the objective function for addition. This process continues until no more attractions can be inserted. We refer to this algorithm as the GreedyLKH_TR algorithm.
b. MCTS_TR: This algorithm combines our time planning approach with a variant of the MCTS algorithm proposed in [4,31]. Its main process is similar to ours, but after selecting a node each time, it directly inserts it at the end of the current itinerary to form the itinerary sequence, without going through itinerary planning. The selection function used additionally incorporates a heuristic for the route cost to the current node on top of our algorithm and adjusts the total number of times all nodes in the exploration term of the selection function have been chosen to the number of times the current node has been selected. The feedback-related function remains the same as in our algorithm. We set the number of iterations to 10,000.

On top of the three cases in Table 1, the experimental cases added the user's interest label as IS = {C1: Historical and Cultural}, while the remaining settings still used default parameters. We employed the following six evaluation indicators to measure the provided itinerary solutions:

(1) The proportion of itinerary time ($TimePropo$), which represents the ratio of the total duration of the recommended itinerary to the user's planned travel duration;
(2) Itinerary Scenic Spot Attractiveness Score ($ISAS$);
(3) Time Utilization Score (TUS);
(4) Feasibility Score (FS);
(5) Comprehensive satisfaction score (CSS);
(6) $Runtime$.

These evaluation metrics can be used to comprehensively measure the quality and practicality of the recommended itinerary solutions. The experimental results are shown in Table 7, which also includes several values relevant to the evaluation metric calculations, such as the number of recommended attractions ($RecNum$), the actual duration of attraction visits planned in the itinerary ($AVisDur$), the duration of the itinerary ($ItinDur$), and the itinerary penalty ($Penalty$). Table 8 outlines the average values of these metrics across the three cases.

Table 7. The evaluation metrics results of tour itinerary recommendation experiments.

Case	Algorithm	*RecNum*	*AVisDur* (min)	*ItinDur* (min)	*TimePropo*	*Penalty*	*TUS*	*ISAS*	*FS*	*CSS*	*RunTime* (s)
1	GreedyLKH_TR	3	420	729	93.46%	8	0.737	4.525	0.989	3.298	0.28
	MCTS_TR	3	402	704	90.26%	4	0.705	4.55	0.994	3.191	3.77
	MCTSLKH_TR	3	420	729	93.46%	8	0.737	4.525	0.989	3.298	0.76
2	GreedyLKH_TR	6	1204	3624	99.02%	115	0.704	4.291	0.968	2.925	2.27
	MCTS_TR	8	1358	3652	99.78%	143	0.794	3.906	0.961	2.980	15.58
	MCTSLKH_TR	10	1508	3652	99.78%	202	0.882	4.26	0.945	3.549	9.61
3	GreedyLKH_TR	12	2046	6503	99.43%	265	0.718	4.247	0.959	2.924	14.45
	MCTS_TR	12	1733	6495	99.31%	139	0.608	4.33	0.979	2.577	28.03
	MCTSLKH_TR	15	2352	6532	99.88%	176	0.825	3.984	0.973	3.199	37.09

Table 8. Average evaluation metric results of tour itinerary recommendation experiments.

Algorithm	*TimePropo*	*TUS*	*ISAS*	*FS*	*CSS*	*RunTime*(s)
GreedyLKH_TR	97.30%	0.720	4.354	0.972	3.049	5.67
MCTS_TR	96.45%	0.702	4.262	0.978	2.916	15.79
MCTSLKH_TR	97.71%	0.815	4.256	0.969	3.349	15.82

We analyze the results as follows:

(1) *TimePropo*: It can be observed that the recommended itineraries generated by all three algorithms effectively utilize the planned travel time of the tourists. The average *TimePropo* exceeds 96% across the board. Maximizing the use of time under time constraints is a fundamental requirement for tourists and is also a necessary condition for enhancing comprehensive satisfaction.

(2) *TUS*, *ISAS*, *FS*, *CSS*: Among the CSS and related evaluation metrics, our algorithm achieves the best results in CSS and TUS across all three cases. The average CSS surpasses that of the GreedyLKH_TR algorithm by 9.84% and exceeds that of the MCTS_TR algorithm by 14.85%. Similarly, the average TUS outperforms that of the GreedyLKH_TR algorithm by 13.20% and that of the MCTS_TR algorithm by 15.99%. Over the accumulated 9-day itineraries in the three cases, our algorithm provides an additional 610 min and 787 min of attraction visitation time compared to the GreedyLKH_TR and MCTS_TR algorithms, respectively. The FS of all three algorithms is relatively similar across the three cases, indicating that the itinerary results generated as part of the algorithm based on time planning closely match the tourists' expectations. While the average ISAS of our algorithm is slightly lower than that of the two baseline algorithms, since the improvement of ISAS is direct and relatively simple, and the proposed algorithm selects the tour itinerary with a slightly lower ISAS but higher comprehensive satisfaction through trade-offs, it can be seen that the optimization ability of the proposed algorithm is superior to the two baseline algorithms.

(3) *Runtime*: It can be observed that the MCTSLKH_TR algorithm requires a longer runtime compared to the GreedyLKH_TR algorithm under the same scenarios. However, this disadvantage is acceptable as it still promptly meets the user's needs. According to the time complexity of the LKH algorithm, which is approximately $O(n^{2.2})$ [29], theoretically, the MCTSLKH_TR and GreedyLKH_TR algorithms both have time complexities ranging from $O(n^{2.2})$ to $O(n^{3.2})$. However, the MCTSLKH_TR algorithm only requires a constant number of complete itinerary planning iterations, with the remaining cases only needing to find eligible solutions. This significantly shortens the runtime and can be further accelerated through hash tables. Therefore, the time complexity of the MCTSLKH_TR algorithm is superior to the GreedyLKH_TR algorithm, and it is foreseeable that as the problem scale increases, the runtime required by the MCTSLKH_TR algorithm will outperform the GreedyLKH_TR algorithm.

Additionally, the MCTSLKH_TR algorithm can directly control the runtime not to exceed a certain limit. The average runtime of the MCTSLKH_TR algorithm across the three cases is comparable to that of the MCTS_TR algorithm and has an advantage in small-scale cases.

In conclusion, the tour itinerary recommendation algorithm proposed in this paper can achieve superior itinerary recommendations, particularly in terms of time utilization. It results in tourists being able to spend more time visiting attractions and accessing a greater number of attractions within essentially the same itinerary duration.

It can be observed that the GreedyLKH_TR algorithm maintains relatively stable itinerary recommendation quality. However, the sequential insertion method is prone to becoming trapped in local optima, making it difficult to achieve better results. As the number of recommended nodes increases, the solution quality of the MCTS_TR algorithm gradually decreases. This is because with the increase in the number of recommendations, the solution space grows factorially, which cannot be compensated for by increasing the number of iterations in the MCTS_TR algorithm. Additionally, the inability to guarantee the rationality of itinerary planning reduces the reference value of itinerary rewards, and the need to select the optimal node at the current position increases the difficulty of selection.

In contrast, our algorithm combines a variant of the MCTS algorithm, balancing exploration and exploitation. It ultimately selects the best option from a global perspective. Furthermore, we separate the itinerary recommendation task into node combination and itinerary planning. Through itinerary planning, we can rapidly cover a large number of solutions and obtain high-quality feedback to guide node selection. Therefore, our method can provide high-quality itinerary planning and maintain stability as the number of recommended nodes increases.

What needs to be added to the discussion is as follows: In this chapter's experiments, penalty metrics are computed by comparing the recommended itinerary with the user's expected scenario. However, in actual travel scenarios, deviations may exist between the user's anticipated dining, accommodation, and sightseeing durations and the expected scenario. Additionally, while many constraints are considered in the temporal dimension, the itinerary may also include other activities with uncertainties. Therefore, these metrics primarily reflect the deviation between the planned itinerary and the expected scenario. We lack real data to reflect the differences between user planning and actual travel scenarios.

Despite the disparities between user-inputted expectations and real scenarios, they do not affect our conclusions. Our algorithm is compared with the baseline under identical conditions. Although the experimental cases have certain limitations, they were not deliberately selected, so the experimental results still have a certain degree of comparability and credibility. Our primary focus is on providing optimal itinerary recommendations based on user expectations. Hence, despite some deviations, they do not significantly impact our research conclusions.

6. Conclusions

This paper analyzed the issues existing in the field of current tour itinerary recommendations. To address these problems, we constructed a comprehensive satisfaction model as the objective function and proposed the MCTSLKH_TR algorithm to solve them. The algorithm aims to provide detailed and satisfying itinerary recommendations for self-drive tours. The experimental results show that our itinerary recommendations exhibit higher comprehensive satisfaction compared to two baseline algorithms. Additionally, our algorithm considers more constraints in the temporal dimension compared to related work in tour itinerary planning, resulting in highly feasible itinerary arrangements.

In the future, we plan to expand this work in several ways. First, we will incorporate scenic roads into the model to provide certain rewards in itinerary planning and recommendations. Second, for dining and accommodation during the user's tour itinerary, we will design the algorithm to not only make arrangements in terms of time but also recommend restaurants and hotels that meet the needs and preferences of the user according

to the location of their tour itinerary. Third, we will use intelligent algorithms to examine user data and evaluate the level of interest in attractions. In summary, tour itinerary recommendations should continue to evolve towards intelligence and personalization.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app14125195/s1. Table S1: Scenic Spot Attributes Data and Travel Time Data Between Scenic.

Author Contributions: Data curation, L.W. and D.L.; methodology, D.L. and Y.Z; software, D.L. and Y.D.; formal analysis, D.L., L.W. and Y.Z.; writing—original draft, D.L. and L.W.; writing—review and editing, Y.Z., J.K. and Y.D.; supervision, Y.D. and J.K.; project administration, J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the key research and development projects of the Department of Science and Technology of Shaanxi Province (Grant No. 2020ZDLSF06-07), the research project of the Shaanxi Provincial Department of Transportation (Grant No. 21-03R), and the Shaanxi Provincial Key Research and Development Plan Project (Grant No. 2023-YBGY-200).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article and Supplementary Materials.

Conflicts of Interest: This research was conducted as part of a collaborative project between Chang'an University and Aerial Photogrammetry and Remote Sensing Group Co., Ltd. Author Yi Dong, affiliated with Aerial Photogrammetry and Remote Sensing Group Co., Ltd., contributed to the design of the study. The funding sponsors had no role in the design of the study.

Appendix A

The following describes the basic concepts involved in the problem and introduces their significance in this study, specific forms, and quantitative methods. The concrete form is designed to adapt to the relevant function. The defined quantification method provides a reasonable reference, and it is used for the calculation of experimental data in this paper. It can be replaced by other methods in practical applications.

Definition 1. *Node. In this paper, all locations involved in the itinerary are collectively referred to as nodes. They possess the following attributes: role (origin, destination, or waypoint), type (scenic spot, restaurant, hotel), time windows, expected duration, travel time to other nodes, and the next node in a fixed itinerary sequence (default is none). For scenic spots, additional attributes include popularity, user interest level, ticket price, category, grade, physical exertion, availability of internal dining and accommodation, and whether they are candidate scenic spots. Additionally, they have attributes such as adjacent nodes in the itinerary sequence, node penalties, internal dining and accommodation arrangements, and time cycles, which are used in the algorithm's calculation process and output.*

Definition 2. *Meal and rest accommodation time parameters (MRTimePar). We guide the scheduling of meal and accommodation times in the itinerary planning with six parameters, including the expected lunch start time, dinner start time, rest and accommodation start time, lunch duration, dinner duration, and rest and accommodation duration. These parameters are collectively referred to as meal and rest time parameters.*

Definition 3. *Node time windows. The time window of a node represents a period during which it is accessible and available. In this paper, each time window is represented by a triplet consisting of the opening time ($Open_{a_i}$), the last entry time ($Last_{a_i}$), and the closing time ($Close_{a_i}$). A node may have multiple time windows, whereas for the starting point, ending point, and nodes open throughout the entire journey, their time windows are defined from the start time to the end time of the planned travel.*

Definition 4. *Node time period. The node time period is a time interval composed of the arrival time of the node ($ArrT_{a_i}$) and the arrival time of the next node ($NextNArrT_{a_i}$). It includes three stages: from the arrival time of the node to the start time ($StartT_{a_i}$) of node visitation, from the start time of node visitation to the end time ($EndT_{a_i}$) of node visitation, and from the end time of node visitation to the arrival time of the next node.*

Definition 5. *Scenic spot physical exertion. We evaluated and quantified the physical exertion (PE) required to visit a scenic spot. As shown in (A1), the PE_{a_i} of a scenic spot a_i is defined as the product of the recommended visit duration ($RVDur_{a_i}$) of the scenic spot and the physical exertion coefficient β_{a_i}. The physical exertion coefficient β_{a_i} depends on the terrain of the scenic spot. In this paper, climbing scenic spots $\beta = 2$, and other scenic spots $\beta = 1$. Under the default daily physical exertion limit, it is possible to avoid planning itineraries that involve continuous climbing in a single day in the algorithm.*

$$PE_{a_i} = RVDur_{a_i} * \beta_{a_i} \tag{A1}$$

Definition 6. *Scenic spot interest level (Int_{a_i}). We define two levels of scenic spot categories, where the secondary category serves as a further differentiation of the primary category. We also define a set of scenic spot labels, denoted as $SS_{a_i} = \{C1_{a_i}, C2_{a_i}, L_{a_i}\}$, comprising three dimensions: primary category $C1_{a_i}$, secondary category $C2_{a_i}$, and grade L_{a_i}. Tourists can select their interests from all category labels and grade labels. Assuming the user selects the set of interest labels $IS = \left\{C1^1, C1^2, C2^1, C2^2, C2^3, L^1, L^2\right\}$, the user's interest level in scenic spot a_i is represented by (A2), where α represents a fixed initial value, set to 0.5 in this study. IS_{a_i} represents the specificity of IS to a_i, as shown in (A3). If IS contains the secondary category $C2_{a_i}$ of a_i , then IS_{a_i} includes the primary category $C1_{a_i}$ of a_i.*

$$Int_{a_i} = \alpha + \frac{|IS_{a_i} \cap SS_{a_i}|}{|SS_{a_i}|} \tag{A2}$$

$$IS_{a_i} = \begin{cases} IS \cup \{C1_{a_i}\}, & C2_{a_i} \in IS; C1_{a_i} \notin IS \\ IS, & otherwise \end{cases} \tag{A3}$$

Definition 7. *Travel style. The travel style denotes the compactness of the itinerary. We introduce inflation coefficients for both the travel time between destinations and the recommended duration for visiting scenic spots. By adjusting the travel time and recommended visit duration with certain proportions, we control the compactness of the itinerary. For user convenience, we have defined three sets of default coefficients corresponding to relaxed, moderate, and compact travel styles.*

Definition 8. *Expectation and penalty. We distinguish problem inputs into hard conditions and expectations. Expectations refer to conditions in tour itinerary planning that are not strictly required but are desired to be achieved as much as possible. These include the duration of visiting nodes, meal and rest times, physical exertion limits, and the absence of waiting times throughout the itinerary. Since the actual itinerary often cannot fully meet these expectations, and also because this is unnecessary, we allow for deviations from these expectations in tour itinerary planning. However, to evaluate and minimize these deviations, we introduce penalty measures.*

Appendix B

We have defined six types of penalties to evaluate the deviation of itinerary planning from each expected scenario, including waiting penalty, late arrival penalty, mealtime deviation penalty, mealtime reduction penalty, rest accommodation delay penalty, and physical fatigue penalty. The following describes the calculation methods for these penalties.

(1) Waiting penalty

The waiting penalty ($WaitP$) arises when the arrival time at a node is earlier than the opening time of the time window currently selected. It is a proportional function of the waiting duration, where the waiting duration equals the time interval between the arrival time at the node and the opening time of the node's time window, minus the duration spent on dining and accommodation during this period.

(2) Late arrival penalty

The late arrival ($LateArrP$) penalty arises when the arrival time at the node cannot satisfy the expected duration of the visit under the currently selected time window. Figure A1 represents $LateArrP$ as a piecewise function of the lost visit duration. The lost visit duration is equal to the expected visit duration ($EVDur_{a_i}$) minus the currently accessible duration, and M represents a sufficiently large number.

(3) Meal- and rest-related penalties

Meal- and rest-related penalties include the meal deviation penalty $MealDeviaP$, meal duration reduction penalty $MealReduP$, and rest accommodation delay penalty $RestDelaP$. For each dining and accommodation arrangement, these penalties arise from the deviation of the scheduled time from the expected time or from not meeting the expected duration. These penalties are proportional functions of the respective deviations.

(4) Physical fatigue penalty.

The physical fatigue penalty ($PhysFatiP$) arises when the cumulative physical exertion at the scenic spots for the day exceeds the expected daily physical limit. It is a proportional function of the additional physical exertion.

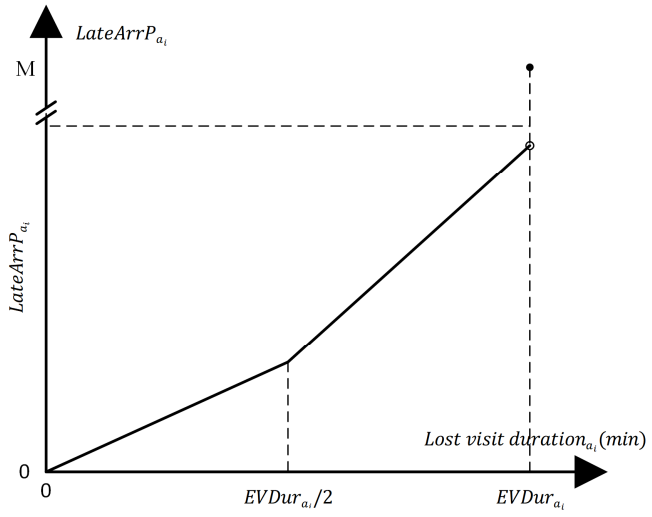


Figure A1. Relation between late arrival penalty and lost visit duration of node.

(5) Node penalty and itinerary penalty

We define the node penalty P_{a_i} in (A4), which equals the sum of all penalties generated within the node's time period, added to the physical fatigue penalty. The physical fatigue penalty is calculated on a daily basis, and it is added to the respective node penalty at each rest accommodation and at the end of the itinerary. Therefore, as shown in (A5), the itinerary penalty $P(I)$ equals the sum of all node penalties within the itinerary I. The itinerary penalty provides a comprehensive assessment of the deviation between the current itinerary planning and the expected scenario.

$$P_{a_i} = WaitP_{a_i} + LateP_{a_i} + \sum_{m_j \in a_i} MealDeviaP_{m_j} + \sum_{m_j \in a_i} MealReduP_{m_j} + \sum_{r_j \in a_i} RestDelaP_{r_j} + \sum_{r_j \in a_i} PhysFatiP_{r_j} \quad \text{(A4)}$$

$$P(I) = \sum_{a_i \in I} P_{a_i} \tag{A5}$$

In this algorithm, the penalty coefficients used are as shown in Table A1. These coefficients indicate the relative weights of each deviation and will influence the comparison of different itineraries as well as the time planning within the itinerary based on the minimum penalty principle.

Table A1. Penalty coefficient.

Type	*WaitP*	*LateArrP*		*MealDeviaP*	*MealReduP*	*RestDelaP*	*PhysFatiP*
Coeff.	0.5	Phase1: 1	Phase2: 2	0.5	1	1	1

Appendix C

The processing flow of the dining and accommodation arrangement algorithm for each node is as follows:

Step 1: Determine the known type of food and lodging arrangements required at the current time. If not specified, make a judgment based on the current time (occurring at the start of the itinerary and after any nodes where tourists have specified no food and lodging arrangements), and proceed to the next step.

Step 2: Determine whether the current node is a restaurant or hotel designated by the user. If not, proceed to the next step. If it is, handle it according to the selected restaurant or hotel, and move on to Step 6.

Step 3: Determine whether the current food and lodging arrangements need to occur within the current node's time period. If so, proceed to the next step; otherwise, exit.

Step 4: Determine if the next node in the itinerary is a designated a restaurant or hotel and supports the current dining and accommodation arrangement. The suitability of the selected node for the current dining and accommodation arrangement is based on two criteria: whether the node type supports the type of dining and accommodation, and whether postponing the dining and accommodation arrangement is acceptable. If both criteria are met, abandon the process; the dining and accommodation arrangement will move to the next node to avoid waiting or missing the time window of the next node. Otherwise, proceed to the next step.

Step 5: Make the dining or accommodation arrangement within the time period of this node; proceed to the next step.

Step 6: Determine whether the next dining and accommodation arrangement falls within the time period of the current node. If it does, proceed with recursion and return to Step 4. If not, exit the process. When a designated restaurant or hotel node triggers a self-call, it is because the expected time for the next dining or accommodation falls within the time interval $(N.EndT, N.NextNArrT]$. At this point, the node is no longer treated as a designated restaurant or hotel.

For selected restaurants or hotels, the handling method in step 2 is as follows:

(1) If waiting is required, schedule other dining and accommodation arrangements normally during the waiting time;
(2) If delayed but still accessible, adjust the schedule backward without affecting the next dining and accommodation arrangement, ensuring that the duration remains unchanged;
(3) If on time or inaccessible, no action is needed.

In step 5, the accommodation and dining arrangement method are based on the following assumptions: dining and accommodation can conveniently take place near scenic spots or along the route between nodes as needed. This aligns with people's habits and objective facts.

The main process is as follows: if the expected timing does not conflict with the time allocated for visiting the scenic spots, then the arrangement proceeds according to the

expectation. Otherwise, by calculating the penalty increment for each alternative, the arrangement selects the one with the smallest penalty increment for scheduling. Taking the dining time arrangement as an example, Figure A2 depicts the scenario in which the expected arrangement start time falls within different stages of the node time period.

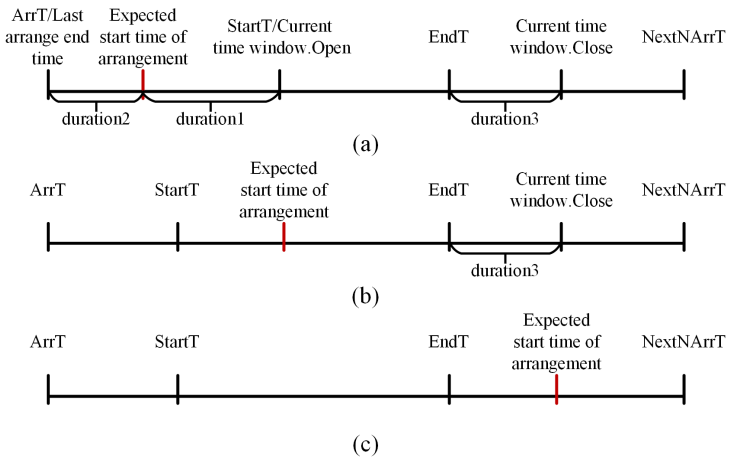


Figure A2. The scenario of expected start time of arrangement falling within different stages of the node time period: (**a**) before the visit; (**b**) during the visit; (**c**) after the visit.

Figure A2a depicts the expected start time for dining within the time interval $(N.ArrT, N.StartT]$. The dining time will be arranged sequentially, with priority given to duration1, followed by duration2, and then duration3, as depicted in Figure A2a. In other words, we will rationalize the use of time by advancing the dining time and postponing the sightseeing time without causing lateness. If the expected dining duration cannot still be met, conflicts will arise. In such cases, we will attempt alternative solutions.

Figure A2b depicts the expected start time for dining within the time interval $(N.StartT, N.EndT]$. If the node supports internal dining and the expected dining duration is not greater than duration3 as depicted in Figure A2b, there is no conflict. Otherwise, alternative arrangements will be considered.

Figure A2c depicts the expected start time for dining within the time interval $(N.EndT, N.NextNArrT]$. In this case, there will be no conflict.

There are three alternative options for the mealtime:

Option 1: Dining is arranged within the original time interval, aiming to meet the mealtime as closely as possible without sacrificing tour time, and any remaining missing mealtime is cancelled.

Option 2: Dining is scheduled after visiting the node; the mealtime is offset at this time, but the mealtime can be guaranteed (this option should be considered only when the offset is within a certain limit).

Option 3: This meal along with the existing meal and accommodation of this node are arranged according to the expected time. The node's time window is then re-selected after this meal, that is, considering backtracking within the node.

Under the setting of the priority of the scenic visit time, these alternatives will not sacrifice the scenic visit time. We choose the one with the smallest penalty increment (the total penalty of the current node) among the alternatives.

In the algorithm, with each dining and accommodation arrangement, the node's penalty and time period are updated, the dining and accommodation arrangement for the node is recorded, and the next type of dining and accommodation arrangement is determined.

For the arrangement of rest and accommodation time, similar to mealtime arrangements, but to ensure tourists receive regular and sufficient rest, rest and accommodation time will not be shifted forward or reduced. The impact of physical fatigue penalties is also considered. The specific process is not elaborated here.

References

1. Dai, X.F.; Yang, M.Y. Holistic Tourism Driving High-Quality Development of the Tourism Industry. *Tour. Trib.* **2022**, *37*, 6–8. [CrossRef]
2. Borràs, J.; Moreno, A.; Valls, A. Intelligent tourism recommender systems: A survey. *Expert. Syst. Appl.* **2014**, *41*, 7370–7389. [CrossRef]
3. Lim, K.H.; Chan, J.; Leckie, C.; Karunasekera, S. Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowl. Inf. Syst.* **2018**, *54*, 375–406. [CrossRef]
4. Halder, S.; Lim, K.H.; Chan, J.; Zhang, X. Efficient itinerary recommendation via personalized POI selection and pruning. *Knowl. Inf. Syst.* **2022**, *64*, 963–993. [CrossRef]
5. Chen, J.; Jiang, W.; Wu, J.; Li, K.; Li, K. Dynamic Personalized POI Sequence Recommendation with Fine-Grained Contexts. *ACM Trans. Internet Technol.* **2023**, *23*, 1–28. [CrossRef]
6. Taylor, K.; Lim, K.H.; Chan, J. Travel Itinerary Recommendations with Must-see Points-of-Interest. In Proceedings of the Eighth International Workshop on Location and the Web, Lyon, France, 23–27 April 2018.
7. Khamsing, N.; Chindaprasert, K.; Pitakaso, R.; Sirirak, W.; Theeraviriya, C. Modified ALNS Algorithm for a Processing Application of Family Tourist Route Planning: A Case Study of Buriram in Thailand. *Computation* **2021**, *9*, 23. [CrossRef]
8. Sarkar, J.L.; Majumder, A. gTour: Multiple itinerary recommendation engine for group of tourists. *Expert. Syst. Appl.* **2022**, *191*, 116190. [CrossRef]
9. Fu, C.Y.; Hu, M.C.; Lai, J.H.; Wang, H.; Wu, J.L. *TravelBuddy: Interactive Travel Route Recommendation with a Visual Scene Interface*; Springer International Publishing: Cham, Switzerland, 2014; pp. 219–230.
10. Zhang, C.Y.; Liang, H.W.; Wang, K. Trip Recommendation Meets Real-World Constraints: POI Availability, Diversity, and Traveling Time Uncertainty. *ACM Trans. Inf. Syst.* **2016**, *35*, 1–28. [CrossRef]
11. Rizwan, A.; Muslim, H.G.; Muna, A.; Mingwei, Z.; Abdullah, A.G.; Ahmed, A.B.A.; Taha, A.; Hussein, A.; Mizanur, R.S.M. A Serendipity-Oriented Personalized Trip Recommendation Model. *Electronics* **2022**, *11*, 1660. [CrossRef]
12. Chen, L.; Cao, J.; Tao, H.C.; Wu, J. Trip Reinforcement Recommendation with Graph-based Representation Learning. *ACM Trans. Knowl. Discov. Data* **2023**, *17*, 1–20. [CrossRef]
13. Zhou, X.; Su, M.Z.; Liu, Z.; Zhang, D. Smart Tour Route Planning Algorithm Based on Clustering Center Motive Iteration Search. *IEEE Access* **2019**, *7*, 185607–185633. [CrossRef]
14. Zhou, X.; Su, M.Z.; Feng, G.H.; Zhou, X.H. Intelligent Tourism Recommendation Algorithm based on Text Mining and MP Nerve Cell Model of Multivariate Transportation Modes. *IEEE Access* **2021**, *9*, 8121–8157. [CrossRef]
15. Zhang, Y.M.; Jiao, L.J.; Yu, Z.J.; Lin, Z.; Gan, M.J. A Tourism Route-Planning Approach Based on Comprehensive Attractiveness. *IEEE Access* **2020**, *8*, 39536–39547. [CrossRef]
16. Yochum, P.; Chang, L.; Gu, T.L.; Zhu, M.L. An Adaptive Genetic Algorithm for Personalized Itinerary Planning. *IEEE Access* **2020**, *8*, 88147–88157. [CrossRef]
17. Saeki, E.; Bao, S.; Takayama, T.; Togawa, N. Multi-Objective Trip Planning Based on Ant Colony Optimization Utilizing Trip Records. *IEEE Access* **2022**, *10*, 127825–127844. [CrossRef]
18. Ding, Y.; Zhang, L.; Huang, C.; Ge, R. Two-stage travel itinerary recommendation optimization model considering stochastic traffic time. *Expert. Syst. Appl.* **2024**, *237*, 121536. [CrossRef]
19. Gasmi, I.; Soui, M.; Barhoumi, K.; Abed, M. Recommendation rules to personalize itineraries for tourists in an unfamiliar city. *Appl. Soft Comput.* **2024**, *150*, 16. [CrossRef]
20. Zhou, X.; Li, S.H.; Gao, K.L.; Liu, C.L.; Wang, H.C.; Hu, J.W. An improved dynamic travel route planning algorithm with multiple constraints. *J. North Univ. China (Nat. Sci. Ed.)* **2019**, *40*, 57–62.
21. Xu, K. Research and Implementation of Smart Travel Route Planning Algorithm Based on Reinforcement Learning. Master's Thesis, Xidian University, Xi'an, China, 2020.
22. Huang, Z.B.; Lin, H.H.; Wang, J.H.; Deng, A.F.; Luo, B.R.; Lin, G.X. Research on improved ant colony algorithm under time window in travel route planning. *Sci. Technol. Innov. Appl.* **2019**, *29*, 28–29.
23. Lu, B.C.; Yang, J.Y.; Wang, X. City Suburban Tour Route Planning Based on Tourist Experience. *J. Chongqing Jiaotong Univ. Nat. Sci.* **2021**, *40*, 161–170.
24. Wu, X.B.; Guan, H.Z.; Han, Y.; Ma, J.Q. A tour route planning model for tourism experience utility maximization. *Adv. Mech. Eng.* **2017**, *9*, 2071942318. [CrossRef]
25. Werneck, H.; Silva, N.; Viana, M.; Pereira, A.C.M.; Mourão, F.; Rocha, L. Points of Interest recommendations: Methods, evaluation, and future directions. *Inform. Syst.* **2021**, *101*, 101789. [CrossRef]
26. Vansteenwegen, P.; Souffriau, W.; Oudheusden, D.V. The orienteering problem: A survey. *Eur. J. Oper. Res.* **2011**, *209*, 1–10. [CrossRef]
27. Blum, A.; Chawla, S.; Karger, D.R.; Lant, T.; Meyerson, A.; Minkoff, M. Approximation algorithms for orienteering and discounted-reward TSP. In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science Cambridge, Cambridge, MA, USA, 11–14 October 2003; pp. 46–55. [CrossRef]
28. Helsgaun, K. *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems: Technical Report*; Roskilde University: Roskilde, Denmark, 2017.

29. Helsgaun, K. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **2000**, *126*, 106–130. [CrossRef]
30. Coulom, R. *in Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 72–83.
31. Lim, K.H.; Chan, J.; Karunasekera, S.; Leckie, C. Personalized Itinerary Recommendation with Queuing Time Awareness. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017.
32. Kocsis, L.; Szepesvári, C. *Bandit Based Monte-Carlo Planning*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 282–293.