A Personal Navigation System for Sightseeing across Multiple days

Takamasa Kinoshita[†], Munenobu Nagata[†], Yoshihiro Murata[†],
Naoki Shibata[‡], Keiichi Yasumoto[†]and Minoru Ito[†]

[†]Graduate School of Information Science, Nara Institute of Science and Technology
Takayama, Ikoma, Nara 630-0192, Japan, {takama-k, muneo-n, yosihi-m, yasumoto, ito} @is.naist.jp
[‡]Department of Information Processing and Management, Shiga University, Hikone 522-8522, Japan,
shibata@biwako.shiga-u.ac.jp

ABSTRACT

Recently, various personal navigation systems are available on the market. In this paper, we propose a new scheduling function to compose sightseeing tours across multiple days. Scheduling for multiple days is challenging since the number of possible sequences to travel destinations becomes huge and accommodation places have to be carefully chosen considering the schedules before and after each stay. So, some techniques are required to calculate the sightseeing schedule across multiple days at practically short time. In our method, we divide the target area into several sub-areas, and we restrict each day trip to be in a limited number of areas with infrequent passing of area boundary, in order to decrease the number of sequences. We have designed and implemented a genetic algorithm to solve this scheduling problem at a practical time. In this algorithm, accommodation places are represented in chromosome as exclusive alleles, to make schedules which contain only one accommodation place for each day. In order to evaluate our method, we made various sightseeing schedules across multiple days using the digital map of the Tohoku area (Japan Geographical Survey Institute numeric map 25000), and confirmed that the proposed algorithm could compute the near best 6 days schedule in several tens of seconds.

Keywords: Personal navigation system, Intelligent transport system, Map data, Combinatorial optimization, Evolutionaly algorithm

1 Introduction

Recently, car navigation systems and personal navigation systems, which navigate user to a destination via portable computing devices such as mobile phones, are widely available. The basic objective of existing navigation systems is to navigate user to a single destination and provide information regarding to the destination. So, they are not suitable for finding an efficient tour schedule including two or more destinations. In order to visit multiple sightseeing spots, tourists may join a package tour provided by travel agencies. However, they have to follow a predetermined schedule in a tour, and thus there may be no plan which perfectly satisfies the tourist's expectation. It also lacks flexibility to change the schedule on the way. Recently, travel agencies offer sightseeing plans in which participants are able to decide destinations and lodging places by themselves according to budget and purpose of visit. However, participants have to make their schedules by themselves from scratch.

We have designed and developed a personal navigation system named "P-Tour" [8] which guides user to multiple destinations selected by taking user's preferences and time restrictions into account. Since P-Tour treats accommodation places similarly to sightseeing places, we may need trial-and-error to find an efficient tour schedule across multiple days which includes an accommodation place at the end of each day schedule.

Finding an efficient schedule for sightseeing across multiple days is harder than single day sightseeing, since the schedule usually includes many sightseeing spots. Besides it, choice of accommodation places is crucial since it largely affects users' satisfaction, and choice of accommodation places have a large influence on total cost, and they have to be chosen taking user's preferences regarding to meals, amenities and environments into account. Also, choice of accommodation places limits sightseeing spots which can be visited right before and after accommodations.

In this paper, we propose a function to find a schedule over multiple days, which can be used for P-Tour. In the proposed method, The region including sightseeing spots is divided into sub areas to reduce the number of possible sequences to travel sightseeing spots. Also, to find schedules including accommodation places, we divide the whole schedule into subschedules per day. The number of possible sequences to travel sightseeing spots is further reduced by asking user to specify approximate regions of accommodation places.

We implemented the proposed method, and evaluated the system using Digital Map 25000 issued by the Geographical Survey Institute of Japan. We input 108 sightseeing spots in Tohoku region, measured calculation time and quality of output schedules. As a result, our method took 50 seconds to find a schedule for a six-day trip including 31 destinations. We also compared the system's output schedules including 15, 16 and 17 destinations with the optimal schedules found by a full search algorithm. We confirmed that our proposed method could calculate solutions within 1 % differences from the optimal ones.

2 Related works

There are several personal navigation systems for sightseeing and entertainment purpose. In [1], a navigation system which navigates user according to contexts (position, time and environment) is proposed. In [4], a system which provides sightseeing information through mobile terminal is proposed.

In [11], Rehrl et al. propose an integrated navigation system with wide variety of functions which navigates user over the whole schedule of sightseeing. It works in both indoor and outdoor environments, and is capable of providing various sightseeing plans, and tells users where and how to change transportation to get to a destination.

In [9], a navigation system named "This Izu Navi" is proposed. The system is developed aiming at activating sightseeing industry in Izu region in Japan. It is capable of navigating user through terminals installed at railway stations, besides ordinary PCs and mobile phones. It provides information regarding to sightseeing spots, maps, transportations and weather through web interface.

In [10], a navigation system "HOKO-NAVI" for handicapped and non-handicapped persons is proposed. This project aims at expanding barrier-free region around railway stations.

All of these existing navigation systems do not have functionality to make a touring schedule including two or more destinations.

There are methods to find a route including two or more destinations. In [7], an algorithm to find a route between two points via one of given points (which are, for example, gas stations or restaurants) is proposed. This algorithm cannot be applied directly to tour schedules, since it cannot find a route traveling multiple destinations taking user's preferences and time restrictions into account.

There are several methods proposed to solve Vehicle Routing Problem. In [5], Guo et al. extended the problem so that existence of clients and requests are given as probabilities, and time restrictions are given as soft time windows. They also propose a GA-based algorithm called HEX (Heuristic Edge-Based Crossover) to solve this problem. However, algorithms for Vehicle Routing Problem and derived problems are usually putting emphasis on delivering products to accumulation point and clients, and thus not suitable to be applied to calculating tour schedules over multiple days.

3 Outline of the proposed method

In this section, we present outline of our navigation system "P-Tour"[8]. Then, we explain the technical challenges of scheduling across multiple days. Finally, we describe techniques employed in the proposed method.

3.1 Outline of P-Tour

When a tourist makes a sightseeing schedule, he/she usually wishes to visit destinations as many as possible. Time restrictions (e.g. opening hours of store), stay time, moving cost, moving time and admission fee have to be taken into account besides the tourists' preferences for each destination. If there are too many sightseeing spots and one cannot visit all of them, he/she must select which spots to visit.

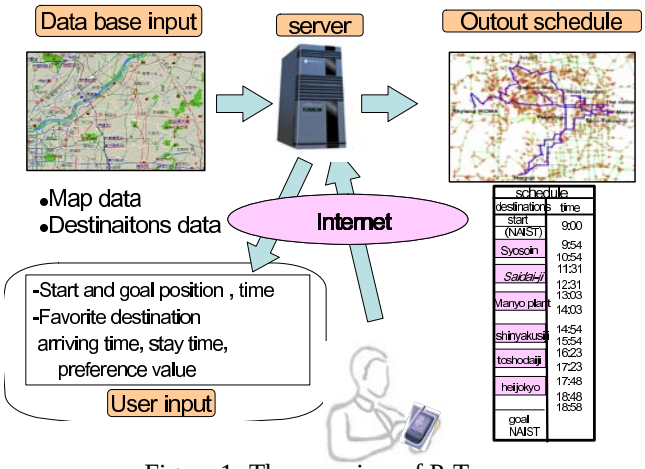

Figure 1: The overview of P-Tour

P-Tour provides a functionality to automatically find an efficient schedule taking all factors above into account. User just inputs starting point, time and relative importance among spots. Then, P-Tour calculates and shows a schedule instantly. The schedule is composed of a route with sightseeing spots and a time table. P-Tour also provides navigation functionality through mobile terminal on the way of the tour.

P-Tour is composed of client modules which run on mobile terminals, and a server module which runs on a PC connected to the Internet, as shown in Figure 1. The server module has data including road map and destinations.

3.2 Difficulty in the case of over multiple days

P-Tour is basically designed to compose a schedule for a single day sightseeing. In the case of sightseeing across multiple days, since the number of destinations increases, it is harder to find an efficient schedule at a practical time. P-Tour can be used to find a multiple day schedule, by calculating schedule day by day. Since this approach is close to greedy method, destinations with high priority are likely to be placed in the beginning of the schedule. Also, accommodation places have to be specified manually by user. In order to decide accommodation places, we need to take into account many factors, including costs, amenity, and the number of nearby sightseeing spots.

P-Tour can calculate and re-calculate one day schedule in a few seconds. However, since scheduling over multiple days is complex, the system needs more time to find a schedule. Since user usually changes the search condition multiple times, time for recalculation have to be shortened.

3.3 Ideas and functions of proposed method

In this paper, we propose a function to efficiently find a schedule across multiple days.

To realize function, we introduce the following techniques: (i) to divide the target sightseeing area into sub areas, and (ii) to utilize GA with certain loci in chromosome to only represent accommodation places.

4 Proposed method

In this section, we describe the overview of our proposed method, definition of the problem to find a tour schedule across multiple days, and idea of the method which solves the problem.

4.1 Overview

The proposed method basically calculates a sightseeing schedule which maximizes sum of preference values of visited destinations. Preference values are the degree representing to what extent user hopes to visit the place. System has default preference values for destinations, so that users can use the system without specifying the values. User can only input preference values for favorite (or not favorite) destinations.

Compared to the case making a single day sightseeing schedule, finding an efficient schedule for multiple day sightseeing is harder, due to increased sightseeing spots and widened search space. In the proposed method, the region including sightseeing spots is divided into sub areas, and the number of areas to which user can visit in one day is limited to two in order to reduce the search space. If these areas are too large, the search space becomes also too large and thus calculation takes long time. If they are too small, user cannot visit sufficient number of destinations in each day. Thus the sizes of areas have to be set appropriately. In the proposed method, users can adjust the sizes of areas by themselves. In the evaluation experiments, we set each area as one prefecture. In the proposed method, user selects an area for accommodation of each day. Our navigation system with the proposed method chooses accommodation place for each day so that the sum of preference values of accommodation places and visited destinations is maximized. Also, user can specify accommodation places directly.

When sightseeing schedule is made manually, users sometimes decide an accommodation place for each day first, and then decide sightseeing places to visit near the accommodation place. However, choice of accommodation places greatly limits sightseeing places to visit. Another difficulty to decide multiple day sightseeing schedule is that sightseeing places at near the end of the schedule is influenced by places at near the beginning. In the proposed method, the whole multiple day schedule is divided into subschedules of each day in order to reduce the influence of the time to move between destinations. In order to obtain a schedule in which destinations near accommodation places are visited before or after accommodation, destinations are managed per area, not per day. Also, in the method, the number of visited areas in each day is limited to suppress generating schedules to visit destinations in zigzags.

4.2 Definition of multiple day schedule planning problem

4.2.1 Input

Input I is composed of database input and user input.

$<$ **database input**$>$

Database input consists of road map and data of destinations. Road map is given as a directional graph $G = (V, E)$. Each edge is assigned a length. The shortest distance between two given vertices v_1 and v_2 is referred to by $dist(v_1, v_2)$. Destination data is given by $D = d_1, ..., d_k$.

Each destination data is given by a tuple of following properties.

- Corresponding vertex $v_i \in V$.
- d_i: destination ($1 \leq i \leq |D|$).
- aco_i: Accommodation flag. If it is an accommodation place, this value is 1. Otherwise 0.

$<$ **User input** $>$

As user input, the number of sightseeing days, starting and returning date and time, preference values of destinations, and sub areas are given.

User input is composed of items as follows:

- L: the number of sightseeing days.
- data on starting and returning locations: this data has following 6 items:
  - $pd_s, pd_g \in D$: The starting location of the first day and the returning location of the last day.
  - $area_l, (1 \leq l < L)$: Accommodation sub areas.
  - $at_l (1 \leq l < L)$: the maximum time to be able to delay to reach the accommodation place.
  - $ps_l, pg_l (1 \leq l \leq L)$: The time restrictions of each day at starting and ending locations.
- destination data : composed of sightseeing spots and accommodation places, and has following items.
  - $pre_i : D \rightarrow N : pre(d_i)$ is preference value of destination d_i.
  - rst_i: The time restriction of arrival at destination d_i (e.g. “before 12:00”).
  - dur_i: The time restriction of stay at destination d_i. (e.g. “30 minutes since reaching time” or “30 minutes from 12:00”).

4.2.2 Objective function

The object of this problem is to maximize the value of evaluation function $f(s, I)$. The details of evaluation function is explained in section 4.3.2.

4.2.3 Output

The output schedule is denoted as $s = (D, Stay)$, where $D = \langle d'_1, d'_2, ..., d'_i, ..., d'_{|k|} \rangle$ is the whole schedule, d'_i is a destination and $Stay = \langle stay_1, stay_2, ..., stay_{|D|} \rangle$ is a set of stay time at each destination.

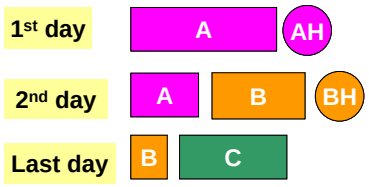

Figure 2: Chromosome for each day (e.g. 3 days tour)

4.2.4 Restrictions

The schedule s must hold the following restrictions:

- Tourist has to reach accommodation place before at_l.
- Tourist visits each sightseeing spot only once.
- Tourist is allowed to stay at least one accomodation place a day.
- Tourist stays at accommodation place in $area_l$ on lth day.
- The number of touring area must be two or less on each day.

4.3 Schedule finding algorithm

In this section, we propose a GA-based schedule finding algorithm.

4.3.1 Encoding of solutions

Schedule is represented by a list of destinations, as shown in Figure 2. This list is divided into destinations for each day. The destinations are arranged in visiting order. Each list of day has an accommodation place in its tail, except for list for the last day. Sightseeing spots are arranged in other position of list.

(1)Dividing into areas

In example shown in Figure 2, A, B and C indicate genes for area A, area B, area C respectively.

AH and **BH** are genes for accommodation place in area A and B respectively. In this case, tourist travels area A in the first day, lodges at somewhere in area A, and continues traveling in area A at the beginning of second day, and moves to area B.

By this design of chromosome, we can obtain a schedule with one accommodation place for each day, without zigzag path.

(2) Accommodation places and sightseeing spots

In our method, each gene represents an accommodation place or a sightseeing spot. Tourist can visit these spots as many as possible, however he/she must not visit the same spots more then once. Also, these spots may have time restriction for visit. If tourist cannot reach those spots within time restriction, value of evaluation function is not increased.

Accommodation places represent hotel, and other lodging places. These spots are used only for accommodation, and tourist has to arrive there before at_l (e.g. A.M.0:00). After that, tourist has to stay after at_l. Then tourist cannot stay two more accommodation places per day. But, tourist can stay at the same accommodation place twice or more. This means that if a tourist travels same area over multiple days, one can use same accommodation place for these days.

4.3.2 Genetic operator and detail of algorithm

The operations of genetic algorithm are composed of generation of initial population, selection, crossover, mutation, and local search. The flow of these operations with details is shown below. N denotes the number of individuals (candidate solutions) in population, and I denotes the number of iterations. Each individual has one chromosome.

Details of the algorithm are as follows.

(1) Generation of initial population: N chromosomes are randomly generated for individuals in initial population.

(2) Evaluation function: Evaluation function calculates the quality of schedule s according to input I. In our method, evaluation function $f(s, I)$ is represented as follows:

$$\begin{aligned} f(s,I) = \quad & \alpha \textstyle\sum_{j=1}^{|D|} pre_j \cdot timeflag(d_j) \\ & -\beta \textstyle\sum_{j=1}^{|D|-1} dist(d_j, d_{j+1}) \\ & -\gamma \textstyle\sum_{l=1}^{L} terminaltime(pg_l) \end{aligned} \quad (1)$$

The distance between two destinations is calculated by A* algorithm. This distance is denoted as $dist(d'_j, d'_{j+1})$. Moving time between two destinations is calculated from moving distance and speed. Tourist's speed is denoted as $speed(d'_j, d'_{j+1})$. We assume that tourist stays at destination d'_j for $stay_j$ minutes (here, $stay_j$ satisfies dur_j). When tourist moves from d'_j to d'_{j+1} and prediction time of arriving at destination d_j is t_j, $t_{j+1} = t_j + stay_j + \frac{dist(d'_j, d'_{j+1})}{speed(d'_j, d'_{j+1})}$. Function $timeflag(d_j)$ returns 1 iff time restrictions on d_j are satisfied. Function $terminaltime(pg_l)$ returns the value of delay from terminal time restriction for each day. Here, α, β, and γ are constant values. α is weight value for sum of preference values of destinations included in D. However, if time restrictions on a destination are not satisfied, the corresponding preference value of the destination is not added to f. β is weight value for total moving distance. The schedule with long path has negative evaluation value. By this term, our method can plan schedule with shorter path. γ is weight value for delay from terminal time restriction per day. By this term, our method can plan schedule with shorter delay.

Also, schedule must satisfy restrictions discussed in section 4.2.4. If schedule does not satisfy one or more restrictions, evaluation function returns 0.

(3) Selection: We use tournament selection. At first, two individuals are selected randomly from population. The individual with better value is saved for next generation. This process is repeated until necessary number of individuals for population are selected.

We use elite preservation strategy to always save the individual with the best evaluation value (elite individual).

(4) Crossover: To reduce probability of destroying chromosome structure at each end, we use two points crossover. Since ordinary two points crossover destroys chromosome structure discussed in section 4.3.1, we preserve the structure by the following method.

1. Crossover points are selected randomly in the same areas of parent1 and parent2.

2. Front part of parent1 and rear part of parent2 are connected to generate an offspring individual.

Also, destination to delete is chosen as follows: (a)duplicate destinations are deleted. (b)if zigzag path (e.g. from area A to area B to area A) is found, destinations of the first area are deleted.

(5) Mutation: Some destinations are inserted, exchanged and deleted randomly, for each chromosome. If multiple loci of a chromosome have the same destination, one of them is deleted at random. Accommodation places are replaced randomly.

(6) Local search: Mutation is applied to elite individual, and if evaluation value increases, its chromosome is updated. Otherwise, the change is discarded. This process is repeated for predefined number until evaluation value increases.

(7) Repetition: Algorithm repeats from step 2. to step 6. until terminal condition is satisfied.

5 Experimental Results

To evaluate our method, we experimented and evaluated in terms of (1) the quality of solution and (2) scalability depending on the number of touring days.

Using the instance composed of 6 prefectures in Tohoku Japan including 108 destinations (numeric map 25000 issued by the Geographical Survey Institute), we executed the proposed algorithm on general PC (Pentium M 2.0GHz,512M Memory, Windows XP pro.). We divided map into sub areas per prefecture and assumed that tourist moves 40 km/h speed by car. Also, we set weighting values that $\alpha = 100, \beta = 1,$ and $\gamma = 10$, the number of individuals is 300, and the number of iteration is 1000. We call these 2 parameters (individuals, iterations) "standard parameters".

5.1 Evaluation on the quality of solution

To evaluate the quality of solution calculated by our method, we compared our solution with optimal solutions. For the experiment, we used the road map of 3 prefectures (Aomori, Akita, Iwate). In addition, we set 15 sightseeing spots and 3 accommodation places per prefecture. We set preference values of all the sightseeing spots to 1, and stay time to 60 minutes. Under above conditions, we calculated 3 days schedules by proposed algorithm. In this experiment, the following instance was used: At first, tourist starts Tappizaki (A6 in Fig. 3) in Aomori prefecture. He lodges at Akita and Aomori respectively, and arrives at terminal Jodogahama (I13 in Fig. 3) in Iwate prefecture.

Table 1: Comparison with optimal solutions

destinations	error rate (%)	computation time (proposed method)	computation time (optimal solutions)
15	0.7	19.6 sec	36 min
16	1.0	19.9 sec	3 hour
17	1.1	20.4 sec	17 hour

Error rate between calculated solutions and optimal solutions, and computation time are shown in Table 1. These results are average values of 10 trials.

In all cases (15, 16, and 17 destinations), our method could find schedule with error rate less than 1%. So, we could confirm that proposed method has high quality solutions. Also, computation time of optimal solution to the instance with 17 destinations is 17 hours, while we could get it in about 20 seconds by our method.

5.2 Scalability according to the number of touring days

To evaluate how big instance can be solved by our proposed method, we experimented about scalability according to the number of days L, using the instance composed of 6 prefectures (Aomori, Akita, Iwate, Yamagata, Miyagi, and Fukusima) in Tohoku Japan, and set 15 sightseeing spots and 3 accommodation places to each prefecture. Stay time of each spot is 60 minutes. The sum of the number of destinations is $6 \times (15 + 3) = 108$. Preference values of sightseeing spots are one to three,

We show the number of touring destinations, total moving distance, delay at terminal in last day, and computation time for each L in Table 2. In the table, "destinations" shows the number of destinations included in the calculated schedule. These values are average values of 10 trials. In this experiment, we used standard parameters. The computation time of proposed method linearly increases according to the size of instance. We consider that it is caused by the time to compute the evaluation function. Also, delay time to accommodation places is 5 minutes or less through the cases, so quality of delay time is good enough. We wanted to compare the quality of solutions calculated by our method with the optimal ones for big size instances. However, it was impossible to find the optimal solutions by full search when the number of destinations is big. In genetic algorithm it is known that we can get higher quality solutions as we increase the numbers of individuals and iterations of GA operations, although it spends much memory and computation time. Then we compared proposed method (standard parameters) with big parameters' one (1000 individuals and 5000 iterations). We show evaluation values and rates for standard parameters case and big parameters case for each L in Table 3. These values are average values of 10 trials. In all cases, differences between standard parameters case and big parameters case are less than about 5 %. This result suggests us that proposed method with standard parameters can calculate enough good schedules at a practical computation time.

Finally, we show the schedule calculated with our method in Figure 3.

Table 2: Schedule properties and computation time

L	destinations	moving distance	delay	computation time
3 days	15	441km	none	23.2 sec
4 days	19	598km	none	32.3 sec
5 days	25	747km	2 minutes	40.8 sec
6 days	30	890km	5 minutes	51.2 sec

Table 3: Result of standard parameter compared with big parameters

L	evaluation value (standard parameters)	evaluation value (big parameters)	error rate (%)
3 days	2264	2359	4.1
4 days	2492	2619	4.8
5 days	2879	3037	5.2
6 days	3224	3435	6.2

Each alphabet shows sub-area (prefecture). A is Aomori, B is Akita, Y is Yamagata, F is Fukushima, M is Miyagi, and I is Iwate. Also, alphabet H shows accommodation places. Each number shows the serial number on database.

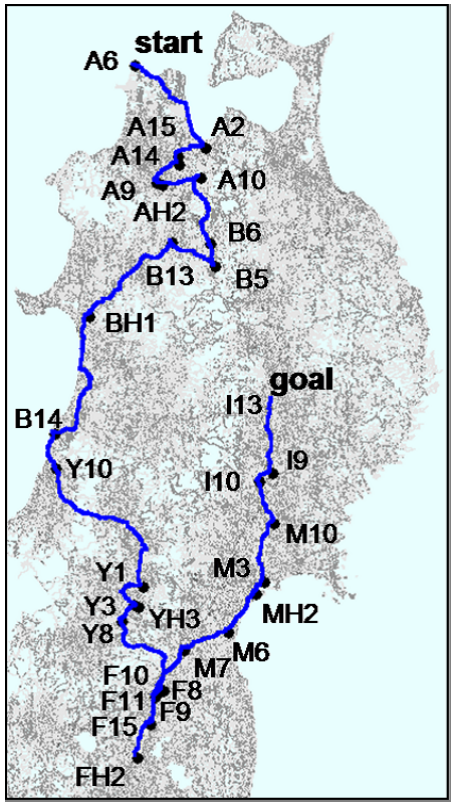

Figure 3: Sightseeing path of 6 days schedule

6 Conclusion

In this paper, we proposed the function to find schedule over multiple days to extend our navigation system called P-Tour. We defined the problem to make a schedule over multiple days, and proposed a genetic algorithm to solve the problem.

We implemented and evaluated the proposed method using Digital Map 25000 issued by the Geographical Survey Institute of Japan and data of 108 sightseeing spots in 6 prefectures in Tohoku region in Japan. We confirmed that the proposed method can find a schedule including 31 destinations in 50 seconds.

We are planning to improve the proposed method so that subdividing region is performed automatically according to user's preferences.

REFERENCES

[1] Baus, J., Krüger, A., Wahlster, W., "A Resource Adaptive Navigation System", in Proc. of IUI2002: International Conference on Intelligent User Interfaces 2002, ACM Press, New York, 2002.

[2] Baus, J., Kruger, A. and Wahlster, W. "A resource-adaptive mobile navigation system", in Proc. International Conference on Intelligent User Interfaces, San Francisco, 2002, pp. 15–22.

[3] Butz, A., Baus, J., Krüger, A., Lohse, M., "A Hybrid Indoor Navigation System", in Proc. of IUI2001: International Conference on Intelligent User Interfaces 2001, ACM Press, New York, pp. 25–33, 2001.

[4] Cheverst, K., Davies, N., Mitchell, K., and Friday, A., "The Design of an Object Model for a Context-Sensitive Tourist Guide", Computers & Graphics Journal, Vol. 23, No. 6, pp. 883–891, 1999.

[5] Guo, Z.G. and Mak, K.L., "A Heuristic Algorithm for The Stochastic Vehicle Routing Probrems with Soft Time Windows", Cogress on Evolutionary Computation, Vol. 2, No. 5, pp. 1449–1456, 2004.

[6] JTB, package tour (in Japanese) http://www.jtb.co.jp/kokunai/PKG/

[7] Kanoh, H. and Nakamura, N., "Route Guidance with Unspecified Staging Posts using Genetic Algorithm for Car Navigation Systems", IEEE Conference on Intelligent Transportation Systems, pp. 119–124, 2000.

[8] Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K. and Ito, M., "P-Tour: A Personal Navigation System for Tourist", in Proc. of 11th World Congress on ITS Nagoya, 2004.

[9] Miyauchi, H., Shiga, K., Omoto, M., Hamanaka, T. and Kano, K., "Style of providing information on tourist resort", World Congress on Intelligent Transport Systems, pp. 3146–3153, 2004.

[10] Sakagawa, M., Takeuchi, A., Nishio, I. and Inoue, T., "Pedestrian navigaion system in Nagoya", World Congress on Intelligent Transport Systems, pp. 3373–3380, 2004.

[11] Rehrl, K., Leitinger, S., Bruntsch, S. and Mentz, H. J., "Assisting orientation and guidance for multimodal travelers in situations of modal change", IEEE Intelligent Transportation Systems Society, Austria, 2005.