Design and Implementation of a Tour Planning System for Telematics Users

Junghoon Lee[1], Euiyoung Kang[2], and Gyung-Leen Park[1,⋆]

[1] Dept. of Computer Science and Statistics, Cheju National University,
[2] Dept. of Computer Education, Cheju National University,
66 Jeju-daehakno, Jeju-City, Jeju-Do, 690-756 Rep. of Korea
jhlee@cheju.ac.kr, euiyoung1@hanmail.net, glpark@cheju.ac.kr

Abstract. Aiming at providing an efficient tour schedule to tourists driving with a telematics device, this paper designs and implements an intelligent tour planning system based on the personalized tour recommender that may generate lots of destinations. To overcome the problem of long response time due to the computation of $O(2^n \cdot n!)$ complexity solver, we used initial set reduction, distributed computing via MPI-based Linux cluster, and finally Lin-Kernighan heuristic. An user interface was also implemented on a portable device using the utility of embedded operating system. Performance measurement results exhibit that the tour schedule can not only be offered to the user within 5 seconds when the number of TPOIs is less than 22, but also find a schedule whose satisfaction degree is very close to the optimal value.

1 Introduction

[1]In September 2004, *Jeju Telematics City* enterprise has been launched in Korea, aiming at not just testing telematics devices and services but also accelerating their instantaneous deployment[1]. With this enterprise, many of rent-a-cars in Jeju province have been equipped with in-vehicle telematics, opening a way to provide such services as tour guide, navigation, safety service, entertainment, and so on. From the viewpoint of an embedded system, the telematics is a computing device within a car, necessarily having a radio interface and a GPS (Global Positioning System) receiver. While the GPS receiver makes it possible to offer various location-based services to the driver, service contents can be delivered to end-users via the radio interface, CDMA (Code Division Multiple Access) in Korea. As Jeju is one of the most famous tourist places in East Asia, the telematics system indispensably puts importance on the efficient method of managing and providing tour information, while one of the most promising services is the personalized tour planning system[2].

The Jeju province has many kinds of unique tourist attractions, or TPOIs (Tourist Point of Interests) from now on, including beaches, volcanoes, cliffs,

⋆ Corresponding author.

[1] This research was supported by the MIC, Rep. of Korea, under the ITRC support program supervised by the IITA (IITA-2006-C1090-0603-0040).

mountains, subsidiary islands and so on, while hosting many kinds of tour activities including ocean sport, golf, hiking, and the like within a relatively small area. As so many diverse tour plans are possible, the tourist needs the assistance from an intelligent tour planning system. Tourists expect that the system presents the useful information with a minimum user input. Particularly, in terms of telematics we are targeting at, it is necessary to minimize the user-system interaction, since the device has limited user interface capability and users possibly access it while they are driving. The personalization feature can meet this requirement, as it can minimize the user interaction with the telematics. As a result, per-person preference input mechanism is more appropriate than the per-TPOI scheme most other recommender systems employ[3].

In the telematics service scenario, an end-user submits necessary data through the telematics device, whether the service is executed within the telematics device or by a backend server which can process even more sophisticated decision logic, handling large volume of data. In case of a recommender system, a user inputs his preference data to the telematics before he starts his trip, then the device transfers these data to the remote server. The server generates and returns a tour plan to the telematics, which stores and displays the route according to the current location of the vehicle. Not to mention, the user may dynamically change his schedule due to the traffic condition change, prolonged stay time at some tour points. To cope with this situation, it is desirable to keep the user data in the server. It will reduce the amount of data exchange via CDMA network, also enabling background preparation in the high-performance server.

Generally, the tour recommender system consists of two steps: First, the system selects the candidate TPOIs sorted by their ranks or scores determined by the system-specific criteria and algorithms. The second step builds the subset from candidate TPOIs to generate a corresponding tour schedule, considering calculated ranks, given time constraint, user location, and current traffic condition. Computing an optimal route for multiple TPOIs is a superclass of TSP (Traveling Salesman Problem) which is known to be NP-hard, the number of candidate TPOIs being the most critical factor to the execution time[4]. However, Jeju province has many lookout points that can make a tourist satisfied without spending much time, so the number of candidates for a tour schedule can get increased. Additionally, when just the personal preference is given, multiple TPOIs may be tied at the same rank. It is necessary to add all tie-ranked TPOIs to the candidate list.

Meanwhile, it is not uncommon to improve the computation speed by means of a parallel or distributed computing environment such as MPI-based Linux cluster. This architecture has many additional advantages such as reliability, scalability, and so on. Particularly, it fits for the location-based service that repeatedly processes a large amount of data. In addition, efficient heuristics for TSP have been already developed and easily applicable. Based on such assertion, this paper is to design and implement an efficient tour planning system capable of maximizing the degree of user satisfaction. The proposed system consists of a master and slaves of the cluster, each one having TSP solver installed, cooperatively yields a schedule that can be reachable within the given total tour length, for the TPOIs.

This paper is organized as follows: After issuing the problem in Section 1, Section 2 reviews related works. The data classification model is specified for the TPOI and user information in Section 3. Section 4 describes the TPOI recommendation method and tour planning system in detail. Section 5 demonstrates the performance measurement results of our system. Finally, in Section 6, the study is summarized with a brief description of future works.

2 Related Works

The recommender system is defined as follows[2]: Recommender systems are an attempt to mathematically model and technically reproduce the process of recommendations in the real world. First of all, this system filters off the unnecessary or less important information[5]. While there are various filtering methods, they can be classified into collaborative, content-based, and knowledge-based filtering schemes[6]. In the knowledge-based model, used most commonly, knowledge is expressed in the form of a detailed user model, a selection & suggestion model, and a rich description of the items to be suggested. Besides, there are also some interesting techniques such as data-mining, entropy, artificial intelligence, and agent schemes[7,8], while it is even possible to combine some of them to increase the accuracy of recommendation.

Maruyama et. al have proposed P-Tour system which generates a tour schedule according to the user input on (1) starting/returning locations and departure/arrival time of the tour, (2) candidate destinations of the tour, and (3) a relative importance degree of each destination[4]. They implemented a route search engine to obtain a semi-optimal solution using genetic algorithms. Afterward, this scheme was evolved to allow users to optimize their tour schedules under multiple conflicting criteria. However, in this approach, users do not only have to specify per-TPOI preference, but also the current traffic information is not considered, making it difficult to be used on the telematics system.

A TPOI recommender system has been proposed to provide personalized tourist information[9]. To provide a personalized recommendation, this system first classifies the related data into user information and TPOI information, then defines detailed attributes specific to each information category. Based on this data model, similarity between users and their preference are analyzed to decide a final recommendation. Though our paper is based on this system model, their system generates just the top-N recommendation result without taking into account the tour route or time constraint. In addition, the service lacks the capability of recommending via portable device such as a telematics or dynamically changing the schedule in the mid of tour.

The tour plan essentially starts from and ends at the customer's hotel as long as the customer does not change his hotel, so it can be considered to be a TSP Problem[1]. TSP is one of the widely studied NP-hard combinational optimization problems and can be described as follows: To begin with, let $G = (V, E)$

be a graph where V is a set of nodes and E is a set of links, and $C = (c_{ij})$ be the distance or cost matrix associated with E. TSP is to find the cheapest way of visiting all of the nodes and return to the starting point. Among plenty of heuristics to solve TSP, the most famous one is Lin-Kernighan's[10,11]. It is known for its efficiency in finding near-optimal results, while the core of this scheme consists of link exchange in a tour. A research web site provides diverse TSP solutions including Lin-Kernighan's in a source level, enabling users to download and integrate it to their own system[10].

3 Information Model

3.1 Data Classification

The first step of building a recommender system is to systematically organize the large amount of data. Our model consists of two major information components, namely, TPOI information and user information. The analyzed TPOI information can be classified as shown in Fig. 1. In the figure, the largest category groups are *objective* and *subjective* attributes, respectively. While the objective attribute represents general tourist information for TPOI such as location and name, subjective attributes express information subject to the character of respective individuals such as visitor's opinion on the particular TPOI. Even with the same tourist information, each user can choose different TPOIs. Among these, the average stay time is directly exploited by the tour planner.

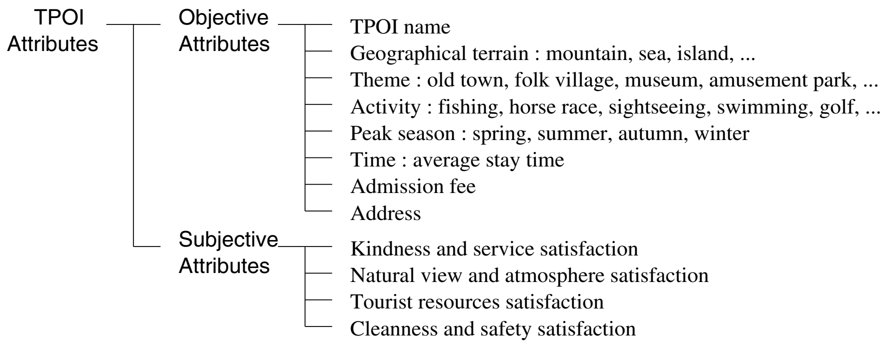

TPOI Attributes
Objective Attributes
TPOI name
Geographical terrain : mountain, sea, island, ...
Theme : old town, folk village, museum, amusement park, ...
Activity : fishing, horse race, sightseeing, swimming, golf, ...
Peak season : spring, summer, autumn, winter
Time : average stay time
Admission fee
Address
Subjective Attributes
Kindness and service satisfaction
Natural view and atmosphere satisfaction
Tourist resources satisfaction
Cleanness and safety satisfaction

Fig. 1. TPOI information

On the other hand, the user attribute, which users need to submit to the system, contains personality, attitude, demographic, and social factors. It is further classified into *preference* attributes and *general* attributes, respectively, as shown in Fig. 2. The preference attribute describes a personal requirement on this tour. For example, one prefers sightseeing on seaside, one wants to have a night tour, and one want to experience motor sports. In addition, the general attribute describes a user's general taste to his tour. For example, demographic factors include age, gender, and income. For the sake of minimizing the input of the user, general attributes are further classified into static and dynamic attributes according to whether the user has to specify data each time he starts another tour. Static attributes only need to be given at the initial registration phase because they don't change during the tour period. However, dynamic attributes need the user's input every time he wants a recommendation. After preference attributes are provisioned initially by the user, they are adjusted automatically with the tourist history of the user and the log-in information. A user can modify preferences at any time he wants.

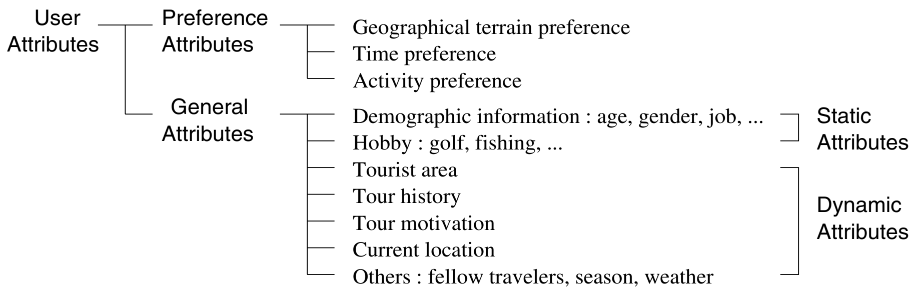


Fig. 2. User information

3.2 Computing Algorithm of the Similarity

The similarity computation scheme can be better described by an example based on the attribute classification proposed in the previous subsection. To begin with, let's assume that there are two TPOIs, namely, A and B, while individual preference attributes are given as shown in Table 1, where TPOI information is specified in terms of terrain, theme, and activity attributes. The *user preference table* entry contains priority field which ranks the degree of preference. The higher the priority value, or score, the more a user prefers.

Table 1. Given preference parameters

TPOI Information

	TPOI A	TPOI B
Terrain	Mountain	Island
Theme	Natural View	Folk Village
Activity	Golf	Sightseeing

User preference

Position	Mountain	Sea	City	Island
Priority	3	4	1	2
Theme	Folk village	Natural view	Park	Museum
Priority	4	2	1	3
Activity	Fishing	Sightseeing	Golf	Casino
Priority	4	3	2	1

With such settings, the user preference is to be compared according to the vector similarity defined in in Eq. (1), where the similarity calculation measures the cosine value between two vectors[5]. However, it has a problem when it computes the similarity between the distributed pattern of two vectors. To resolve this problem, the similarity criteria is modified to take into account both the length ratio of vectors and their size, as shown in Eq. (2).

$$VS(\overrightarrow{P_u}, \overrightarrow{O_t}) = cos\theta = \frac{\overrightarrow{P_u} \cdot \overrightarrow{O_t}}{|\overrightarrow{P_u}| \times |\overrightarrow{O_t}|} \tag{1}$$

$$VS(\overrightarrow{P_u}, \overrightarrow{O_t}) \times \frac{\overrightarrow{P_u}}{\overrightarrow{O_t}} = \frac{\overrightarrow{P_u} \cdot \overrightarrow{O_t}}{|\overrightarrow{O_t}| \times |\overrightarrow{O_t}|} \tag{2}$$

where $\overrightarrow{P_u}$ denotes the preference of user u, $\overrightarrow{O_t}$ the objective attribute of TPOI t, and $cos\theta$ the pattern similarity of u to t. Based on this similarity analysis method, the recommender system first compares the similarity between the TPOI objective attribute and user preference. If some user preference field is not given or not clear, analysis on similar users are performed to fill the undefined fields. In this step, unrelated information will be filtered out. For the more detailed description, refer to [9].

4 Recommender System

4.1 System Architecture

The overall architecture is shown in Fig. 3. To begin with, each telematics installed Microsoft Windows Mobile operating system and is connected to our system via CDMA interface using RAS (Remote Access Service) utility. Users can input their preference in the form of numbered value through the interface implemented in the telematics, the recommended result being shown also in this device. Additionally, the telematics device provides a basic map viewing functions such as zoom in-out, pan, and distance calculation based on the road network. The road network has only nodes and links, which are intersection and two end points of the road segment, respectively. In addition, along with a corresponding user interface, the device can perform A* path finding algorithm.

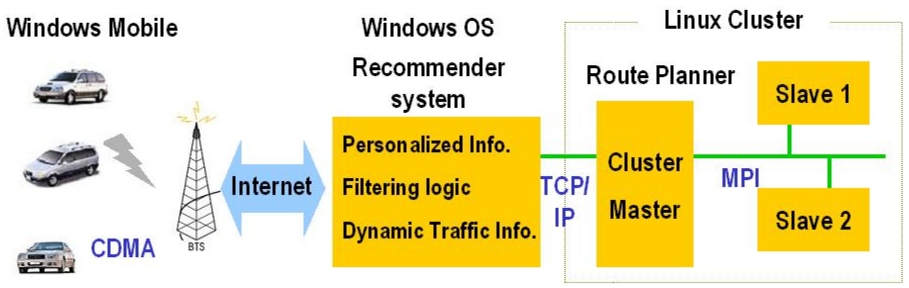


Fig. 3. System architecture

The recommender system has been built on top of general desktop Windows operating system and relational database which stores information on each personal preference and TPOI characteristics. It implements the filtering logic, as previously explained. Dynamic traffic information is available in Jeju area via the web interface to the Jeju ITS (Intelligent Transport System) or Jeju Taxi Telematics system[1]. After the recommender system selects the candidate TPOIs, they will be handed over to the tour planning server, or route planner. The main function of tour planning server is to generate a subset of TPOIs and their tour schedule which maximizes the satisfaction degree, considering the user-specified tour duration and traffic condition. We define the satisfaction degree of a set as the sum of the score of all TPOIs in the set. The planning server mainly plays a role of TSP solver, and as this procedure needs a great deal of computation, we build a Linux cluster consist of a master and 2 slaves. But more computing nodes can be added to our system just with a minor revision of the software.

4.2 Tour Plan Generator

To begin with, we define that a TPOI set has a feasible schedule when there exists a TPOI sequence $\{S_1, S_2, ..., S_n\}$ that meets the Ineq. (3).

$$\sum T(S_i) + \sum D(S_i \cdot S_{i+1}) + D(S_{i+1} \cdot S_0) \leq T_c \tag{3}$$

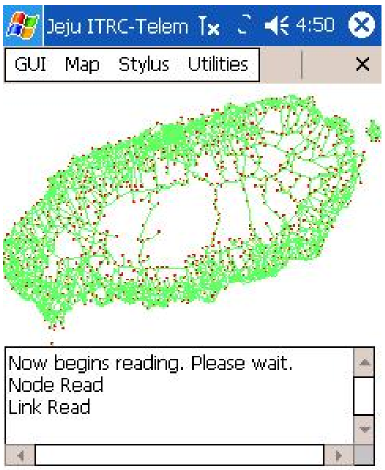
where $T(S_i)$ denotes the stay time at S_i, $D(S_i \cdot S_j)$ the driving time from S_i to S_j, and T_c the user-specified tour length. At the filtering step, candidate TPOIs are selected and sorted by their scores. If there exists a feasible tour schedule for the initial set, all TPOIs are selected for the final recommendation. Otherwise, the system should extract the appropriate subset that has a feasible schedule. A TPOI may be discarded if a tour schedule including it exceeds the given tour length, or if a tour schedule excluding it has a higher satisfaction degree. Furthermore, even if S_i has the higher score than S_j, S_i may be excluded when any set including S_i doesn't have a feasible schedule, while S_j has.

When the number of candidate TPOIs are too many, it may prolong the response time in generating a feasible schedule. Hence, it is necessary to remove the TPOIs that have little possibility to be selected for a final recommendation from the first. To this end, we select the initial subset of the TPOIs. From the sorted list, we choose TPOIs one by one until the sum of stay time is less than $1.5{\cdot}T_c$. In addition, if exists, the TPOIs whose scores are same or almost same with the last TPOI will be selected to the initial TPOI set. Our experience shows that this selection rarely misses the optimal route. If the number of candidate TPOIs is n after such initialization, we should inspect whether there exists a feasible schedule for 2^n subsets. Among the feasible ones, the set of the highest satisfaction degree is finally chosen for the final recommendation. However, the time complexity of TSP is $O(n!)$, where n is the number of visiting nodes, so the total complexity of tour planning system is estimated to be $O(2^n \cdot n!)$. As natural, there are some constraints that can remove the unnecessary computation. Namely, the system doesn't have to calculate TSP for a set, either when the sum of stay time of all element already exceeds the given tour length or when the satisfaction degree is less than the current maximum.

As the TSP overwhelms the computing time of tour planning system, we need an efficient heuristic for this problem. Lin-Kernighan algorithm is known to be one of the most efficient algorithms, while running a Lin-Kernighan algorithm needs the cost matrix having the driving time between every node pair. If we have n nodes, the cost matrix needs $n \times (n-1)$ times of A* calculations, while each execution time is dependent on the number of nodes and links in the road network. The road network of Jeju province consists of about 17,000 nodes and 30,000 links, so it takes a non-negligible time to compute this matrix but Linux cluster provides a cost-effective way to enhance the performance of such computation.

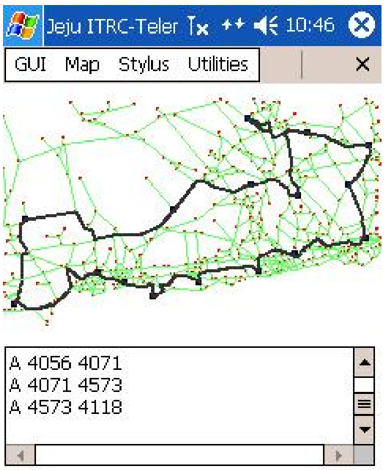
In the Linux cluster of tour planner system, the master node is equipped with 1.0 GHz Pentium IV CPU and 512 MB memory, while the slave node 700 MHz Pentium 3 CPU and 384 MB memory, and they are connected with 100 MBps Ethernet interface. In addition, the NETGEAR 8-port Fast Ethernet switch connects all of cluster nodes to build a private network. Finally, all nodes installed Redhat Linux version 9.0 and LAM-MPI version 7.1.2. The master initiates the cooperative computation by sending an MPI (Message Passing Interface) message[12]. Each node exchanges the necessary information such as candidate set, user-specified tour length, and partial result through the shared file system. If we make b_i, when set, indicate that S_i is included in the trial set, the task partition will be done as follows: The master takes the set if $b_{n-2}b_{n-1}$, namely, bits for the last two TPOIs, is 10, 11. The set of 01 and 00 will be processed at each slave. After each node calculates, the master collects the partial maximum to choose the final maximum.

Fig. 4 shows the execution result displayed in the PDA. Fig. 4(a) shows the main menu of our system interface including map viewing functions and GUI. After a tour planning system calculates a feasible schedule, it will return the tour schedule $\{S_0, S_1, ..., S_{n-1}\}$. The the system interface program will calculate A* algorithm along the received route to display the final route as in Fig. 4(b).



```
Now begins reading. Please wait.
Node Read
Link Read
```

(a) Main menu



```
A 4056 4071
A 4071 4573
A 4573 4118
```

(b) Plan generation

Fig. 4. Tour plan generation

5 Performance Measurement

Based on this system, we measured the performance of out planning system in terms of execution time, the probability of finding the optimal schedule, and the optimality of the tour schedule according to the number of candidate TPOIs as well as satisfaction degree. For the experiment, we have generated 14 set groups, each of which has 50 sets having same number of candidate TPOIs. The number of candidate TPOIs of respective groups ranges from 12 to 28. Each TPOI has its own rank distributing randomly from 1 to 10 as well as stay time exponentially distributing centered on two average, 1.0 and 0.2 for the lookout-type TPOI.

Fig. 5 plots the execution time of the proposed tour planning scheme according to the number of candidate TPOIs. This curve indicates that the tour schedule can be offered to the user within 5 seconds when the number of TPOIs is less than 22. The execution time of full search, which investigates every feasible schedule, grows beyond 1 minute, when the number of TPOIs is larger than 11. In addition, Fig. 6 shows the probability of finding optimal schedule. The probability means the ratio of the candidate sets the system finds the optimal schedule to all of the generated sets. As shown in this figure, when the number of TPOIs is less than or equal to 17, the probability of finding the optimal schedule is above 90 %.

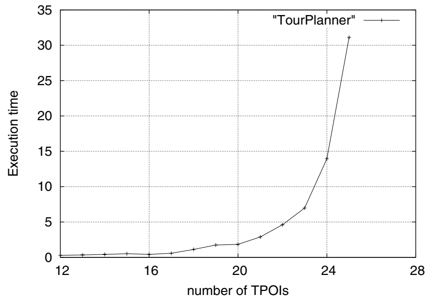


Fig. 5. Execution time measurement

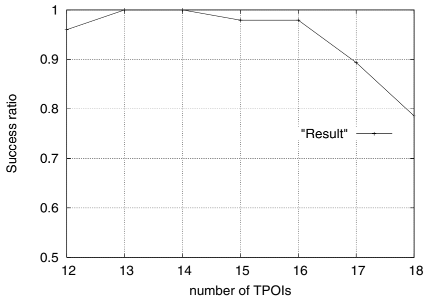


Fig. 6. Finding optimal solution

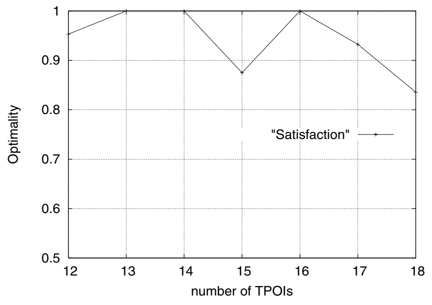


Fig. 7. Optimality vs. # of TPOIs

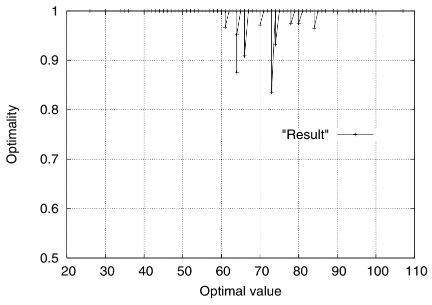


Fig. 8. Optimality vs. optimal value

Fig. 7 and Fig. 8 evaluate the quality of schedule generated by our system by measuring how close its satisfaction degree is to the optimal degree according to the number of TPOIs and optimal value. These experiments indicate that both factors have not so significant impact to the optimality when the number of TPOIs are less than 18. However, the optimality always exceeds 0.8 for those parameters.

6 Conclusion

This paper has designed and implemented an intelligent tour planning system capable of 1) selecting TPOIs based on a personalized information rather than per-TPOI preference, 2) generating a tour plan that maximizes the satisfaction degree for a tourist. To reduce the response time resulting from the calculation of $O(2^n \cdot n!)$ complexity, we used initial set reduction, distributed computing via MPI-based Linux cluster, and finally Lin-Kernighan heuristic. An user interface was also implemented on a portable device using the utility of embedded operating system. Performance measurement results show that the tour schedule can not only be offered to the user within 5 seconds when the number of TPOIs is less than 22, but also find a schedule whose satisfaction degree is close to the optimal value. We believe that this system can provide a useful information to tourists for Jeju province as a good example of telematics application. As a future work, we will keep finding a prospective application for the telematics network for the dissemination of telematics device.

References

1. Lee, J., Park, G., Kim, H., Yang, Y., Kim, P., Kim, S.: A telematics service system based on the Linux cluster. In: Shi, Y. (ed.) ICCS 2007. LNCS, vol. 4490, pp. 660–667. Springer, Heidelberg (2007)
2. Ponnada, M., Sharda, N.: A high level model for developing intelligent visual travel recommender systems, ENTER (2007)
3. Ricci, F., Werthner, H.: Case base querying for travel planning recommendation. Information Technology & Tourism 4, 215–226 (2002)
4. Maruyama, A., Shibata, N., Murata, Y., Yasumoto, K., Ito, M.: P-tour: A personal navigation system for tourism. In: Proc. of 11-th World Congress on ITS 2, pp. 18–21 (2004)
5. Breese, J., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Annual conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
6. Schubert, P., Koch, M.: The power of personalization: Customer collaboration and virtual communities. In: Proceedings of the Conference on AMCIS, pp. 1955-1965 (2002)
7. Nasraoui, O., Petenes, C.: An intelligent web recommendation engine based on fuzzy approximate reasoning. In Proceeding of the IEEE International Conference on Fuzzy System 2, 1116–1121 (2003)
8. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M.: Combining content-based and collaborative filters in an online newspaper. In: Proceedings of the ACM SIGIR Workshop on Recommender Systems (1999)

9. Kang, E., Kim, H., Cho, J.: Personalization method for tourist point of interest (POI) recommendation. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4251, pp. 392–400. Springer, Heidelberg (2006)
10. http://www.tsp.gatech.edu/concorde.html
11. Goldberg, A., Kaplan, H., Werneck, R.: Reach for A*: Efficient point-to-point shortest path algorithms. MSR-TR-2005-132. Microsoft (2005)
12. Pacheco, P.: Parallel Programming with MPI. Morgan Kaufmann Publishers, San Francisco (1996)