View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NAIST Academic Repository

A Personal Tourism Navigation System to Support Traveling Multiple Destinations with Time Restrictions *

Atsushi Maruyama, Naoki Shibata, Yoshihiro Murata, Keiichi Yasumoto and Minoru Ito
Graduate School of Information Science, Nara Institute
of Science and Technology, Ikoma, Nara 630-0192, Japan
{atsu-mar,n-sibata,yosihi-m,yasumoto,ito}@is.aist-nara.ac.jp

Abstract

In this paper, we propose a personal navigation system (called PNS) which navigates a tourist through multiple destinations efficiently. In our PNS, a tourist can specify multiple destinations with desired arrival/stay time and preference degree. The system calculates the route including part of the destinations satisfying tourist's requirements and navigates him/her. For the above route search problem, we have developed an efficient route search algorithm using a genetic algorithm. We have designed and implemented the PNS as a client-server system so that the portable device users can use the PNS through the Internet. Experiments using general map data and PDAs show that our PNS can calculate a semi-optimal route almost in real-time.

1 Introduction

Recent progress of information technology brought us high performance potable computing devices, small GPS units and radio communication devices such as wireless LANs, PHS and cellular phones. Personal navigation systems (hereafter, called *PNS*) on those portable devices have been attracting people's attention.

There are a lot of researches on "e-tourism" which navigates tourists to sightseeing places using portable devices. In Europe, there are several research projects such as DeepMap [1] and CRUMPET [2]. DeepMap and CRUMPET propose frameworks based on multi-agent systems where each of the components of the PNSs corresponds to an agent such as a GIS, a tour planning, a travel information database, and comprehensive interface. CRUMPET provides personalized location aware navigation to users through user-friendly interactions. There are some other researches on Web based location aware navigation services[5].

In this paper, we propose a PNS which navigates tourists through multiple destinations efficiently, where a time restriction and a preference degree are specified to each of the destinations. In PNSs used for tourism, only showing the shortest route between two locations like car navigation systems might be unsatisfactory. Instead, it is desirable to treat requirements of tourists such as (1) an efficient route traveling multiple destinations, (2) time restrictions at those destinations such as arrival/departure/stay time considering entertainments, opening hours and so on, (3) a sudden route change, *e.g.*, a tourist may want to skip/add some destinations and delay/shorten the stay time at some destinations. Since the problem of searching routes with the above requirements is a superclass of TSP (traveling salesman problem) which is known to be NP-hard, finding the exact solution would take very high computation cost. So, in this paper, we develop a fast route search algorithm which computes semi-optimal solutions using a genetic algorithm (GA).

[3] proposes a GA-based algorithm which computes semi-optimal routes with given restrictions so that a tourist can add some unspecified intermediate destinations. However, this algorithm basically treats routes between two locations, and cannot be applied to searching routes including multiple destinations.

We suppose that tourists use portable computing devices such as PDAs or cellular phones capable of communication via the Internet. Since it requires large processing power to search a route, we have designed and implemented our PNS as a client-server (C/S) system where our route search algorithm is implemented on the WWW server. So, portable device users can use the PNS through the Internet.

2 Outline of our Navigation System

The proposed navigation system consists of the following two parts: (1)portable devices with GPS receivers and capability of communication via the Internet, such as cellular phones or PDAs, (2)a WWW server on the Internet.

The server executes the route search algorithm, which is

*This work is partly supported by 21st Century COE Program"Ubiquitous Networked Media Computing".

implemented as a Java Servlet. The server has map data used by the algorithm.

We suppose that our system is used as follows:

1. A user inputs an outline of the tour (corresponds to the user's requirements) to the server using a web browser in advance. The outline is defined as a set of destinations (including the start and the end points), where each destination consists of its name, the timezone during which the user wants to arrive at the destination, the stay time, and the preference value indicating how strong the user wants to visit the destination.
2. The server searches for the route including part of the destinations which satisfies the user's requirements and sends the result to the portable device.
3. The portable device displays the received route with arrival and departure time for each destination with the corresponding map data.
4. During the tour, the portable device continuously retrieves the user's position from its GPS receiver, and displays current position on the route.
5. When the system detects that the user is departing from the scheduled route, the portable device automatically communicates with the server. Then, the server searches for a new route, taking into account of the user's movement until that time. The new route is transferred to the client and the user is guided along the new route.
6. The user can also change the outline by adding or deleting destinations after the tour begins.

3 Route Search Algorithm

3.1 Basic Strategy

It is preferable that the navigation system can search an alternative route when a sudden change of the plan occurs during the tour. Thus, we have designed our algorithm to search an alternative route quickly.

To search a route under the restrictions in sections 1 and 2, we need the following two steps. The first step searches the shortest path between each pair of destinations. Dijkstra's algorithm and A* algorithm are frequently used for this purpose in car navigation systems on the market. A* algorithm is an improved method of Dijkstra's algorithm. It searches for a path using straight-line distance between two destinations, as a heuristic function. We use A* algorithm to search a path between two destinations.

The time to pass each path is calculated by dividing the length of the path by the moving speed of the user.

The second step determines the visiting order of the destinations. We use a GA, since it always retains multiple candidate solutions during search, and it is always possible to give approximate solutions regardless of search time. Also, the GA can compute multiple routes, so that the user can select a desirable route from the returned routes.

3.2 Outline of our algorithm

The input of our algorithm consists of (1) map data, (2) a set of destination data, and (3) the upper limit of the search time, denoted $search\_time$. The map data is given by a graph $G = (V, E)$. The set of destination data is denoted by $D = \{d_1, d_2, ..., d_n\}$. Each destination data d_i consists of five components, v_i, st_i, et_i, dur_i and $pref_i$, where

v_i	:	location in the map
st_i	:	starting time of the timezone
et_i	:	ending time of the timezone
dur_i	:	stay time
$pref_i$	:	preference value indicating how strong the user wants to visit a destination

For the above input, our algorithm outputs a route whose fitness value (explained later) is as high as possible. Calculation method of the fitness value is described later.

The algorithm is shown below.

Algorithm *MAIN*$(G, D, p_s, p_g, B, search\_time)$

```
1  Begin
2      // G is map data. D is a set of all destinations.
3      // p_s, p_g are starting point and goal point, respectively.
4      // B is an associative array which is a database of routes
       between predefined destinations.
5      // search_time is the upper limit of search time.
6      timer := 0;
7      for each (d_s, d_g) ∈ D × D do
8          if d_s = d_g then continue;
9          if B[d_s][d_g] is assigned then
10             path[d_s][d_g] := B[d_s][d_g];
11         else
12             path[d_s][d_g]
13             := srch_betw_2dest(G, d_s, d_g);
14     next ;
15     // Decide the traveling order using GA.
16     return srch_rt_cmb(D,route,p_s,p_g, search_time);
17 End
```

Algorithm *srch_betw_2dest*(G, d_1, d_2)

```
1  Begin
2      // G is map data. d_1 and d_2 are destinations.
3      This function searches and returns the shortest path be-
       tween d_1 and d_2 using A* algorithm.
4  End
```

Algorithm *srch_rt_cmb*$(D, route, p_s, p_g, search\_time)$

```
1  Begin
2      // D is a set of all destinations. route contains the shortest
       paths generated by any combinations of destinations in D.
       p_s, p_g are starting point and goal point, respectively.
3      This function searches and returns the route which in-
       cludes part of destinations in D and maximizes the fit-
       ness function described below as much as possible un-
       til search_time seconds of time elapses since search
       started, using GA.
4  End
```

3.3 GA-based algorithm for determining traveling order of destinations

In order to add or delete destinations in candidate solutions, a candidate solution is coded as a variable length list of destinations. The search procedure is as follows.

(1)Predefined number of candidate solutions are generated randomly. A fitness value is calculated and assigned to each candidate solution. The evaluation method will be explained later. (2)The *elite individual* is selected in al the candidate solutions. The elite individual is the candidate solution with the best fitness value. (3)Two candidate solutions are selected using the GA operator called the *tournament selection*. From these two solutions, two new candidate solutions are generated using the GA operator called the *one-point crossover*. The crossover point is chosen randomly, and redundant destinations are deleted. (4)A GA operator called *mutation* is applied to the newly generated candidate solutions. Our mutation operator consists of 2-opt and addition/deletion of a destination. 2-opt is the technique to swap the positions of two destinations selected at random in the selected solution. (5)Each candidate solution's fitness value is calculated. (6)Steps (2) to (5) are repeated until *search_time* expires.

Calculation of fitness values by preference values:

To evaluate each candidate route, the sum of the preference values of the destinations which satisfy time restrictions is calculated. Using this function simply as the fitness function may result in devious routes. This is because detour is required to satisfy time restrictions of some destinations. Therefore, we have defined the fitness function as follows.

$$fitness\ value = \alpha \sum_{i=1}^{k} pref_i$$
$$-\beta\ \frac{\sum_{i=1}^{k-1} dist[d'_i][d'_{i+1}]}{k},\ (pref_i \in d'_i)$$

Where d'_k is the k-th destination to travel. $dist[d_s][d_g]$ is the function to calculate the distance of between d_s and d_g, and α and β are constant values (weighted coefficients). In order to save the search time, we prepared a database consists of the distances between predefined destinations in advance. If some input destinations do not exist in the database, values for these destinations are calculated from the scratch.

4 Implementation of PNS as a C/S system

We have implemented our PNS as a client-server application. Since portable computing devices are poor in resources such as CPU power, battery amount and data storage, it will be difficult to let those devices have the entire map data or compute the route. Therefore, we have implemented the route search algorithm in a server computer located on the Internet which also keeps the entire map data as well as other navigation information. We use a portable computing device as a client which connects to the server via the Internet to send requests and to download the route information computed by the server.

Implementation of Server Program as Java Servlet:

We have implemented the server side program as a Java servlet. The servlet is invoked by the WWW server when a user accesses to the specified URL and inputs data in a form. Then it computes a semi-optimal route and sends back the result to the client as an HTML file including the following information: (1)Destinations to visit and their traveling order, (2)Estimated arrival time at each destination, (3)A link to the map (in JPEG format) around the current user's location where the route to the next destination is drawn.

5 Experiments

We have carried out several experiments to investigate (1) route search time and optimality of the output route, (2) re-computation time and (3) response time including communication.

server: The server including the route search algorithm is implemented as a Java servlet, and runs on an ordinary personal computer with Pentium4 2.4GHz CPU, 512MByte memory, GNU/Linux, Tomcat4.2 Servlet, and Apache.

client: We used SHARP Zaurus SL-C700 PDA and PHS card "b-mobile" as a client. Its actual data transmission speed is 20.71kbps on upload, 48.38kbps on download.

5.1 Route search algorithm

We have carried out some experiments with our system under the conditions shown in Table 1 where we used a map data with 3800 nodes.

Setting: We designated 12 and 13 destinations. Time restrictions were specified to four of those destinations. We searched the optimal route with setting described above. It took 30 minutes to search the optimal route with 12 destinations, and 390 minutes for the optimal route with 13 destinations.

We measured computation time of our route search algorithm which does not include communication time.

Results: The execution time, evaluation value of the output route and the evaluation value of the optimal route are shown in Table 1. Here, note that it takes extra time to compute the distance between two destinations which is not in the database. The entire route obtained by the algorithm is shown in Figure 1.

The average time spent in the route search is about 8.5 seconds if all destinations are contained in the database. When the user designated 17 destinations in the database and 3 destinations not in the database, the search time is

num. of designated dest.	num. of dest out of database	Fitness value	Optimal value	Error(%))	Search Time
12	0	3284	3438	5.5	8.5(sec.)
12	1	3284	3438	5.5	14.0(sec.)
12	3	3284	3438	5.5	27.4(sec.)
13	0	3231	3440	6.1	8.5(sec.)
20	0	-	-	-	8.5(sec.)
20	1	-	-	-	16.3(sec.)
20	3	-	-	-	47.3(sec.)

Table 1. Condition and result

about 47.3 seconds. In this case most of the time difference was spent by A* algorithm. Route search using A* algorithm is basically only required in the route calculation before tour, and not needed in re-computing unless the user specifies new destinations which is not contained in the database. Thus, we believe that these results are sufficient for practical use.

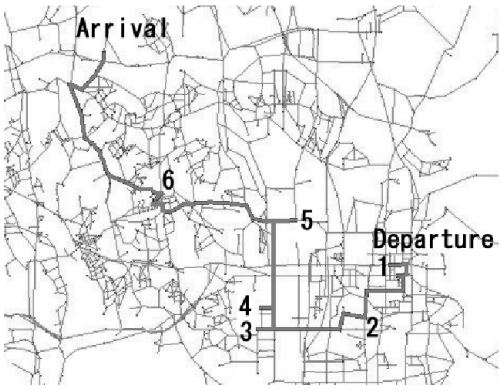

Figure 1. Output route of 6 destinations by our algorithm

5.2 Whole system with Internet

Next, we show the result of an experiment using the server and client.

In this experiment, the client sends a request of 2KB data size and receives 3KB of data as a response from the server.

It took 16.5 sec in average to get the response after sending a request in HTML. The time consists of route search (8.5 sec.) and data transmission (8 sec.). We believe that this response time would be sufficient for practical use. The transmission speed of the latest communication devices is faster than our PHS card. We can improve response time of our system by using other devices such as cellular phones.

6 Conclusion

In this paper, we proposed a personal navigation system which efficiently guides tourists to multiple touring destinations so that the tourists can visit each of primal destinations in a preferred timezone.

Our system consists mainly of the fast route search algorithm and the navigation functions. We developed a GA-based algorithm which searches for a semi-optimal route with multiple destinations. Our system was implemented as a C/S system: the route search algorithm is executed on a server, and each user is supposed to carry a portable device.

We evaluated our system using a digital map used for car navigation systems. As a result, we confirmed that our system can search a semi-optimal route in practical time.

We used a digital map for car navigation systems, and only treated moves by car. We are planning to extend our route search algorithm to handle multiple transports such as railroads, buses, and walks.

7 Acknowledgements

We thank the Navigation System Researchers' Association for providing the digital road-map format.

References

[1] Malaka, R. Zipf, A. "DEEP MAP - Challenging IT research in the framework of a tourist information system." ENTER 2000, Barcelona, Spain, pp. 15–27, 2000.

[2] IST, "CRUMPET Creation of User-friendly Mobile services Personalised for Tourism", http://www.ist-crumpet.org/

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

[4] Korf, R. "Real-Time Heuristic Search", Artificial Intelligence, Vol. 42, No. 2–3, pp. 189–211, 1990.

[5] Zipf, A. Malaka, R. "Developing Location Based Services (LBS) for tourism - The service providers view", Information and Communication Technologies in Tourism 2001. Proceedings of ENTER 2001, 8th International Conference. Montreal. Springer Computer Science. Wien, NewYork. pp. 83–92, 2001.