applied sciences

MDPI

Article

Application of Ant Colony Optimization Computing to a Recommended Travel Itinerary Planning System with Repeatedly Used Nodes

Shuo-Tsung Chen [1], Tsung-Hsien Wu [2,*], Ren-Jie Ye [3,*], Liang-Ching Lee [1], Wen-Yu Huang [1], Yi-Hong Lin [1] and Bo-Yao Wang [1]

[1] Department of Medical Informatics, Chung Shan Medical University, Taichung 40201, Taiwan; shough34@yahoo.com.tw (S.-T.C.); s1058016@gm.csmu.edu.tw (L.-C.L.); s1058011@gm.csmu.edu.tw (W.-Y.H.); s1058032@gm.csmu.edu.tw (Y.-H.L.); s1058020@gm.csmu.edu.tw (B.-Y.W.)
[2] Bachelor's Program in Business Management, Fu Jen Catholic University, New Taipei City 242062, Taiwan
[3] Graduate School of Applied Chinese Studies, National Yunlin University of Science and Technology, Yunlin 640301, Taiwan
* Correspondence: 083214@mail.fju.edu.tw (T.-H.W.); suntedfriend@yahoo.com.tw (R.-J.Y.)

Abstract: Recommended travel itinerary planning is an important issue in travel platforms or travel systems. Most research focuses on minimizing the time spent traveling between attractions or the cost of attractions. This study makes four contributions to recommended travel itinerary planning in travel platforms or travel systems. The first contribution is to consider recommended travel itinerary planning which can account for attractions, restaurants, and hotels at the same time. Due to the fact that restaurants and hotels can be repeated on the recommended itinerary, the second contribution is to propose an improved ant colony system (ACS) with repeatedly used nodes for the optimization of travel itinerary planning. In the third contribution, the proposed improved ACS allows repeated use of certain nodes without falling into a pattern of infinitely hovering within a certain interval or over certain nodes, through the interactive operation of a Watch List and a Tabu List. In the fourth contribution, the user satisfaction calculation for restaurants and hotels is also added to the travel itinerary planning in order to fully meet the needs of tourists. The experimental results verify the efficiency of the proposed improved ACS.

Keywords: travel itinerary planning; ant colony system (ACS); repeated used node; watch; tabu list

Citation: Chen, S.-T.; Wu, T.-H.; Ye, R.-J.; Lee, L.-C.; Huang, W.-Y.; Lin, Y.-H.; Wang, B.-Y. Application of Ant Colony Optimization Computing to a Recommended Travel Itinerary Planning System with Repeatedly Used Nodes. *Appl. Sci.* **2023**, *13*, 13221. https://doi.org/10.3390/app132413221

Academic Editor: Richard C. Millham

Received: 7 November 2023
Revised: 26 November 2023
Accepted: 28 November 2023
Published: 13 December 2023

1. Introduction

Generally, the tourist trip design problem refers to the problem of optimizing travel-route planning for tourists interested in visiting multiple points of interest (POIs) [1–15]. Today's travel is mainly self-planned, and the sources of information are relatives and friends or online information [1–3,5–14]. How to take into account the user's scenic satisfaction choices within a limited budget and within a limited period of time is what many tourism papers want to solve using an ant colony system (ACS). In [1], the proposed agent can recommend to the tourist a personalized travel route to enjoy Tainan City according to the tourist's requirements. It includes a context decision agent and a travel route recommendation agent. The context decision agent finds a suitable location distance, measures the context relation, and infers the context information based on the tourist's requirements and the travel ontology of Tainan City. The travel route recommendation agent is responsible for finding a personalized tour and plotting this travel route on Google Maps. Tseng et al. [3] adopted genetic algorithms and ant colony algorithms to carry out travel itinerary planning and compare the performance of the routes found using the two methods. In addition, to prove the practicality of the proposed algorithms, a travel itinerary planning website was set up. Yang et al. [5] proposed an approach which automatically

generates tourist itineraries by comprehensively considering transportation, lodgings, and POIs between/within each destination. In addition, they developed an approach based on the ant colony optimization (ACO) algorithm to solve the tourist planning problem due to the NP-complete nature of the itinerary planning. Zhang et al. [15] proposed to use modified mating operators and initialization genetic and ant algorithms to solve transport problems in tourism. The article analyzed modern methods of optimization of routes used to transport tourists between locations from the perspective of efficient use of resources. It performed this by analyzing the behavior of ant colonies, such as the ability to find the shortest route by providing mating pheromones, and features two solutions. In [14], Xu and You used the K-means algorithm and the ant colony optimization algorithm (ACO) to make a framework for solving the multi-constraint combination optimization problem and designing a real schedule for every feasible tourist route.

In order to solve the problem that the ant algorithm cannot complete travel schedule recommendations on its own—such as recommended travel itinerary planning including attractions, restaurants, and hotels with repeats of restaurants and hotels—this study proposes an improved ant colony system (ACS) with repeatedly used nodes for the optimization of travel itinerary planning. Moreover, the proposed improved ACS allows repeated use of certain nodes without falling into a pattern of infinitely hovering within a certain interval or over certain nodes, based on the interactive operation of a Watch and a Tabu List. The user satisfaction calculation for restaurants and hotels is also added to the travel itinerary planning in order to fully meet the needs of tourists. In other words, in addition to the path distance, the node's selection will also refer to tourist satisfaction with attractions, restaurants, and hotels.

The rest of this paper is organized as follows. Section 2 reviews background and preliminaries. Section 3 presents the proposed improved ACS. Section 4 explains the database and user-defined code. Section 5 shows the experimental results. Section 6 concludes our work.

2. Background and Preliminaries

The ant system (AS) was first published by Dorigo et al. (1996) [16] at an international seminar. It is an artificial intelligence heuristic algorithm for solving optimal problems. Its concept is based on the process by which ants effectively search for food in nature. When the ant faces an unknown path, as shown in Figure 1a, it will decide which path to follow based on the pheromones left on the path. If there is an obstacle, as in Figure 1b, the ants from A to E and E to A will randomly select one side of the obstacle (F or C) at points B and D, respectively. Normally, pheromones will gradually disappear over time. In Figure 1c, since the F path is long and the pheromone is lighter, the number of ants selecting path F gradually decreases. Path C is shorter and the pheromones are stronger, so more ants go this way and continue to increase the pheromones, and finally almost all ants choose this route. Dorigo et al. mentioned that ants in ant systems have the following characteristics:

1. They have memories.
2. They are not completely blind (the length of the path can be perceived).
3. The time in the system is intermittent.

Moreover, the probability of the kth ant from the node i to the node j at time t is $P_{ij}^{k}(t)$ formulated as

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}}{\sum_{k \in J_k(i)} [\tau_{ik}(t)]^{\alpha} \times [\eta_{ik}(t)]^{\beta}}, & if \ \ j \in J_k(i) \\ 0, & otherwise \end{cases} \quad (1)$$

where

$\tau_{ij}(t)$: pheromone intensity on the path $\overline{ij}$ at time t;
$J_k(i)$: a set of neighboring nodes that have not been visited by ant k at node I;
d_{ij}: length of path ij;

η_{ij}: $\eta_{ij} = 1/d_{ij}$;

α, β: adjustment factors which are used to determine the relative importance of pheromones and distance.

In Equation (1), pheromone is updated by the following formula:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \ 0 < \rho < 1 \tag{2}$$

where $\rho \in [0,1]$ is the volatility coefficient of pheromones and

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k, & \text{if the } k\text{th ant traverses } (i,j) \\ 0, & \text{others} \end{cases},$$

where Q is a parameter; L_k is the length of the path which the kth ant walks along.

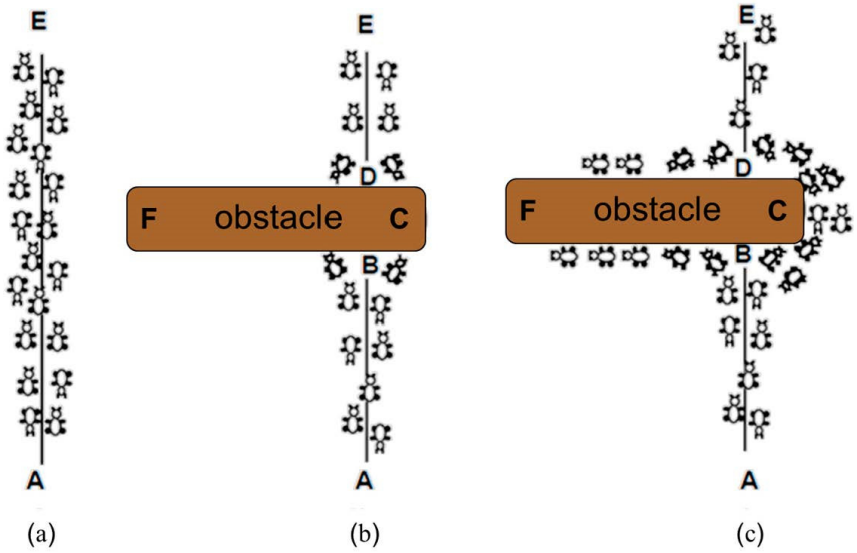

Figure 1. Ant behavior. In (**a**), it will decide which path to follow based on the pheromones left on the path. If there is an obstacle, as in (**b**), the ants from A to E and E to A will randomly select one side of the obstacle (F or C) at points B and D, respectively. Normally, pheromones will gradually disappear over time. In (**c**), since the F path is long and the pheromone is lighter, the number of ants selecting path F gradually decreases.

However, due to efficiency issues, AS in Equation (1) will not be able to find a feasible solution when dealing with TSPs in more than 30 cities. In 1997, Dorigor and Gambardella [17,18] improved the AS in Equation (1) to the ant colony system (ACS), which is described in the following three points:

(i) Add the State Transition Rule

When the kth ant goes to a certain node, users define a threshold q_0 and the ACS will first generate a random number q. If $q \leq q_0$, $P_{ij}^{k}(t)$ is the maximum value of pheromone multiplied by the path coefficient; if $q > q_0$, the formula of $P_{ij}^{k}(t)$ is returned to Equation (1).

$$P_{ij}^{k}(t) = \begin{cases} \arg\max\limits_{u \in J_k(i)} \left\{ [\tau(r,u)]^{\alpha} \times [\eta(r,u)]^{\beta} \right\}, \text{ if } q \leq q_0 \\ \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}}{\sum\limits_{k \in J_k(i)} [\tau_{ik}(t)]^{\alpha} \times [\eta_{ik}(t)]^{\beta}}, & if\ j \in J_k(i) \\ 0, & \text{otherwise} \end{cases}, \text{ otherwise} \end{cases} \tag{3}$$

where q_0 is a given threshold between 0 and 1; q is a random number.

(ii) Improve the Global Update Rule

The pheromone in Equation (4) is updated by the following equation:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}\ ,\ 0 < \rho < 1 \tag{4}$$

(iii) Add the Local Update Rule

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \tau_0 \tag{5}$$

In addition, the ACS has joined the concept of optimizing parameters [19,20] and local search as follows [21,22]. After the ants have completed a complete path, that is, after performing an iteration, the order of the random exchange paths is passed, and then it is judged whether it is better than the solution before the exchange. If the solution after the exchange is better than the solution before the exchange, the exchanged solution replaces the pre-exchange solution. This concept speeds up the efficiency of finding the best solution and can also escape the limitation of the optimal solution of the region.

3. The Proposed Improved ACS

In the improved ant algorithm for repeatable nodes proposed by us, the nodes that ants can visit are divided into two points: the Hard node and the Soft node. A Hard node represents a node with a limited number of times, and a Soft node is a node that can be visited repeatedly. Through these two types of nodes and using Watch to control the movement of ants, the ants will not form an infinite loop on the same path or node. Moreover, using the improved ACS of Watch will allow the ant algorithm to solve more diversified scheduling problems, especially the travel schedule recommendation problem that the ant algorithm cannot complete alone.

When using the ACS for travel itinerary planning, the attractions, restaurants, and hotel information in the database will be used to create a full connected graph for the ACS to connect. At this time, the ant, as a trial staff, records the journey while moving various nodes, such as play time, transportation time, restaurant and hotel satisfaction, etc. In addition to trying out different travel schedules for each traveler, these ants will also recommend their restaurants and hotels.

Then, the entire travel process is carried out under the control of parameters, Watch, and step through user-defined code (UDC). Each ant will generate a journey schedule according to the rules (different from the rules of traditional ants). Users can freely control the number of days of play to produce results that meet user satisfaction and choose from each trial player's journey.

At the end of the entire ACS, the ant with the highest satisfaction and the most time-saving transportation route will be launched to generate the best travel sequence, and the itinerary recorded by this ant will be output for user reference.

Figure 2 shows the flowchart of the proposed improved ACS which is described in detail by the following subsections.

3.1. Initialize Parameters and Build Map

At this step, the attractions, restaurants, and hotels in the database will be jointly built into a full connected map that can be used by the ACS. However, in some cases, some routes between nodes are not allowed to pass. For example, it does not make much sense to connect a hotel with a hotel and a restaurant with a restaurant so we will not consider these two situations.

3.2. Sprinkle Ants

The traditional ACS sprinkles ants on each node evenly. However, if all the ants are evenly distributed on each node in the case of tourism, some ants will start from the attractions and pass through the starting point on the way, and some ants start from the restaurants. These travel situations are unreasonable. Therefore, this study assumes that all

ants were sprinkled on the starting point and proceeding from the starting point because of the condition that "tourists must start from the starting point". In addition, there is no need to sprinkle ants after iterations since the ant comes back to the starting node. Instead, we just need to delete the ants' memory.

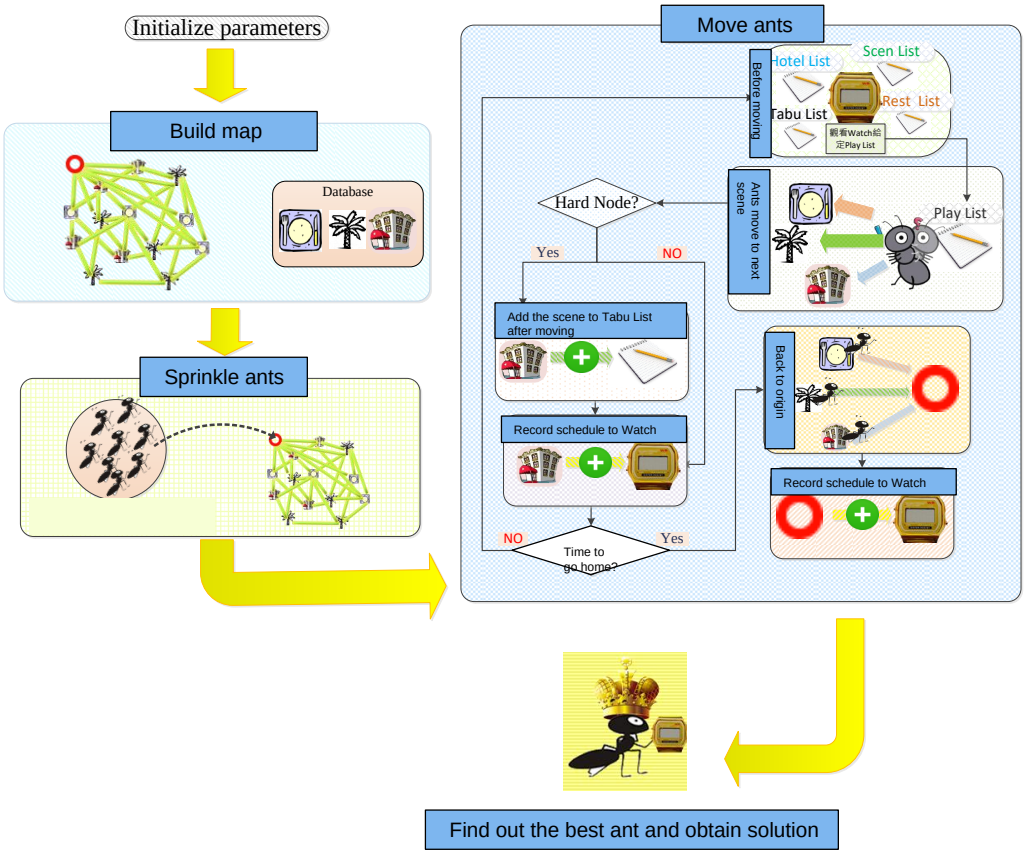

Figure 2. Flowchart of the proposed improved ACS.

3.3. Move Ants

When each ant is looking for the next node, it will generate a different Play List according to the proposed rules, so that the ants will know which nodes they can go to. The ant will pick from the Play List and will select the next node according to the transition rule in Formula (3).

3.4. Ant Algorithm with Repeatable Nodes

If the feature of repeatable nodes is added to the traditional ant algorithm, it will make the ants choose to stay in a place to obtain the shortest path or repeat a certain long-distance path to obtain the longest path. Both of these situations deviate from situations where multiple visits were intended to be optimally ordered. Therefore, this study uses the combination of Tabu List and Watch-controlled Play List to propose that the ant algorithm can repeat nodes without hovering in the same interval indefinitely. That is, each ant of the proposed ant algorithm will have: a Watch that monitors it specifically, a Tabu List of nodes that are not recorded, and a Play List given by the Watch. We envision each ant as a tourist. These virtual travelers will replace real users in trying out attractions, eating in every restaurant, and staying in every hotel. During the trial, each ant will move according to the Play List, and the Play List will be affected by the Watch and Tabu List. In Figure 3, details are listed as follows.

(1) Watch

The Watch will be responsible for recording the time that each ant arrives at each node, and it is divided into four time rules according to the difference of Watch time, and then capture the Scene List (attractions), Rest List (restaurants), and Hotel List (hotels) to generate a Play List by filtering of a Tabu List. Each ant will update its watch time according to the traffic time when it reaches the next node.

Therefore, in addition to the origin, each node visited by the ant will have two record Watch times, namely arrival time and departure time. The origin is either the starting point or the ending point, so an ant cannot have both arrival and departure times at the origin. If the number of nodes (n) that the ant needs to visit is n, then the Watch time recorded by Watch is $2n - 2$.

(2) Play List

As the ants walk different paths, the attractions visited by each ant may be different. Even if all the ants move one step, the time to move to each attraction and the recommended play time for each attraction are not necessarily the same, resulting in the travel sequence and situation of each ant also being different, and the contents of the generated Play List will not be the same.

(3) Tabu List

The node representatives put into the Tabu List will never be visited again, and the rules of recycling can also be determined according to the situation. For example, when designing a travel itinerary, each tourist attraction will only be visited once, so whenever an ant visits a tourist attraction, the tourist attraction will be recycled to the Tabu List. The next time the ant's Play List is brought into the Scene List again, the attraction will not enter the Play List for ant selection because of the filtering of the Tabu List. Restaurant and hotel nodes may be visited repeatedly, so even if they have been visited before, they do not need to be included in the Tabu List, and because of the open specific node feature given by Watch, ants will not repeatedly visit the same restaurant or hotel.

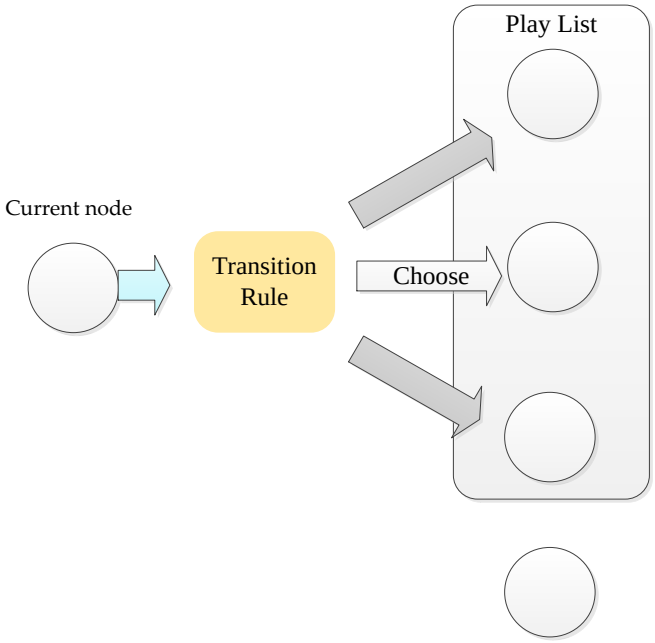

Figure 3. During the trial, each ant will move according to the Play List, and the Play List will be affected by Watch and Tabu List.

For the timing rules, lunch time is set from 11 a.m. to 1 p.m. and dinner time is from 6 to 8 p.m. During these two periods, the attractions brought by ant's Play List are the Rest

List, and each meal time is 1.5 h. Breakfast time is usually eaten at the accommodation, so breakfast time will be combined with the rest time of the accommodation. The accommodation time is set after 9 p.m., the attractions brought in at this time are the Hotel List, and the given rest time is 9 h (including personal time, sleep, and breakfast time). Outside of these three periods, it is determined that the ants will go to the tourist attractions, and the time spent on the attractions can be learned from the respective recommended tourist attractions.

As a result, we conclude the proposed ant algorithm with repeatable nodes in five steps.

Step 1. Watch to see the current time of the ants and select list:

Based on the current time rules of the ant, choose one from the Scene List, the Rest List, and the Hotel List, and then take out the nodes in the list as preparations for inclusion in the Play List.

Step 2. Watch to view the ant's Tabu List and perform node filtering:

The current Tabu List of the ant learns which nodes will not be visited again and deletes these nodes from Step 1.

Step 3. Watch outputs the list of attractions to the ant as Play List:

The nodes that meet the current time rule and do not exist in the Tabu List are included in the Play List, which are the nodes that the ants can visit at the moment.

Step 4. The ant selects the attractions from the Play List and goes to these attractions:

The ant learns the accessible nodes from the Play List and selects the next node to visit according to the transition rule.

Step 5. The ant records and changes the Watch time:

After arriving at the node, the ant will change the Watch time for the first time according to the traffic time, and then change the Watch time for the second time based on the time shown by each node as the time to leave the node. When the ant is ready to leave the current node, the representative ant will make the next node selection, so it will use the updated Watch time to return to Step 1 and start the node selection again.

In addition, when the ants visit the node, the problem is that the ant simply skips the meal time and proceeds to the next node because the traffic distance or play time exceeds the meal segment. Therefore, this study is designed. When Watch is playing a Play List for ants, Watch selects the Scene List and includes it in the Play List, and Watch will consider whether going to and visiting various attractions will cause ants to skip the dining area and not eat. Attractions that make ants skip the dining area will not enter the Play List. In other words, ants going to and visiting the attractions in the Play List will not skip the dining area. If the Watch finds that each of the attractions that the ant can walk to will make the ant exceed the meal time, the Watch will directly bring the Rest List into the Play List, which means that the ant is forced to eat first, so that it can avoid the situation of continuing to the next trip without eating.

3.5. Fitness Value for Evaluation for Optimization

As mentioned above, the ants in this study will visit various nodes according to different situations during the entire movement process. In addition to the path distance, the node's selection will also refer to the satisfaction with attractions, restaurants, and hotels. The conditions included in attraction satisfaction (A.S.), restaurant satisfaction (R.S.), and hotel satisfaction (H.S.) are shown in Table 1.

Attraction satisfaction, restaurant satisfaction, and hotel satisfaction can be expressed by the following formulas:

$$A.S. = A.T. * u_1 + A.P. * u_2 - A.C. * u_3 \quad (6)$$

$$R.S. = R.T * u_4 + R.P. * u_5 - R.C. * u_6 - W.S. * u_7 \quad (7)$$

$$H.S. = H.T. * u_8 + H.P. * u_9 - T.C. * u_{10} \quad (8)$$

Among them, u_1 represents the weight of interest; u_2 represents the user's interest in popular attractions; u_3 represents the user's degree of interest in spending on attractions; u_4 and u_8 represent the user's restaurant and hotel interest weights, respectively. u_5 and u_9 indicate whether users like the weight of popular restaurants or restaurants; u_6 and u_{10} indicate whether they care about excessively expensive restaurant or restaurant weights; u_7 represents how much users care about waiting time.

Table 1. Conditions included in attraction satisfaction.

A.S., Attraction Satisfaction	R.S., Restaurant Satisfaction	H.S., Hotel Satisfaction
1. A.T., attraction type: 0 or more attractions A.P, attraction popularity: 1~10 (The bigger the number, the more popular it is). 2. A.C., attraction cost: 0 or more	1. R.T., restaurant type: 0 or more meals 2. R.P., restaurant popularity: 1~10 (The bigger the number, the more popular it is). 3. W.S., waiting score: 0~10 (The higher the number, the longer the wait). 4. R.C., restaurant cost: 0 or more	1. H.T., hotel type: 0 or more stays 2. H.P., hotel popularity: 1~10 (The bigger the number, the more popular it is). 3. H.C., hotel cost: 0 or more

After each ant completes its entire journey, it will have its own path length, denoted as P.L. In traditional TSP, this path length (or P.L.) is the ant's fitness value. However, in this study, the ant will deduct the calculated A.S., H.S., and R.S. from P.L., as shown in the following Formula (9). Even if the total journey distance is the same, itineraries with high-satisfaction restaurants and hotels will have a lower fitness value,

$$\text{Fitness Value} = P.L. - (R.S. + H.S. + A.S.) \tag{9}$$

Accordingly, we have the following optimization problem in the mathematical form included with constraints:

$$\text{Minimize } P.L. - (R.S. + H.S. + A.S.)$$

Subject to

$$A.S. = A.T. * u_1 + A.P. * u_2 - A.C. * u_3$$

$$R.S. = R.T * u_4 + R.P. * u_5 - R.C. * u_6 - W.S. * u_7$$

$$H.S. = H.T. * u_8 + H.P. * u_9 - T.C. * u_{10}$$

In the end, the proposed ACS will select an ant with the best fitness and list the time recorded in the Watch of the ant and all the travel sequences and become the recommended itinerary for this system.

4. Database and User-Defined Code

4.1. Database Fields and Data Description

The data used in this study are all virtual nodes as shown in Figure 4. There are a total of 75 node data, of which Node is the aggregate of all nodes, and the nodes are classified into three types of tables: attractions, hotels, and restaurants. The value "1" represents one-to-many. The data content of these three tables is recorded separately. For information about attractions, hotels, and restaurants, the amount of data on attractions needs to be the same as the number of attractions set by the user in the parameters, so the amount of data on attractions can be regarded as the number of attractions that users want to visit. The number of hotels is 10, and there are 13 restaurants to choose from. The last is the dist table, which contains the distance (km) between each attraction, and the distances between nodes are generated randomly. The traffic distance of the attractions is also recorded at the same time, so the amount of data is not a fixed value.

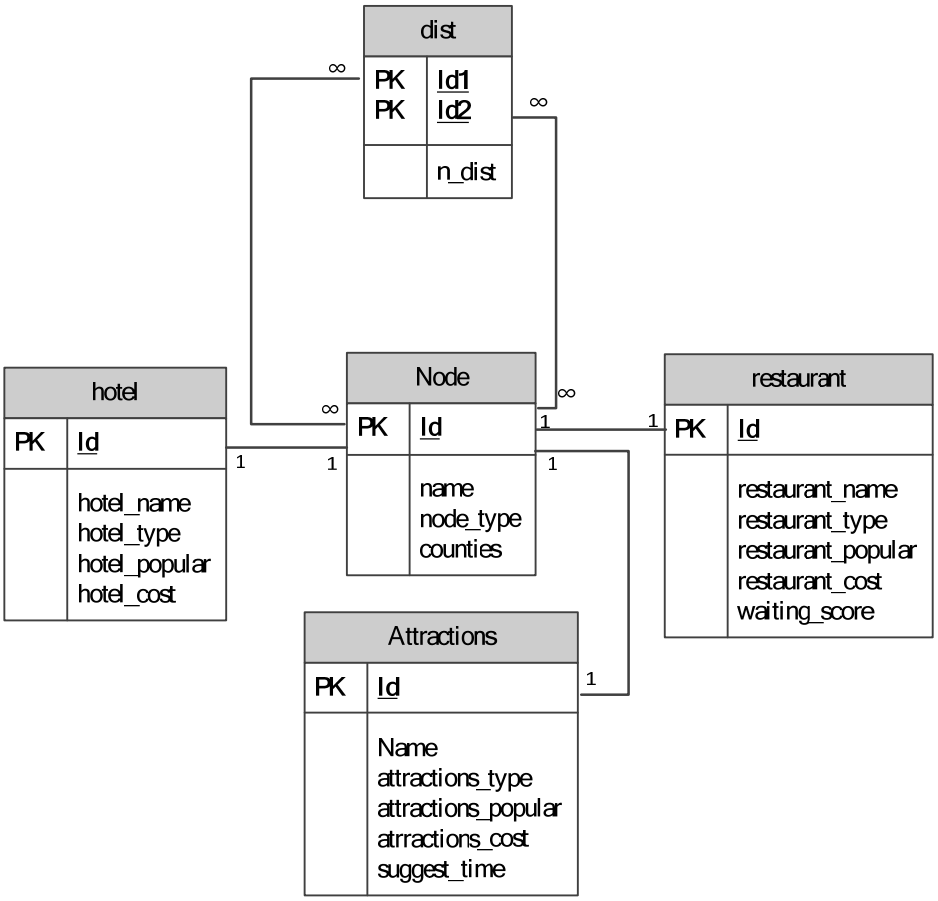

Figure 4. Relational Model.

1. Node

Space name	Data type	Allowed null	Description
Id	integer	N	PK, label of each node
name	nchar (10)	N	name of each node
node_type	nchar (10)	N	"ST": departure, "A": attraction "H": hotel, "R": restaurant
counties	nchar (10)		the county where the attraction is located

2. Attractions

Space name	Data type	Allowed null	Description
Id	integer	N	PK, label of each node
name	nchar (10)	N	name of each attraction
attractions_type	nchar (10)	N	attraction types
attractions_popular	integer	N	popularity of attraction, 1~10
atrractions_cost	integer	N	spend on attraction
suggest_time	integer	N	recommended play time (min)

3. Hotel

Space name	Data type	Allowed null	Description
Id	integer	N	PK, label of each node
hotel_name	nchar (10)	N	name of each hotel
hotel_type	nchar (10)	N	hotel types
hotel_popular	integer	N	popularity of hotel, 1~10
hotel_cost	integer	N	hotel fee

4. Restaurant

Space name	Data type	Allowed null	Description
Id	integer	N	PK, label of each restaurant
restaurant_name	nchar (10)	N	name of each restaurant
restaurant_type	nchar (10)	N	restaurant types
restaurant_popular	integer	N	popularity of restaurant, 1~10
restaurant_cost	integer	N	spend on restaurant
waiting_score	integer	N	waiting time (min)

5. Dist

Space name	Data type	Allowed null	Description
Id1	integer	N	PK1, label of departure node
Id2	integer	N	PK2, label of end node
n_dist	float	N	distance between two nodes

4.2. User-Defined Code and Parameter Description

First, the main custom parameters are explained as follows.

- numCities—number of cities;
- numAnts—number of ants;
- numAttractions—number of attractions;
- numHotel—number of hotels;
- numRestaurant—number of restaurants;
- realnum—actual number of nodes;
- start_d—departure date;
- start_h—departure time;
- start_m—departure time.

numCities represents the number of nodes that the ant actually needs to visit, and the value of numAnts will be the same as numCities. numAttractions represents the number of attractions to visit, numHotel represents the number of accommodations, numRestaurant is the number of meals, and the number of these three will directly affect the value of numCities, because the ant's travel itinerary is determined by the above three attractions, hotels, restaurants, and starting and ending points, so the value of numCities is actually the sum of numAcctrations, numRestaurant, numHotel, and starting and ending points. The start_d, start_h, and start_m are the starting times set by the user.

In addition, we need to mention the difference between realnum and numCities. In the general ant algorithm, the number of nodes that the ant actually walks to is the same as the number of cities. In the ant algorithm of repeatable nodes in this study, the ant will have nodes that have not been visited during a trip. Because restaurants and hotels allow repetition, ants who walk to enough restaurants or hotel nodes will not deliberately go to the nodes that they have not walked through. Therefore, numCities is not equal to realnum.

Therefore, realnum is required to confirm the actual number of nodes. The distance array between nodes is updated when performing steps such as Initialize.

Next, we introduce user-defined code (UDC) such as Initialize, Watch, Step, and Evaluation. The operation relationship of these four UDCs is shown in Figure 5. The steps and Evaluation on the pink background represent that the system has built-in code rules for users to use, while the Initialize and Watch on the blue background must be written by users. Each part and each UDC will be explained one by one later.

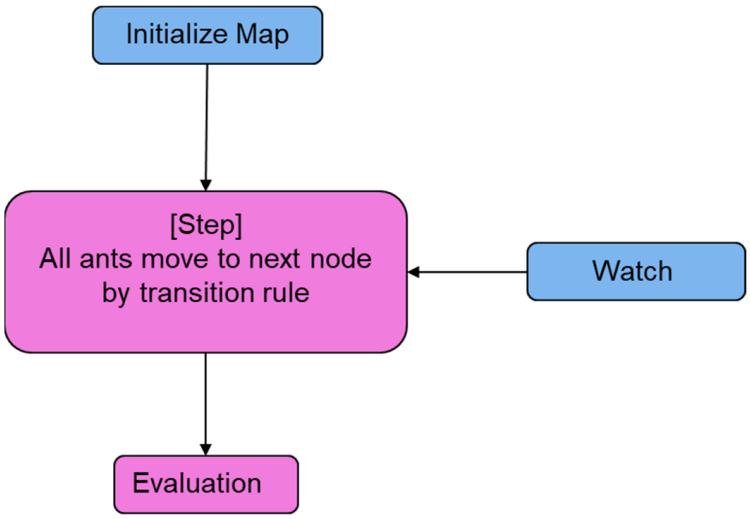

Figure 5. The operation relationship of the four UDCs.

(1) public override void Initialize (ref double[,] Dists, int numCities)

- dists[,]: 2D distance array between nodes;
- numCities: number of ants.

The UDC, Initialize, must initialize the path distance between nodes. In addition to initializing the traffic distance of each node, Initialize also needs to be responsible for initializing various lists and starting Watch and other tasks.

Each node in the ant's journey will take a different amount of time, so an additional node time array of int type needs to be recorded so that the ant can view this array at any time while walking. Initialize will grab the type of node from the database and store the node time according to each type.

(2) private ArrayList Watch(int AntIndex, int step)

- AntIndex: ant labeling, labeling from zero;
- step: the ant is going to a node.

Watch is one of the core contents of the study. Watch in the program is mainly divided into two parts. One is used to generate a Play List that drives ants according to the user's rules. The other part is responsible for recording the ant playing time. Watch time and this UDC is responsible for the former function.

Just as for the watch time operation rule, this UDC will catch each ant and check the Watch time and Tabu List on them. One that matches the ant time point from the Scene List, Restaurant List, and Hotel List is selected and excluded. The nodes in the Tabu List become the ant's Play List. In addition to processing time, compared to the Tabu List that is always excluded from the Play List, one can also perform special processing on nodes. For example, after the total number of visits to node A exceeds 2, one must have visited each of B, C, and D. Node allows node A to appear again.

And at the end of this UDC, an Array List named validLoc will be returned to Step as a Play List for ant reference.

(3) public override int Step(int AntIndex, int step, ref int[] Ant, ref bool[] Visited, ref double[,] dists, ref double[,] Pheromones, int numCities, int iteration, ref double[,] Attractive)

- AntIndex: ant labeling, labeling from zero
- Step: the ant is going to a node
- Ant[]: record ants' visiting nodes from zero
- Visited[]: record ants' visiting nodes from zero
- dists[,]: path distance between cities
- Pheromones[,]: city-to-city pheromone intensity
- numCities: number of cities
- Attractive: attractive matrix

The Step is used to determine the next node to be moved to by ants, which is one of the cores of the ant algorithm and this research. In this UDC, the validLoc of the previous Watch is received and becomes the Play List and drives the ant to walk. The UDC of Step has another part of the Watch, which is to record the watch time of each ant. The Step will select all the next nodes that the ant can walk to according to the built-in parameter q0 of the ACS, which can be divided into direct selection of the most attractive node and roulette method selection. At the end, the number of a node is returned to the main program of the ACS, allowing the ants to formally move. If the last node selected by the ant is a scenic spot, as the scenic spots in this study are all Hard nodes, which is a one-time scenic spot, they will be added to the Tabu List after a visit. After Step returns the selection result to the ACS node, the ant also uses AntWatch to record the arrival time and end time of the node, and pushes the Watch time, which again affects the next Watch for this Play List selected by the ant.

(4) public override double Evaluation(int AntIndex, ref int[] AntTrail, ref double[,] dist, int numCities, int iteration)

- AntIndex: ant's index (from 0)
- AntTrail []: ant's node path (from 0)
- dists[,]: path distance between cities
- numCities: number of cities

In the traditional ant algorithm, Evaluation is a UDC that calculates whether each ant's travel is superior or not. In the TSP problem, it calculates the path length of the entire journey. The recommendation of the travel itinerary in this study is similar to the TSP problem. Therefore, the evaluation of an ant's journey is based on the length of the ant's path.

The difference from the TSP problem is that restaurant and hotel satisfaction needs to be calculated in Evaluation. This is because, in the general TSP problem, the shorter path is considered as the better itinerary. Therefore, if the calculated restaurant and hotel satisfaction is deducted from the path length, one can choose the final ant selection of restaurants and hotels with higher satisfaction itineraries and shorter travel itineraries.

5. Experimental Results and Discussion

This section presents the scheduling results of the proposed ACS improved for travel recommendations. The termination condition is set to 50 generations, and the limit execution time is 1800 s or 30 min. The smaller the adaptive value, the better. The travel days of the travel itinerary are 1 day, 3 days, and 7 days, and the relevant parameters of ACO are set as follows:

- α = 0.5 (pheromone coefficient in transition rule);
- β = 1 (path factor in transition rule);
- $\tau_{ij}(0)$ = 0.00001 (initial value of pheromone on the path);
- q_0 = 0.0 (full selection of the highest attractive nodes);
- ρ_G = 0.01 (pheromone volatility of global updating rule);
- ρ_L = 0.01 (pheromone volatility of local updating rule).

In addition, the weights of user restaurant and hotel satisfaction are set as follows:

- Type of diet: 0 (no effect);
- Hot recommendations for restaurants: 10 (very interested);
- Restaurant average wait score: 1 (uninterested);
- Care for restaurant costs: 5 (not too expensive);
- Type of accommodation: 0 (no effect);
- Hotel popularity: 1 (uninterested).

5.1. One-Day Tour Schedule

The scheduled departure time is 9:00 in the morning, visiting 4 attractions (I, AA, AU, AW), 2 meals, no accommodation, and 50 generations of evolution. Figure 6 shows the search history of fitness value. From Table 2 and Figure 6, it can be seen that there is not much difference in the itinerary between the first and last generations, and the schedule of the first generation arrives at 11:17 p.m., while the last generation is at 9:58 p.m., just arrived home.

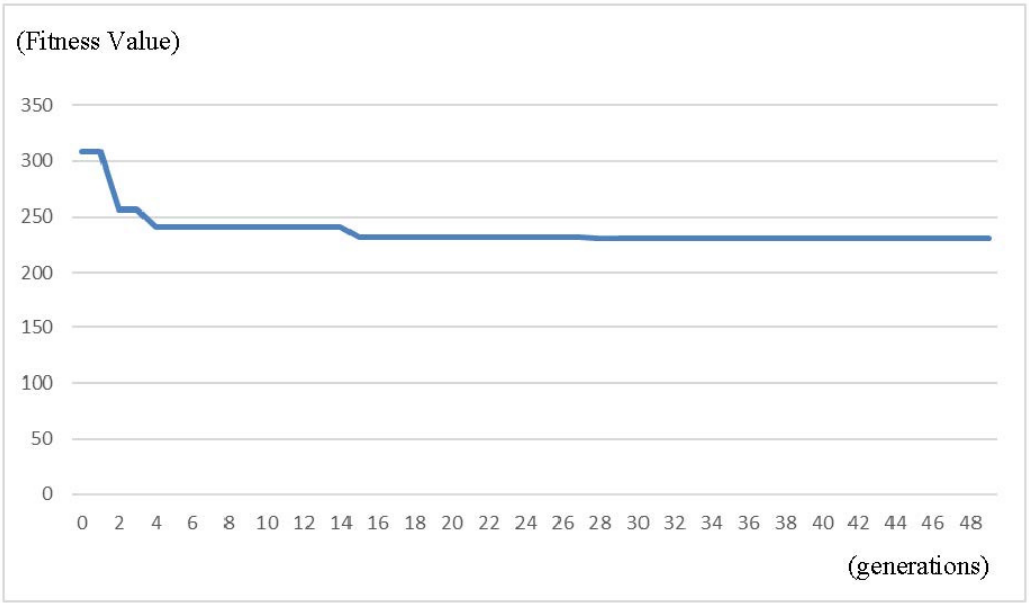

Figure 6. Evolution history of fitness value of one-day travel itinerary.

Table 2. The results of one-day schedule.

Scheduling Results of the 1st Generation
Fitness Value:
09:00 Departure
09:29 Arrive at attraction AA
10:29 Depart from attraction AA
10:57 Arrive at attraction AU
12:07 Depart from attraction AU
12:20 Arrive at piccolo store 6
Lunch time
13:50 Depart from piccolo store 6
16:25 Arrive at attraction AW
17:15 Depart from attraction AW
18:26 Arrive at attraction I
19:26 Depart from attraction I
20:27 Arrive at piccolo store C
Dinner time

Table 2. *Cont.*

21:57 Depart from piccolo store C
23:17 Home
Scheduling results for the last generation
Fitness value:
09:00 Departure
10:18 Arrive at attraction I
11:18 Depart from attraction I
11:49 Arrive at piccolo store 2
Lunch time
13:19 Depart from piccolo store 2
14:03 Arrive at attraction AW
14:53 Depart from attraction AW
17:21 Arrive at attraction AW
18:31 Depart from attraction AW
18:44 Arrive at piccolo store 6
Dinner time
20:29 Arrive at attraction AA
21:29 Depart from attraction AA
21:58 Home

5.2. Three-Day Tour Schedule

The scheduled departure time is 3:00 in the afternoon, visiting 10 attractions (A, B, C, D, E, F, G, H, I, J), 5 meals, 2 accommodations, and 50 generations of evolution. Figure 7 shows the search history of fitness value. Among the three-day schedule results, as shown in Table 3, the first-generation itinerary of the second day did not arrive at the hotel in the early morning on the third day due to the long route, and the last generation stayed on the second day at a normal time. On the last day, the first generation arrived home at 10:30 p.m., while the last generation's itinerary arrived home at 7:37 p.m., a difference of almost 3 h.

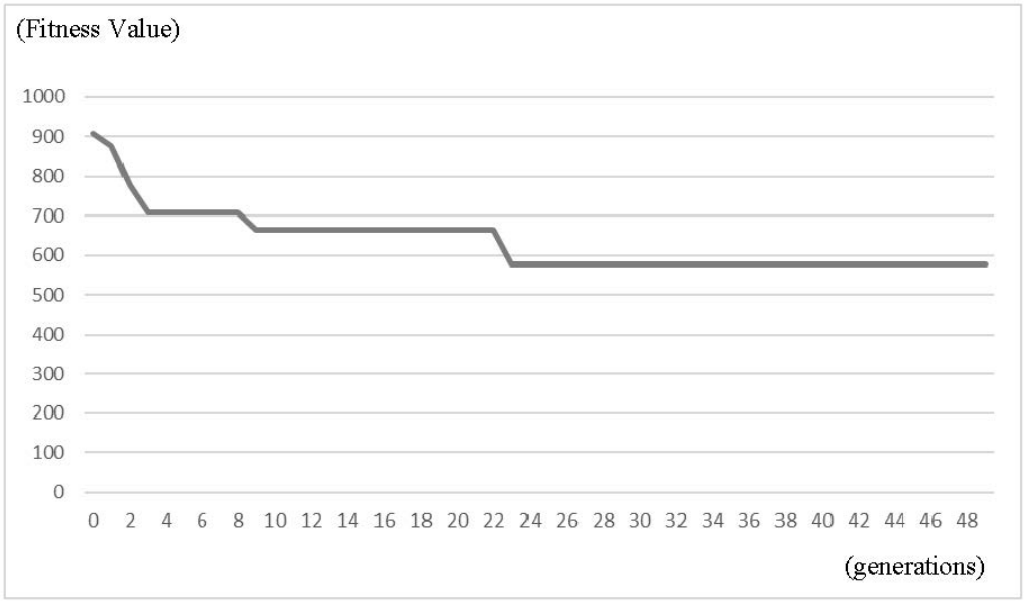

Figure 7. Evolution history of fitness value of three-day travel itinerary.

Table 3. The results of three-day schedule.

Scheduling Results of the 1st Generation
Fitness Value:
1st day
15:00 Departure
16:36 Arrive at attraction J
19:36 Depart from attraction J
20:22 Arrive at piccolo store C
Dinner time
21:52 Depart from piccolo store C
22:25 Arrive at hotel A
Sleep time
2nd day
07:25 Depart from hotel A
07:45 Arrive at attraction D
09:45 Depart from attraction D
10:15 Arrive at attraction B
12:45 Depart from attraction B
14:05 Arrive at piccolo store B
Lunch time
15:35 Depart from piccolo store B
15:35 Arrive at attraction A
16:50 Depart from attraction A
16:52 Arrive at attraction C
17:52 Depart from attraction C
17:52 Arrive at attraction G
20:52 Depart from attraction G
22:31 Arrive at piccolo store C
Dinner time
00:01 Depart from piccolo store C
00:34 Arrive at hotel A
Sleep time
3rd day
09:34 Depart from hotel A
09:37 Arrive at attraction E
11:37 Depart from attraction E
12:42 Arrive at piccolo store C
Lunch time
14:12 Depart from piccolo store C
14:49 Arrive at attraction F
15:19 Depart from attraction F
16:11 Arrive at attraction H
18:11 Depart from attraction H
18:18 Arrive at piccolo store C
Dinner time
19:48 Depart from piccolo store C
21:13 Arrive at attraction I
22:13 Depart from attraction I
22:30 Home
Scheduling results for the last generation
Fitness value:
1st day
15:00 Departure
15:17 Arrive at attraction G
18:17 Depart from attraction G
18:59 Arrive at piccolo store 10
Dinner time
20:29 Depart from piccolo store 10
20:33 Arrive at attraction N
21:03 Depart from attraction N

Table 3. *Cont.*

22:44 Arrive at hotel B
Sleep time
2nd day
07:44 Depart from hotel B
08:46 Arrive at attraction S
09:46 Depart from attraction S
11:30 Arrive at attraction C
12:30 Depart from attraction C
12:31 Arrive at piccolo store C
Lunch time
14:01 Depart from piccolo store C
14:17 Arrive at attraction J
17:17 Depart from attraction J
17:21 Arrive at attraction E
19:21 Depart from attraction E
19:55 Arrive at piccolo store A
Dinner time
21:25 Depart from piccolo store A
22:05 Arrive at hotel A
Sleep time
3rd day
07:05 Depart from hotel A
07:45 Arrive at attraction AV
08:05 Depart from attraction AV
09:59 Arrive at attraction AE
10:59 Depart from attraction AE
11:17 Arrive at attraction A
12:32 Depart from attraction A
12:44 Arrive at piccolo store B
Lunch time
14:14 Depart from piccolo store B
15:05 Arrive at attraction AL
16:25 Depart from attraction AL
17:15 Arrive at piccolo store 1
Dinner time
18:45 Depart from piccolo store 1
19:37 Home

5.3. Seven-Day Tour Schedule

The scheduled departure time is 9:00 in the morning, visiting 25 attractions (A,B,. . .), 14 meals, 6 accommodations, and 29 generations of evolution. Figure 8 shows the search history of fitness value. In the seventh day's schedule, as shown in Table 4, the first and last generations have the same difference in the one-day and three-day itineraries. On the last day, the first-generation itinerary was too long, so instead of going home on time on the seventh day, they had to stay for an extra day. The last-generation itinerary can be seen to be on time on the seventh day, arriving at 9:43 p.m.

Table 4. The results of seven-day schedule.

Scheduling Results of the 1st Generation
Fitness Value:
1st day
09:00 Departure
09:12 Arrive at attraction A
10:27 Depart from attraction A
11:01 Arrive at attraction P
11:51 Depart from attraction P

Table 4. *Cont.*

12:37 Arrive at piccolo store C
Lunch time
14:07 Depart from piccolo store C
15:30 Arrive at attraction AW
16:20 Depart from attraction AW
17:18 Arrive at attraction U
18:08 Depart from attraction U
18:33 Arrive at piccolo store C
Dinner time
20:03 Depart from piccolo store C
20:19 Arrive at attraction J
23:19 Depart from attraction J
00:27 Arrive at hotel A
Sleep time
2nd day
09:27 Depart from hotel A
09:46 Arrive at attraction B
12:16 Depart from attraction B
14:40 Arrive at piccolo store 1
Lunch time
16:10 Depart from piccolo store 1
16:42 Arrive at attraction V
17:32 Depart from attraction V
18:18 Arrive at attraction AL
19:38 Depart from attraction AL
22:10 Arrive at piccolo store 8
Dinner time
23:40 Depart from piccolo store 8
23:48 Arrive at hotel C
Sleep time
3rd day
08:48 Depart from hotel C
12:19 Arrive at attraction Q
12:39 Depart from attraction Q
13:15 Arrive at piccolo store 2
Lunch time
14:45 Depart from piccolo store 2
15:16 Arrive at attraction I
16:16 Depart from attraction I
18:22 Arrive at attraction AN
18:42 Depart from attraction AN
19:39 Arrive at piccolo store 8
Dinner time
21:09 Depart from piccolo store 8
21:17 Arrive at hotel C
Sleep time
4th day
06:17 Depart from hotel C
07:16 Arrive at attraction AQ
08:16 Depart from attraction AQ
08:57 Arrive at attraction AD
09:17 Depart from attraction AD
09:46 Arrive at attraction AC
11:16 Depart from attraction AC
14:31 Arrive at piccolo store 6
Lunch time
16:01 Depart from piccolo store 6
16:39 Arrive at attraction AF
16:49 Depart from attraction AF

Table 4. *Cont.*

20:26 Arrive at attraction AP
20:36 Depart from attraction AP
20:49 Arrive at attraction X
32:39 Depart from attraction X
23:31 Arrive at hotel A
Sleep time
5th day
08:31 Depart from hotel A
09:13 Arrive at attraction AB
13:43 Depart from attraction AB
11:33 Arrive at attraction AV
11:53 Depart from attraction AV
12:41 Arrive at piccolo store 5
Lunch time
14:11 Depart from piccolo store 5
16:44 Arrive at attraction T
18:21 Depart from attraction T
21:03 Arrive at piccolo store 8
Dinner time
22:33 Depart from piccolo store 8
22:41 Arrive at hotel C
Sleep time
6th day
07:41 Depart from hotel C
11:44 Arrive at attraction K
12:54 Depart from attraction K
13:30 Arrive at piccolo store 7
Lunch time
15:00 Depart from piccolo store 7
15:22 Arrive at attraction AX
16:42 Depart from attraction AX
17:24 Arrive at attraction E
19:24 Depart from attraction E
20:58 Arrive at piccolo store 1
Dinner time
22:28 Depart from piccolo store 1
00:37 Arrive at hotel D
Sleep time
7th day
09:37 Depart from hotel D
11:22 Arrive at attraction AA
12:22 Depart from attraction AA
12:37 Arrive at piccolo store 6
Lunch time
14:07 Depart from piccolo store 6
15:29 Arrive at attraction G
18:29 Depart from attraction G
20:08 Arrive at piccolo store 4
Dinner time
21:38 Depart from piccolo store 4
21:53 Arrive at hotel B
Sleep time
06:53 Depart from hotel B
09:45 Home

Scheduling Results for the Last Generation
Fitness Value:
1st day
09:00 Departure

Table 4. *Cont.*

11:05 Arrive at attraction AN
11:25 Depart from attraction AN
14:47 Arrive at piccolo store 7
Lunch time
16:17 Depart from piccolo store 7
16:19 Arrive at attraction AX
17:59 Depart from attraction AX
18:19 Arrive at attraction AW
19:09 Depart from attraction AW
19:35 Arrive at piccolo store 7
Dinner time
21:05 Depart from piccolo store 7
23:35 Arrive at hotel C
Sleep time
2nd day
08:35 Departure
09:27 Arrive at attraction A
10:42 Depart from attraction A
11:16 Arrive at attraction P
12:06 Depart from attraction P
12:52 Arrive at piccolo store C
Lunch time
14:22 Depart from piccolo store C
14:38 Arrive at attraction J
17:38 Depart from attraction J
17:42 Arrive at attraction E
19:42 Depart from attraction E
20:38 Arrive at piccolo store 4
Dinner time
22:00 Depart from piccolo store 4
22:15 Arrive at hotel B
Sleep time
3rd day
07:15 Departure
08:03 Arrive at attraction Q
08:23 Depart from attraction Q
09:19 Arrive at attraction K
10:29 Depart from attraction K
11:24 Arrive at attraction U
12:14 Depart from attraction U
12:39 Arrive at piccolo store C
Lunch time
14:09 Depart from piccolo store C
14:38 Arrive at attraction B
17:08 Depart from attraction B
19:01 Arrive at attraction AD
19:21 Depart from attraction AD
20:51 Arrive at piccolo store 5
Dinner time
22:21 Depart from piccolo store 5
23:11 Arrive at hotel A
Sleep time
4th day
08:11 Departure
09:18 Arrive at attraction C
12:18 Depart from attraction C
13:40 Arrive at piccolo store 6
Lunch time
15:10 Depart from piccolo store 6

Table 4. *Cont.*

15:48 Arrive at attraction AF
15:58 Depart from attraction AF
17:16 Arrive at attraction V
18:06 Depart from attraction V
20:14 Arrive at piccolo store 4
Dinner time
21:44 Depart from piccolo store 4
22:14 Arrive at hotel A
Sleep time
5th day
07:14 Departure
08:24 Arrive at attraction I
09:24 Depart from attraction I
12:38 Arrive at attraction AP
12:48 Depart from attraction AP
13:18 Arrive at piccolo store 8
Lunch time
14:48 Depart from piccolo store 8
15:17 Arrive at attraction X
16:07 Depart from attraction X
18:15 Arrive at attraction AL
19:35 Depart from attraction AL
22:07 Arrive at piccolo store 8
Dinner time
23:37 Depart from piccolo store 8
23:45 Arrive at hotel C
Sleep time
6th day
08:45 Departure
09:44 Arrive at attraction AQ
10:44 Depart from attraction AQ
11:43 Arrive at attraction AV
12:03 Depart from attraction AV
14:08 Arrive at piccolo store 8
Lunch time
15:38 Depart from piccolo store 8
16:59 Arrive at attraction AC
18:29 Depart from attraction AC
19:00 Arrive at piccolo store A
Dinner time
20:30 Depart from piccolo store A
21:28 Arrive at attraction AB
22:58 Depart from attraction AB
23:40 Arrive at hotel A
Sleep time
7th day
08:40 Departure
11:14 Arrive at attraction T
12:54 Depart from attraction T
13:55 Arrive at piccolo store C
Lunch time
15:25 Depart from piccolo store C
18:24 Arrive at attraction AA
19:24 Depart from attraction AA
19:39 Arrive at piccolo store 6
Dinner time
21:09 Depart from piccolo store A
21:43 Home

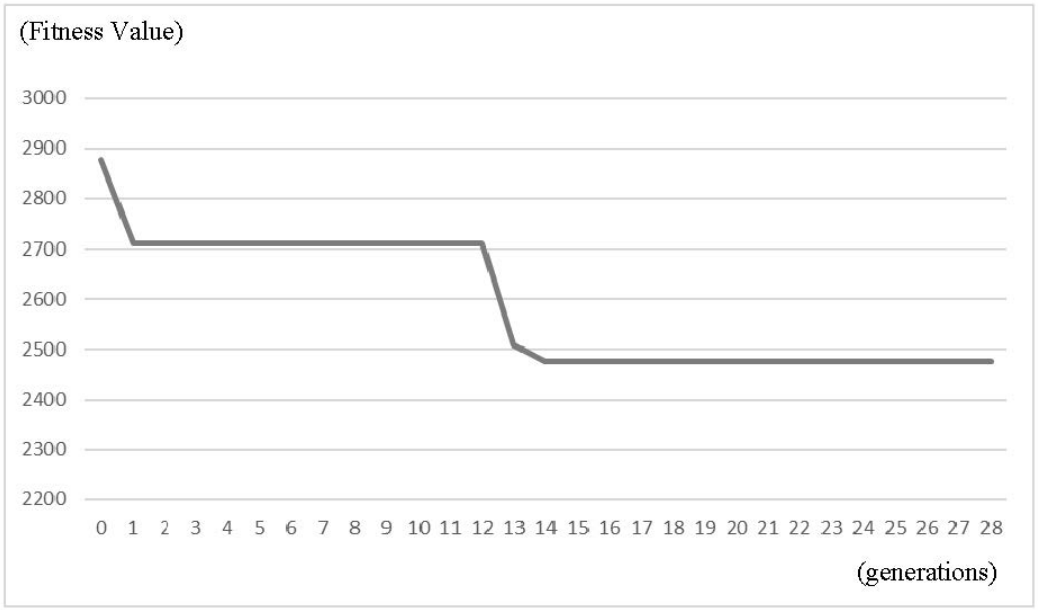

Figure 8. Evolution history of fitness value of seven-day travel itinerary.

Finally, the proposed improved ACS in this study has three differences from the general methods (Gavalas et al. (2015); Lim et al. (2015); Wörndl et al. (2016); Wang et al. (2016); Xu et al. (2016); Rybchak et al. (2017); Zhang et al. (2016)):

1. The general methods spread the ants evenly on different nodes, which will cause some ants to start from the scenic spot and some ants to start from the restaurant, passing through the starting point on the way. Since actual travel starts from the departure point, this study scatters all the ants at the origin, which is the starting point.
2. The general method restricts each node to be visited only once to avoid the situation where ants wander to the same point indefinitely. However, this study uses Watch to control ants to repeatedly visit a node to solve the problem that accommodation and dining locations may be repeated and that they still need to return to the starting point at the end of the trip.
3. The general method is to use the total length of the path as the fitness value. The fitness value of this study considers satisfaction with restaurants, attractions, and hotels.

Moreover, we give a comparison between the proposed ACS and traditional ACS: traditional ACS cannot perform node duplication operations, but the ACS proposed in this study can perform node duplication operations.

6. Conclusions

This study proposed an improved ant algorithm that allows the nodes to be divided into Hard Nodes and Soft Nodes through the interactive operation of Watch and Tabu List, so that the ants can walk to repeated nodes without falling into infinitely hovering over the limits of a certain interval and nodes. By applying the improved ant algorithm to recommend travel itineraries, a series of itineraries can be efficiently planned with optimized travel paths. In addition, in order to meet the highest degree of user satisfaction with restaurants and hotels, a user satisfaction calculation for restaurants and hotels has been added to the schedule.

However, there are still two disadvantages. Because various emergencies will be encountered in reality, such as roads being closed due to heavy rain, or the original traffic route needing to be rerouted due to parades, all attractions, restaurants, and hotels in this study are virtual. Therefore, the travel itineraries recommended in this study may not be as expected in reality. In addition, it can be seen that the travel schedule under the operation of a large number of nodes may take a lot of time. We may be able to find some

way to reduce the time consumption. Future work can focus on overcoming these two disadvantages.

Author Contributions: Conceptualization, S.-T.C.; methodology, S.-T.C.; software, S.-T.C., L.-C.L., W.-Y.H., Y.-H.L. and B.-Y.W.; validation, S.-T.C. and T.-H.W.; writing—review and editing, S.-T.C. and R.-J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

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

References

1. Lee, C.-S.; Chang, Y.-C.; Wang, M.-H. Ontological recommendation multi-agent for Tainan City travel. *Expert Syst. Appl.* **2009**, *36*, 6740–6753. [CrossRef]
2. Batabyal, A.A. Scheduling trips during the slack season: An aspect of the economics of seasonal tourism. *Tour. Econ.* **2009**, *15*, 261–266. [CrossRef]
3. Tseng, S.-Y.; Ding, J.-W.; Chen, R.-C. WEB-based tour planning support system using genetic and ant colony algorithms. *J. Internet Technol.* **2010**, *11*, 901–908.
4. Batabyal, A.A.; Beladi, H. Transport provision to tourists by a cost minimizing firm: A stochastic characterization. *Lett. Spat. Resour. Sci.* **2011**, *4*, 103–108. [CrossRef]
5. Yang, L.; Zhang, R.; Sun, H.; Guo, X.; Huai, J. A Tourist Itinerary Planning Approach Based on Ant Colony Algorithm. In Proceedings of the Web-Age Information Management: 13th International Conference, WAIM 2012, Harbin, China, 18–20 August 2012; pp. 399–404.
6. Gavalas, D.; Konstantopoulos, C.; Mastakas, K.; Pantziou, G. A survey on algorithmic approaches for solving tourist trip design problems. *J. Heuristics* **2014**, *20*, 291–328. [CrossRef]
7. Bin, R. Research on Tourism Service Intelligent Recommendation System Based on Apriori-MD Algorithm. *Appl. Mech. Mater.* **2014**, *651–653*, 1642–1646. [CrossRef]
8. No, E.; Kim, J.K. Determinants of the Adoption for Travel Information on Smartphone. *Int. J. Tour. Res.* **2013**, *16*, 534–545. [CrossRef]
9. Gavalas, D.; Kasapakis, V.; Konstantopoulos, C.; Pantziou, G.; Vathis, N.; Zaroliagis, C. The eCOMPASS multimodal tourist tour planner. *Expert Syst. Appl.* **2015**, *42*, 7303–7316. [CrossRef]
10. Lim, K.-H.; Chan, J.; Leckie, C.; Karunasekera, S. Personalized Tour Recommendation Based on User Interests and Points of Interest Visit Durations. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, 25–31 July 2015; pp. 1778–1784.
11. Wörndl, W.; Hefele, A. Generating Paths Through Discovered Places-of-Interests for City Trip Planning. In Proceedings of the Information and Communication Technologies in Tourism 2016, Bilbao, Spain, 2–5 February 2016; pp. 441–453.
12. Wang, X.; Li, X.; Zhen, F.; Zhang, J. How smart is your tourist attraction?: Measuring tourist preferences of smart tourism attractions via a FCEM-AHP and IPA approach. *Tour. Manag.* **2016**, *54*, 309–320. [CrossRef]
13. Xu, M.; You, X. Optimum design for tourism line based on improved ant colony system algorithm. *Int. J. Manag. Value Supply Chain. (IJMVSC)* **2016**, *7*, 1–8.
14. Rybchak, Z. Optimization of travel routes based on modified genetic and ant algorithms. *Econtechmod. Int. Q. J.* **2017**, *6*, 85–90.
15. Zhang, L.; Wang, Y.P.; Sun, J.; Yu, B. The sightseeing bus schedule optimization under Park and Ride System in tourist attractions. *Ann. Oper. Res.* **2016**, *273*, 587–605. [CrossRef]
16. Dorigo, M.; Maniezzo, V.; Colorni, A. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B* **1996**, *26*, 29–41. [CrossRef] [PubMed]
17. Marco, D.; Gambardella, L.M. Ant colonies for the traveling salesman problem. *BioSystems* **1997**, *43*, 73–81.
18. Marco, D.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman Problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66.
19. Peker, M.; ŞEN, B.; Kumru, P.Y. An efficient solving of the traveling salesman problem: The ant colony system having parameters optimized by the Taguchi method. *Turk. J. Electr. Eng. Comput. Sci.* **2013**, *21*, 2015–2036. [CrossRef]
20. Mou, L. An efficient ant colony system for solving the new Generalized Traveling Salesman Problem. In Proceedings of the 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, Beijing, China, 15–17 September 2011; pp. 15–17.

21. Thammano, A.; Oonsrikaw, Y. Improved Ant Colony Optimization with Local Search for Traveling Salesman Problem. In Proceedings of the 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Toyama, Japan, 8–11 July 2019; pp. 8–11.
22. Dahan, F.; Hindi, E.; Mathkour, H.; Salman, A. Dynamic Flying Ant Colony Optimization (DFACO) for Solving the Traveling Salesman Problem. *Sensors* **2019**, *19*, 1837. [CrossRef] [PubMed]