Optimal Sequenced Route Query Algorithm Using Visited POI Graph

Htoo HTOO, Yutaka OHSAWA and Noboru SONEHARA

Abstract: The optimal sequenced route (OSR) query performs a critical role in finding the most economical route for a trip, for instance, location base services. OSR finds the minimum-length route, starting from an origin location and passing through a number of locations or points of interest (POIs), following the prespecified route sequence. An OSR query on road networks tends to expand an extremely large number of nodes, which leads to an increase in processing time. To reduce the number of node expansions, we propose a visited POI graph (VPG) to register a single found path that connects neighboring POIs. We also perform experiments to show the effectiveness of our method compared with an original approach, in terms of the number of expanded nodes and processing time.

Keywords: Optimal Sequenced Query, Road Network, Location Based Services

1. Introduction

The optimal sequenced route (OSR) search method has been proposed in recent years. It has been used for trip planning in several applications, such as location-based services (LBSes) and car navigation systems. The OSR finds the minimum-length route, starting from an origin location and passing through a number of locations or points of interest (POIs), following a prespecified route sequence.

Figure 1 shows an example of an OSR query. You are currently at the "current position," and the final destination of your trip is home, which is labeled "destination." During the trip, you want to stop at a bank to withdraw money, then at a French restaurant to have dinner, then at a movie theater, and finally return home. Although there may be many banks, restaurants, and movie theaters in the area, the OSR query chooses one from each category according to the specified sequence, in order to minimize the total cost of the trip. However, in this paper we measure the cost according to the total trip distance.

Figure 1: Optimal Sequenced Route

The OSR query was first proposed by Sharifzadeh et al.[1]. They proposed several algorithms to find the k-OSR in both on the Euclidean distance and the road-network distance. In their proposed method, progressive neighbor expansion (PNE) algorithm is used for query on road networks. The computation cost for the nearest neighbor (NN)object on road networks can drastically differ from the cost on the Euclidean distance[8]. Euclidean distance-based algorithm can

Y.Ohsawa : School of Eng., Saitama University
Shimo-okubo 255, Sakura-ku
Saitama 338-8570, Japan
Tel. +81-48-858-3717
ohsawa@mail.saitama-u.ac.jp

use simple spatial index structures, such as, R-tree [2], to narrow the search space. In this paper, we propose efficient algorithms for an OSR search on road networks. In our trip planning method, the starting (usually the current position) and the destination positions of the trip are provided explicitly. When the destination is specified explicitly, we can adopt an efficient A* algorithm and bidirectional search [4] for an OSR search.

2. Related Work

The OSR query was first proposed by Sharifzadeh et al.[1]. They proposed several algorithms for an OSR search to operate on the Euclidean distance. The authors also proposed a more efficient algorithm called the R-LORD (R-tree-based LORD). However, these algorithms cannot be adapted directly to the road-network distance. For a road network, they proposed the PNE[3]. During almost the same time, Li et al. proposed the trip planning query (TPQ) [5].The TPQ is similar to an OSR query, but with only the types of visiting POI categories specified. For the practical use of the TPQ, Ohsawa and Fujino [6] proposed the simple trip planning query (STPQ) algorithm to be applied in car navigation systems. It finds the route from the current location to the destination by passing a POI.

3. OSR AND PNE

Let U_i be a category of the POI to be visited, and M be a sequence of U_i to specify the visiting order. That is, $M = \{U_1,U_2,..,U_m\}$, and here m is the length of M ($m = |M|$). The OSR query finds k optimal sequenced routes from the starting point S to the destination E, visiting each POI belonging to $U_i(1 \le i \le m)$ one after another, according to the given sequence M. As we mentioned in the previous section, Sharifzadeh [3] proposed a PNE algorithm that can be applied to OSR queries on road networks for a specified sequence of M. Their original OSR query starts from the starting point S and finds a POI in U_1 as the first visit, then finds the next POI in U_2 as the second visit. This is repeated until the last POI in U_m is visited. They then determine the shortest SR(Sequenced Route). They indicated that the final destination E can be added into the sequence as the last-visited POI category that contains only one POI (E).

In this paper, we deal with the case that S and E are given explicitly. The partial sequenced route (PSR) is the shortest route from the starting point S to one of the POIs in U_i, by passing through the POIs one after another choosing from U_j $(1 \le j < i)$ on the way according to the given sequence. SR is the total routes from S to E, visiting the POIs according to the given sequence M.

In PNE, k-NN searches started from every visited POI cause enormous node expansions in the road network. The k-NN searches are performed independently, and then a node in the road network is expanded several times, which causes inefficiency in the PNE algorithm. The proposed algorithm in the next section attempts to improve this inefficiency.

4. OSR Query Applying Bidirectional Search

A bidirectional search starts from S and E simultaneously under the control of one PQ. The search starting from S tries to find the POI according to the predetermined POI sequence M, and the search starting from E tries to find the POI in the reverse order of M; that is, $E \rightarrow U_m \rightarrow U_{m-1} \rightarrow ... \rightarrow U_1 \rightarrow S$. The search is terminated when the search paths from both origins meet at a POI. Every time a search encounters the next POI to be visited, the arrival to the POI from another origin is checked. The bidirectional search appears simple when only one shortest OSR is requested. However, when multiple k-OSRs are requested, there are some problems to be considered. In Section 5, we explain the problems and propose a solution.

5. Suppressing Duplicated Node Expansion

5.1 Control for the Duplicated Search

Both the abovementioned approaches start a new node expansion from a POI belonging to U_i toward a POI belonging to U_{i+1}, every time a POI belonging to U_i is found.

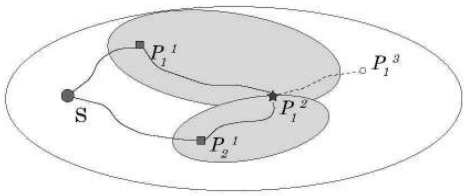

Figure 2: Path arrival to the same POI

In Figure 2, a search for a POI belonging to U_1 starts from S, and then the search finds P_1^1 and P_2^1 in that order. New searches for a POI belonging to U_2 start from P_1^1 and P_2^1 . Then, the search that started from P_1^1 finds P_1^2 as the second-visited POI, at which a further new search starts for a POI belonging to U_3. Later, the search starting from P_2^1 reaches P_1^2, and then another new search for a POI belonging to U_3 starts. However, these two searches will consequently find the same path (PSR) from P_1^2 to E. Therefore, we need to suppress this redundant node expansion because starting a further node expansion from the same POI that is the head of another path is not useful.

5.2 Visited POI Graph

When we need to find k-OSR ($k > 1$), late arrival PSRs can be a part of the SRs. In this case, we need to consider the late-arrival PSR. Simultaneously, we need to consider suppressing the duplicated node expansions, which gives the same result. To achieve these, we adopt a visited POI graph (VPG). The vertices of the graph correspond to detected POIs, including S and E, and the edges indicate the shortest path from the previous POI on the road network connecting the neighboring two vertices. The VPG graph is constructed according to the progress of the search.

6. Experimental Results

To evaluate the performance of the proposed methods, we conducted extensive experiments. We used digital road network data published by the Geospatial Information Authority of Japan (GSI), in the area of Saitama city, Japan. The road map consists of 268,245 road segments. We implemented the algorithms using the C# language. The computer used in the experiments was a Core 2 Quad 2.40GHz CPU with 4GB RAM, and the operating system was Windows Vista. We generated several sets of POIs in a pseudo-random sequence, with varying distribution density (p), which means the existence probability of a POI on a road segment.

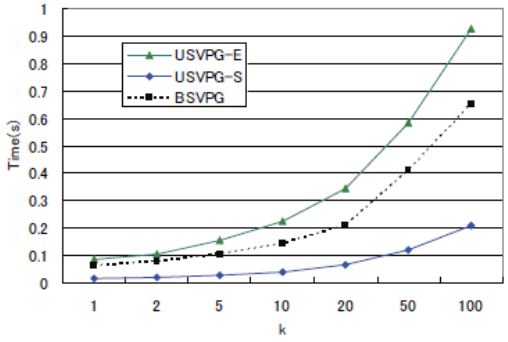

Figure 3: Comparison between USVPG and BSVPG

Figure 3 shows the results between the unidirectional search (USVPG) and the bidirectional search (BSVPG). This figure compares their calculation time among the unidirectional searches that start from S (USVPG-S), E (USVPG-E), and BSVPG. The density of the POI increases from the first (0.001), the second (0.002), and the last (0.01). In these cases, the USVPG-S is the fastest, the USVPG-E is the slowest, and the BSVPG is moderate. When the density distributions of POIs are not equal, the unidirectional search starting from the less-dense side always outperforms the other. The experiments using the other combinations showed the same tendency. From these results, we can say that the BSVPG shows the moderate performance among them.

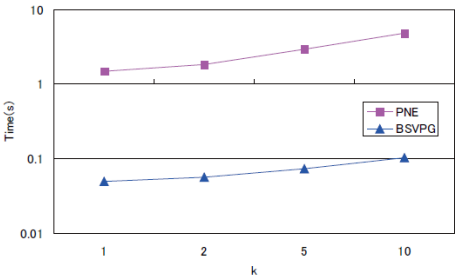

Figure 4: Time on 0.001→0.002→0.01

Figures 4 compares the performance between BSVPG and PNE according to the processing time. The experiments were conducted under three POI categories in which p = 0.001, p = 0.002, and p = 0.01, with shuffling of the order to be visited. The probability of the sequence used in Figure 4 is 0.001 → 0.002 → 0.01. As shown in this figure, the

BSVPG outperforms the PNE approximately 100 times.

Figure 5 compares the results of all combined patterns. In this figure, the pattern is shown by the three-digit number that corresponds to the three different POI densities: 1 corresponds to 0.001, 2 to 0.002, and 3 to 0.01. This figure shows the result of k = 10. As indicated by this result, the BSVPG always outperforms the PNE. According to the experiments varying k values, the calculation time becomes stable with increasing k, independent of the POI density patterns.

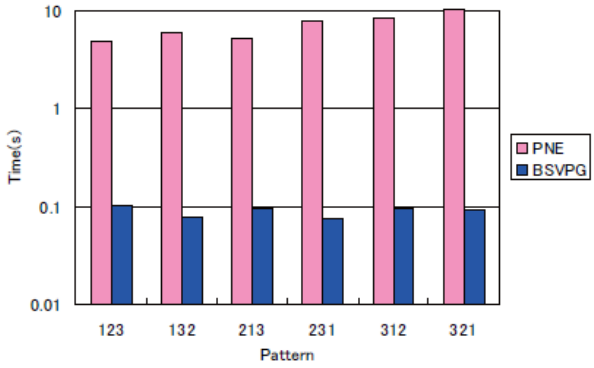

Figure 5: Performance Comparison in all Patterns

7. Conclusion

In this paper, two fast algorithms called USVPG and BSVPG have been proposed to search the k-OSR for the road-network distance. Both algorithms are controlled by the A* algorithm. The BSVPG searches POIs from S and E simultaneously. This paper also proposes the VPG to reduce multiple node expansions, which is unavoidable in trip planning. Experimental results confirm that the presented algorithm can search the OSR approximately 100 times faster than the PNE. The USVPG and BSVPG largely contribute to the improvement in the VPG. In the case when the POI density cannot be known in advance, the BSVPG can be a good selection. The BSVPG can be improved further. The search paths from opposite origins will meet on a road-network node between two neighboring POI categories. When the two searches meet each other, a path connecting the POIs, each of which belongs to the neighboring POI sequence in order, is fixed. The node expansion is then not necessary and will be terminated at this point. In future, we will study this modification, which can improve the search performance

Acknowledgement

This study was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), 21500093, and Japan Digital Road Map Association.

References

[1] M.Sharifzadeh, M.R.Kalahdouzan, and C.Shahabi. The optimal sequenced route query. Technical report, Computer Science Department, University of Southern Calfornia, 2005.

2] A. Guttman. R-Trees: a dynamic index structure for spatial searching. In Proc. ACM SIGMOD Conference on Management of Data, pages 47_57, 1984.

[3] M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi. The optimal sequenced route query. The VLDB Journal, 17:765_787, 2008.

[4] T. Ikeda, M.-Y. Hsu, H. Imai, S. N. H. Shimoura, T. Hashimoto, K. Tenmoku, and K. Mitoh. A fast algorithm for finding better routes by AI search techniques.
In 1994 Vehicle Navigation & Information System Conference, pages 291_296, 1994.

[5] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In Proc. SSTD 2005, pages 273_290, 2005.

[6] Y. Ohsawa and K. Fujino. Simple trip planning algorithm on road networks without pre-computation. IEICE Transactions on Information and Systems, J93-D(3):203_210, March 2010. (In Japanese).

[7] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In Proc. 29th VLDB, pages 790_801, 2003.

[8] M. Kolahdouzan and C. Shahabi. Voronoi-based K nearest neighbor search for spatial network databases. In Proc. 30th VLDB, pages pp.840_851, 2004.