A Method for Composing Tour Schedules Adaptive to Weather Change

Bing Wu
Graduate School of Information Science
Nara Institute of Science and Technology
Email: bing-w@is.naist.jp

Yoshihiro Murata
Graduate School of Information Sciences
Hiroshima City University
Email: yoshi-m@hiroshima-cu.ac.jp

Naoki Shibata
Department of Information Processing and Management
Shiga University
Email: shibata@biwako.shiga-u.ac.jp

Keiichi Yasumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Email: yasumoto@is.naist.jp

Minoru Ito
Graduate School of Information Science
Nara Institute of Science and Technology
Email: ito@is.naist.jp

Abstract—People do sightseeing in their spare time to relax, and sightseeing is an important industry for some regions. The satisfaction of tourists critically depends on weather during their tours. In order to give people maximum satisfaction, we have to take care of the weather when planning a schedule. However, if there are many possible patterns for weather changes, the number of possible schedules will be very large, and in this case, it is difficult to find a good schedule. In this paper, we formulate the problem to compose schedules for probabilistically changing weather when the probability of future weather is given. We also propose an approximation algorithm for this problem based on the greedy search and the neighborhood search techniques. To evaluate the proposed method, we compare our method with Brute force method and a greedy method for an instance of Beijing sightseeing. As a result, the proposed method found the optimal solution in 6 sec, while the Brute force method took 16 hours. The proposed method composed a schedule whose expected satisfaction is 17.9% higher than that composed by the greedy method, for an instance with 20 destinations.

I. INTRODUCTION

In recent years, there have been proposed various personal navigation systems which navigate users to their destinations and provide them surrounding information. The main usage of such navigation systems is for the entertainment such as sightseeing. Existing navigation systems for entertainment include a system which dynamically calculates and provides an adequate route based on user’s context [1], a system for navigating users in an indoor facility [2], and a system which provides users sightseeing information through portable computing devices [3]. Aiming at navigating users during the entire tour period, a system for scheduling a tour, guiding transfers, and providing surrounding information at both indoor and outdoor has been proposed [4]. In order to compose a good schedule, various problems are formalized, and there have been proposed various methods for finding solutions of those problems. Traveling Salesman Problem (TSP) is the well-known problem for planning the shortest route to visit all of given destinations. Since this problem is known as a NP-hard problem, there have been proposed some efficient heuristic algorithms to solve the problem, such as Lin-Kernighan method [5]. Theme Park Problem (TPP) is the problem for planning the best schedule to visit attractions taking their congestion degree into account. For TPP, there has been proposed a method for maximizing users satisfaction degree through navigation to the users [6]. Some of the above existing systems have functions for providing context-aware and flexible navigation to users. However, the existing systems only navigate users to the next destination, and do not have a function for planning a tour schedule with high user satisfaction taking into account the visiting order of multiple destinations, distance between destinations, and tour time.

In our previous work, we have proposed a personal navigation system called P-Tour [7]. P-Tour has functions for navigating users and planning a tour schedule to visit multiple sightseeing spots taking into account the user’s preference, time restrictions, and distance between spots. However, P-Tour does not change tour schedules depending on weather change. Since the user satisfaction in a sightseeing tour depends greatly on weather, the best schedule differs among various weather types. A simple way to cope with this problem is to estimate expected user satisfaction for each spot in rainy weather as well as fine weather so that the schedule for rainy weather can also be planned. However, when weather can change during the tour, the best tour schedule varies depending on the weather change patterns. Since there will be so many weather change patterns, planning the optimal schedules for all of these patterns is a difficult problem, and to the best of our knowledge, there is no good algorithm to solve this problem.

In this paper, assuming that the probability of weather change pattern is given from weather forecast, we propose an algorithm for planning tour schedules which can adapt to various weather change patterns. In our algorithm, we represent the schedules as a schedule tree consisting of ordered sequences of visiting spots where its root node corresponds to the starting location and its branches correspond to visiting different spots for weather types (e.g., fine, cloudy, and rainy)
at the time just after the previous spot has been visited. In a tree, each path corresponds to a sequence of spots in the visiting order where a user satisfaction degree based on the corresponding weather type is attached to each node (i.e., spot) and a probability of being either fine, cloudy, or rainy is attached to each branch (i.e., movement between spots). Given the user satisfaction degree for each pair of a spot and weather type, weather forecast, map information, and starting and returning locations and time, our problem is to compose the schedule tree maximizing the expectation of total user satisfaction degree. To solve this problem, we propose a heuristic algorithm based on a greedy algorithm and a neighborhood search method. Our neighborhood search method swaps spots in a subtree and it is advantageous in the sense that it always produces a schedule tree without a path including the same spot more than once, to avoid redundant tour schedules.

For evaluation, we applied our method to compose a schedule tree for sightseeing in Beijing city. As a result, our method could compose the optimal schedule tree in 6 sec., while a Brute force search took 16 hours to find the optimal schedule tree, for an instance with 6 destinations. For another instance with 20 sightseeing spots, our algorithm utilizing both greedy and neighborhood search techniques could compose 17.9% a better schedule tree than a greedy method.

II. PROBLEM FORMULATION

In this section, we formulate the problem to compose the schedule tree for adapting to weather changes during sightseeing. We first explain the overview of the problem by examples, and then we define the inputs, the outputs, and the objective function of the problem. We also show the NP-hardness of the formulated problem.

A. Problem Overview

The user’s satisfaction of visiting a sightseeing spot may differ depending on the weather. The proposed method compose the schedule tree that maximizes the expected user satisfaction degree calculated from weather forecast (predicted future weather).

We assume that the weather forecast information is given beforehand. The weather forecast information is the probability transition for each weather occurring at a place over time, as shown in Table I. A user inputs the positions of candidates of sightseeing spots. We assume that the satisfaction degree of visiting a destination differs on each weather. For example, suppose that the TianAnMen Square and the BeiHai park have good satisfaction degree only when the weather is fine, and the Palace museum and the Military museum have good satisfaction degree only when the weather is rainy. In this case, if the weather is fine throughout the day, the sightseeing schedule to visit the TianAnMen Square and the BeiHai park is desirable, as shown in Fig. 1 (Left). However if the weather is rainy throughout the day, visiting the Palace museum and the Military museum is desirable as shown in Fig. 1 (Center). If the weather is fine in the morning and starts raining in the afternoon, switching the destinations during the trip is desirable (Fig. 1 (Right)). Thus, the schedule for deciding sightseeing spots according to weather changes can be expressed as a tree.

For the sake of simplicity, we assume that each user observes and decides the next destination just after finishing visiting each destination. The schedule tree can be expressed by a graph shown in Fig. 2. Since the method knows the probability transition for each weather over time, we know the probability of a user to visit each node of the schedule tree. In Fig. 2, the user departs Beijing Hotel at 9AM, and he goes to the TianAnMen square if the weather is fine. The values “(100 points, 20 %)” under the node means that the user satisfaction degree of visiting museum is 100 points when the weather is fine, and the probability of being fine at that time (i.e., visiting the node) is 20%. By multiplying the user satisfaction degree with the probability, we can calculate the expected user satisfaction degree. The objective of the problem is to plan the schedule tree which maximizes the total of expected user satisfaction degree.

B. Input

Input I consists of a database input which is given beforehand and the data input by the user.

Database Input It consists of the road map, the data for each destination, and weather forecast information. The details of the database input are shown below.

- Road map data: It is given as a graph \( G = (V, E) \) representing the road network. Each edge is assigned the following data.
  - Distance between vertices: distance between vertices \( v_1 \) and \( v_2 \) is given by function \( dist(v_1, v_2) \).
- Destination data: It represents all candidate destinations.

### Table I

<table>
<thead>
<tr>
<th>Time</th>
<th>Weather</th>
<th>Destination</th>
<th>Time</th>
<th>Weather</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-10:00</td>
<td>Fine</td>
<td>TianAnMen</td>
<td>09:00-10:00</td>
<td>Fine</td>
<td>The Palace museum</td>
</tr>
<tr>
<td>10:00-11:00</td>
<td>Fine</td>
<td>BeiHai Park</td>
<td>10:20-11:00</td>
<td>Rain</td>
<td>Military museum</td>
</tr>
<tr>
<td>11:20-11:28</td>
<td>Fine</td>
<td>Go back</td>
<td>11:55-12:00</td>
<td>Fine</td>
<td>Go back</td>
</tr>
</tbody>
</table>

**Fig. 1.** Example of schedules
- $D = \{d_1, d_2, \ldots, d_n\}$: Set of all candidate destinations (i.e. Tiananmen Square, National Palace Museum, etc.)
- $V = \{v_1, v_2, \ldots, v_n\}$: Set of all positions of the candidate destinations. Each position corresponds to one of vertices of the road map graph.

- Weather forecast information
  - $W$: Set of all weather types (i.e. $W = \{\text{fine}, \text{cloudy}, \text{rainy}\}$). It is assumed that only one of these types occurs at a time.
  - $p(w_i, t) \rightarrow \mathbb{R}$: Function which returns the probability of weather $w_i$ occurring at the destination at time $t$. These probabilities are given beforehand as a table shown in Table I.

**User Input** The user inputs the duration of sightseeing, starting location and time, finishing location and time, satisfaction degree for each destination, and time restriction of visiting each destination. The satisfaction degree expresses how eagerly the user wants to visit each destination at each weather. This data is input for each combination of a destination and weather. In order to alleviate the labor of inputting the user data from the scratch, the proposed method has default values for all items of the user input, and the user is just prompted to change these values.

User input consists of the following data items.

- arrival/departure data:
  - $pd_i, pd_j \in D$: starting and finishing points of the whole sightseeing tour.
  - $pt_i, pt_j \in D$: time restrictions of the starting and finishing points.
  - $speed$: Moving speed of the user

- Destination data for each spot
  - $pre_{ij}: D \rightarrow N$: $pre(d_i, w_j)$ is the satisfaction degree of the destination $d_i$ when weather is $w_j$.
  - $rst_i$: Time restriction for arrival time at destination $d_i$.
  - $dur_i$: Time restriction for visiting duration at destination $d_i$.
  - $\alpha, \beta, \gamma$: Constants for calculating total satisfaction. $\alpha$ is the weight for satisfaction on destinations. $\beta$ is the weight for the penalties of moving time between destinations. $\gamma$ is the weight for the penalty of exceeding the finishing time of the whole sightseeing tour.

**C. Output**

The proposed method outputs a tree where the root is the starting point, the leaf nodes are the finishing point, and the internal nodes are the sightseeing destinations as shown in Fig. 2. The tree is $|W|$-ply tree, which means, its root and internal nodes have $|W|$ children nodes, since the user decides one of the next destinations in the tree based on the weather after the user finishes visiting each destination.

**D. Objective**

The total satisfaction is defined based on our previous method[7]. We calculate the overall satisfaction by summing up the satisfaction degrees for all visited destinations satisfying the time restriction and by subtracting penalties for moving time and exceeding time from the arrival time for the finishing point. In this paper, since we handle the problem where the user decides the next visiting destination according to the weather, the objective is to compose the schedule tree (such as Fig. 2) which maximizes the total of the expected user satisfaction degree (hereafter, expected degree). The expected degree is calculated as follows.

1) Calculate satisfaction for each node. Let $w \in W$ be the weather just after the user finished visiting the previous destination, and $d \in D$ be the destination corresponding to the node. The satisfaction for the node is $\alpha \times pre(d, w)$.

2) Calculate the penalties for each edge. This penalties are imposed on moving time between destinations. The penalty for moving between $d'$ and $d''$ is $\beta \times \text{dist}(d', d'')$.

3) Calculate the arrival time to each destination. Arrival time at $d''$ can be obtained by adding the moving time $\text{dist}(d', d'')/\text{speed}$, and the stay time at destination $d'$ to the arrival time at the previous destination $d'$.

4) Calculate the satisfaction for each leaf node (finishing point). If the time exceeds the specified finishing time, we impose penalty of $\max(0, \gamma \times \text{exceeding time})$.

5) Check the restrictions on each node. If the corresponding restriction is not met for a node or the destination is
already visited, the satisfaction for the node is set to 0.
6) Calculate the probability for visiting each node. The probability can be calculated by multiplying the probability for the previous node and the probability of occurrence of the corresponding weather, as shown in Fig. 2.
7) Calculate the total satisfaction (i.e., expected degree) by multiplying satisfaction and the probability on each node, and summing all these values up.

E. NP-hardness

The problem to compose the schedule tree can be converted into the original P-Tour [7] problem of planning the sightseeing schedule without taking account of the weather changes, if the weather forecast is stuck 100% to a certain weather pattern. The P-Tour problem is NP-hard [7]. Thus, the problem to plan the schedule tree is NP-hard.

III. APPROXIMATION ALGORITHM

Because the problem to compose the optimal schedule tree adaptable to weather change is NP-hard, it is difficult to get the optimal solution in practical time. Therefore, we propose an approximation algorithm using a greedy method and a neighborhood search technique.

In the case that the tourist stays for almost same period at each destination and the time to move between the destinations are almost same, the height of the schedule tree can be predicted in advance. Below, we assume that the height of the schedule tree denoted by height is given in advance for simplification.

Our proposed algorithm consists of two phases: (1) Greedy Search Phase and (2) Neighborhood Search Phase. Below we give detailed explanation of these two phases.

A. Greedy search phase

This phase selects a destination with the highest user satisfaction degree one by one, and composes a schedule tree.

In the proposed approximation algorithm, a schedule tree is represented by a heap structure \( D' = (d'_0, d'_1, d'_2, ..., d'_{N-1}) \), where \( N = \sum_{i=1}^{\text{height}} |W|^i-1 \), \( d'_0 \) is the starting point. \( d'_0 \) must be \( pd_s \) of user input. The children nodes of the node \( d'_i \) (which are the candidate destinations visited after visiting \( d'_i \)) are \( \{d'_{i+|W|+1}, d'_{i+2|W|+1}, ..., d'_{i+|W|^i}|W|\} \). These nodes correspond to the cases of weather \( w_1, r_2, \ldots, w|W| \), respectively. The parent node of \( d'_i \) (the destination before visiting \( d'_i \)) is \( d'_{i-1}/|W| \). Similarly, stay time at node \( d'_i \) is denoted by \( \text{stay}_i \), and \( \text{Stay} = \{\text{stay}_0, \text{stay}_1, \text{stay}_2, ..., \text{stay}_{N-1}\} \). Using them, we represent a schedule tree by \( s = (D', \text{Stay}) \).

The pseudocode for this phase is shown below.

1) Set the current node number \( i = 0 \). Let \( p_i \) be probability of visiting node \( d'_i \). Set \( p_0 = 1 \) and, starting time \( t_0 = pt_s \).
2) If distance between \( d_i \) and root of tree (so, depth of node) is more than height, the algorithm is terminated.
3) If \( p_i = 0 \) (probability of visiting is zero), goto step 6.
4) Repeat the following steps for \( j = 1, 2, \ldots, |W| \):
   a) Let denote the child node number by \( k = |W| \times i + j \). Let \( p_k \) be visiting probability of node \( d'_k \) when weather is \( w_j \). Set \( p_k = p_i \times \text{prob}(w_j, t_i + \text{stay}_i) \).
   b) If \( p_k = 0 \), visiting probability is zero. Set \( j = j + 1 \), goto top of loop.
   c) If depth of \( d'_k \) is height, then set \( d'_k = pd_y \), and \( j = j + 1 \), and goto top of loop. Let \( D'' \) be a set of destinations that belong to \( D - \{d'_i\} \) but do not belong to ancestors of \( d'_i \).
   d) Repeat the following steps for \( d'' \in D'' \):
      i) Calculate moving time \( t'_ik = \text{dist}(d'_i, d'')/\text{speed} \).
      ii) Let \( t''_k \) be the visiting time of child node \( d'' \). Set \( t''_k = t_i + \text{stay}_i + t'_ik \).
      iii) Calculate the user satisfaction degree at \( d'' \) such that \( \alpha \times \text{pre}(d'(i), w_j) - \beta \times t'_ik \).
      e) Find \( d'' \) which has the maximum user satisfaction degree in the loop described above, and set \( d'_k = d'' \). Similarly, set \( t'_k = t''_k \). Also, give \( \text{stay}_k \) such that time restriction at \( d'_k \) is satisfied. For example, set \( \text{stay}_k = 60 \) in the case that “the stay time is longer than 60 min.”
5) Goto step 7.
6) Repeat the following steps for \( j = 1, 2, \ldots, |W| \):
   a) Let \( k \) be the child node number. Set \( k = |W| \times i + j \).
   b) Set \( p_k = 0 \).
7) Set \( i = i + 1 \), and goto step 2.

B. Neighborhood search phase

In the proposed method, the neighborhood search is repeatedly applied to the solution derived by the greedy search phase to improve it. In this phase, two nodes in the schedule tree are randomly selected, and the destinations corresponding to these nodes are swapped. If the expected satisfaction is improved, the new tree replaces the old one. Otherwise, the change is discarded. Schedules made by the above method must not be a schedule to visit the same destination more than one time. This kind of faulty schedules are always discarded since the satisfaction degree for such schedules is low. This function is applied until the termination condition is satisfied.

The pseudocode for the neighborhood search is shown below. We assume that the heap structure \( D' \) for a schedule tree is given by the greedy search phase in advance.

1) Set \( D'_{\text{odd}} = D' \).
2) Select node \( d'_i \in D' \) randomly. Let \( T_i \) be sub tree where root node is \( d_i \).
3) Let \( D'' \) be a set of destinations that belong to \( D - \{d'_i\} \) and do not belong to ancestors of \( d'_i \).
We use the termination condition which is constant number of repetition.

IV. EXPERIMENTS AND EVALUATION

A. Experimental setting

To evaluate our proposed method, we measured calculation time and the expected degree of the obtained schedule tree.

In the experiments, we executed the proposed algorithm on an ordinary PC with Celeron 2.0GHz, 2.00GB Memory, WindowsXP pro., and Java 2.3.2.

We used the map of Beijing prefecture. Both of the starting and the finishing points are Beijing Hotel. We assumed that moving speed of a user is 40 km/h, dur<sub>i</sub> of each destination is 1 hour. Also, we set parameter values according to the preliminary experiments as follows: α = 1, β = 5, and γ = 1. So, if a user delays 1 min. at the finishing point, the satisfaction value is decreased by 5 points.

B. Experimental Results

1) Optimality: To evaluate optimality of the obtained schedule tree, we compared our method with the Brute force search method and a greedy method for a small instance. This instance has 6 destinations. The other settings are as follows: pt<sub>s</sub> is 9:00, pt<sub>f</sub> is 13:00, and height is 3.

We show experimental results in Table II. By our method, we obtained the optimal solution in 6031 msec. for this instance, while the Brute force search method took 16 hours. On the other hand, a greedy method obtained a solution in 16 msec. However it could not obtain the optimal solution. Therefore, we confirmed that our method is speedy and effective.

2) Quality: To evaluate the quality of the obtained schedule tree, we compared our method with a greedy method for a bigger instance. This instance has 20 destinations. The other settings are as follows: pt<sub>s</sub> is 9:00, pt<sub>f</sub> is 18:00, and height is 6. We used 3 weather change patterns which are fine-to-fine case, fine-to-rain case, and fickle case.

We show experimental results in Table III. These results are the average values of 10 trials. In fine-to-fine and fickle cases, the results of our method were almost similar to that of the greedy method. This implies that the greedy strategy is effective in these cases. However, in the case that weather changes from fine to rain, our method composed a schedule whose expected degree is 17.9% higher than that composed by the greedy method. This implies that our method is effective in the case that the future weather will surely change. However, our method took for calculation about 2 minute for this instance. We need to decrease the computation cost for the cases that the sightseeing area includes more destinations.

V. CONCLUSIONS

In this paper, we defined a problem to compose a schedule tree which can adapt to arbitrary weather change patterns, based on weather forecast, and proposed a heuristic algorithm based on a greedy method and a neighborhood search technique to solve the problem efficiently. To evaluate the proposed method, we compared the efficiency of the proposed method and a greedy method for sightseeing in Beijing city. As a result, our method could compose the optimal schedule tree in 6 sec., while the Brute Force Search took 16 hours to find the optimal schedule tree for the instance with 6 destinations. For the instance with 20 destinations, our algorithm composed a 17.9% better schedule tree than the greedy method.

Currently our method supposes that the height of a schedule tree is given in advance and derives a schedule tree such that all path lengths are the same. However, when the sightseeing time of spots differs from each other, we need to compose a schedule tree such that path lengths are different from each other. We will extend our algorithm to treat such a case. Using the different satisfaction degree of each destination among different weather conditions is an important point in the proposed method, but we did not mention in the paper how we decide adequate values for the satisfaction degree. We can set it through the questionnaire, the web page of the sightseeing, etc. Also, the influence of weather for transportation will be taken into account by our algorithm too. Since our method relies on a neighborhood search technique for improvement of solutions, if the target problem contains a lot of local optimums, our method may not derive a good solution. There are some techniques to avoid local optimums such as a technique to solve TSP with a technique to solve Vehicle Routing Problem (VRP) [9]. In the future, we will extend our method based on these techniques.
REFERENCES


[8] Yahoo Japan, “Yahoo! the weather information, ” http://weather.yahoo.co.jp/weather