A fewest-turn-and-shortest path algorithm based on breadth-first search

Yan ZHOU*, Weisheng WANG, Di HE and Zhe WANG

School of Resources and Environment, University of Electronic Science and Technology of China, Chengdu, P.R. China

(Received 26 August 2014; final version received 15 October 2014)

Many cognitive studies have indicated that the path simplicity may be as important as its distance travelled. However, the optimality of paths for current navigation system is often judged purely on the distance travelled or time cost, and not the path simplicity. To balance these factors, this paper presented an algorithm to compute a path that not only possesses fewest turns but also is as short as possible by utilizing the breadth-first-search strategy. The proposed algorithm started searching from a starting point, and expanded layer by layer through searching zero-level reachable points until the endpoint is found, and then deleted unnecessary points in the reverse direction. The forward searching and backward cleaning strategies were presented to build a hierarchical graph of zero-level reachable points, and form a fewest-turn-path graph ($G^*$). After that, a classic Dijkstra shortest path algorithm was executed on the $G^*$ to obtain a fewest-turn-and-shortest path. Comparing with the shortest path in Baidu map, the algorithm in this work has less than half of the turns but the nearly same length. The proposed fewest-turn-and-shortest path algorithm is proved to be more suitable for human beings according to human cognition research.

Keywords: fewest-turn-and-shortest path; breadth-first search; hierarchical graph

1. Introduction

Map direction for navigation is a basic and indispensable function in WebGIS services. In general, the shortest path algorithm is the core of route planning services. Such services are provided by almost all existing WebGIS platforms, e.g. Google, Bing, and Baidu Map. Actually, human could select routes by referring to many different criteria. According to statistics, over 20 different types of criteria existed, among which the shortest distance and least time were ranked most highly, while the fewest turn was Probably the most used (1, 2). Some researches on human cognition manifested that the complexity of route instruction may be as important as its length in navigation. Focusing on the application of verbal instructions in navigation, an investigation found that people prefer suboptimal routes that were potentially easier to describe or follow, although these routes may not be the optimal one in length (3, 4).

A concept named path simplicity was proposed in Ref. (5), which quantifies the complexity of verbal instruction of a path. In an unfamiliar environment, the more simple a path is, the less possibility of misleading it has (6). Utilizing massive GPS data in London, a comparison was carried out between two routing strategies, Manhattan distance minimum and Angular distance minimum which reflects the number of turns. The result showed that most people indeed follow the fewest-turn routes rather than the shortest-distance routes (7). Although many path optimizations focused on distance travelled minimum are not aiming at reducing path simplicity, this reduction is their byproduct. A path solution could prefer major road by incorporating road network knowledge, which has lower simplicity before optimizing (8, 9). Based on the topology representation of graph, an algorithm was presented to compute all fewest-turn paths using natural roads connectivity (10). This is a complicated solution by utilizing the breadth-first-search thought to compute the shortest topological distance ($D_t$) first, and then using the depth-first-search strategy to determine whether the current topological distance equals to $D_t$, finally obtaining one of the shortest-turn path, if the equation is established.

This paper presented a new algorithm to plan the fewest-turn-and-shortest path using the breadth-first search. The remainder of this work is organized as follows. Before introducing the fewest-turn-and-shortest path algorithm in Section 2, there are four concepts that will be defined. In Section 3, the correctness proof of presented algorithm in mathematics will be given. And then, the proposed algorithm performance will be evaluated by comparing it with the shortest path algorithm suggested by Baidu Map in Section 4.

2. The proposed fewest-turn-and-shortest path algorithm

2.1. Four concepts

Concept 1: Turn is a switching between two roads with different name. In Figure 1, there is a path from B to D. Though there exists a large angle switching at C in forward direction, C is not a turn, because both BC and CD are named as “Road R.”
2.2. The principle of the fewest-turn path algorithm

The proposed algorithm includes two key processes to search all fewest-turn paths from starting point \(S\) to endpoint \(T\): forward searching and backward cleaning. Forward searching starts from \(S\) to \(T\) to form a hierarchical graph \((HG)\) of zero-level reachable points. Backward cleaning starts searching in \(HG\) in the inverse direction (from \(T\) to \(S\)), deletes unnecessary points, and then extracts a fewest-turn-path graph \((G^*)\). For a given road network, the \(HG\) of zero-level reachable points is a graph to records relationship between the other vertexes and the starting point in a hierarchical form while the \(G^*\) is a product of \(HG\) after deleting all non-turn vertexes. The details of these key processes are described as follows.

**Process 1:** forward searching

There are three types of vertexes used in the process of forward searching: black, gray, and white. Black and gray vertexes denote those vertexes which have been searched. Among them, the black vertexes have existed in \(HG\) while the gray vertexes are waiting for being added to \(HG\). White vertexes denote that these vertexes have never been found. In addition, a FIFO (first in first out) queue \((Q)\) is used to manage all gray vertexes. And a hierarchical flag \((hf)\) is for marking the last vertex of one layer in \(Q\).

The pseudocode of forward searching is presented in algorithm 1 below. For a given road network \(G = (V, E, RNJun)\), forward searching includes three steps. Step 1: lines 1–4 initialize \(S\) to be gray and place it at the tail of \(Q\); assign \(S\) in \(Q\) to \(hf\), and then paint other vertexes white. Step 2: lines 6–9 remove the head of \(Q\) successively into \(HG\) and search its zero-level reachable points. If zero-level reachable point is white, then paint it gray, and place it at the tail of \(Q\). When \(hf\) detaches from \(Q\) and steps into \(HG\) in lines 10–11, a new layer in \(HG\) will be created and the last element of \(Q\) will be assigned to \(hf\) because all vertexes of an old layer have stepped into \(HG\) and all gray vertexes in \(Q\) are zero-level reachable points of last layer in \(HG\). Step 3: return to Step 2 if \(Q\) is not null and \(HG\) does not contain \(T\), or return \(HG\) in line 12. After Process 1, \(HG\) includes all possible fewest-turn paths. Its first layer contains only \(S\) while its last layer contains \(T\). The number of turns is equal to the number of middle layers. This result will be used as an input in Process 2.

**Algorithm 1: To build a hierarchical graph of zero-level reachable points.**

**Initial conditions:** \(G = (V, E, RNJun)\) is a connected, simple graph. \(SEV\) is the starting point. \(TEV\) is the endpoint.

**Input:** Road network \(G = (V, E, RNJun)\), \(S\), and \(T\).

**Output:** Hierarchical graph

**Initialization**

1. \(HG = null\).
2. All vertexes are set to be white except \(S\) is gray.
3. \(Q = \{S\}\).
4. \(hf = S\).
5. While \((HG != null\) and \(HG\) does not contain \(T\)) Do
   6. Current vertex = Retrieves and removes the head of \(Q\).
   7. For each vertex \(v\) in zero-level reachable points of Current vertex
      8. Set \(v\) to be gray and add \(v\) into \(Q\) if \(v\) is white.
      9. Set Current vertex to be black and add it into \(HG\).
   10. If \((Q\) does not contain \(hf))\) Then
      11. Set \(hf\) = the last element of \(Q\).
   12. Return \(HG\).

Figure 1. Non-turn illustration.

**Concept 2:** zero-level reachable point. \(A\) is a zero-level reachable point of \(B\) if there exists a path without any turn from \(B\) to \(A\). In Figure 2, for example, \(B, C,\) and \(D\) are zero-level reachable points of \(A\), but \(E\) is not a zero-level reachable point because there exists one turn at least on the path from \(A\) to \(E\).

**Concept 3:** road intersection set is a set which contains all intersections on a specified road, denoted as \(RNJun(road)\). In Figure 2, for example, \(RNJun(R5) = \{A, B, C, D\}\). Obviously, each point in \(RNJun\) is a zero-level reachable point for the others.

**Concept 4:** road network intersection set is an Union of all road intersections in a specified road network, denoted as \(RNJun\). For example, \(RNJun(G) = RNJun(R1)∪RNJun(R2)∪RNJun(R8)\), where \(G\) is the street network in Figure 2. Being different from general graph expression \(G = (V, E)\) in graph theory, an extended expression of graph is represented as \(G = (V, E, RNJun)\) in this study, where \(RNJun\) is the road network intersection set.

Figure 2. Manhattan Street Network \(G\).
**Process 2:** backward cleaning

To extract the $G^*$, backward cleaning deletes unnecessary vertexes in $HG$ which are independent of the fewest-turn-and-shortest path. This procedure is presented in algorithm 2 below. Suppose that $HG$ returned from forward searching have $N$ layers. And the first layer is marked as layer 1, the second as layer 2, and so on. There are two main steps included in the process of backward cleaning. Step 1: line 1 deletes all vertexes in the last layer except $T$. Step 2: the for loop of lines 2-4 searches zero-level reachable points of vertexes in layer $i + 1$ from layer $i$ and links two points together between these layers if they are zero-level reachable points for each other, while deletes vertexes in layer $i$ if they are not zero-level reachable points of vertexes in layer $i + 1$.

After backward cleaning, $HG$ has been transformed into the fewest-turn-path graph, denoted as $G^*$.

We can prove that each path in $G^*$ from $S$ to $T$ is a fewest-turn path and the number of turns is equal to $N - 2$. This conclusion will be proved in Section 3. After computing the length of each edge in $G^*$, the fewest-turn-and-shortest path will be obtained by executing any shortest path algorithm to $G^*$. This part will be discussed in Section 2.3.

An example to search the fewest-turn paths is given here. As shown in Figure 3(a), road network $G = (V, E, RNJun)$ is a simple and undirected graph, where $S$ is starting point and $T$ endpoint, $V = \{S, A, B, C, D, E, F, G, T\}$, $E = \{SA, SB, AC, BC, CD, CE, DF, DG, EF, FT, GT\}$, $RNJun(G) = RNJun(R1) \cup RNJun(R2) \cup \ldots RNJun(R7)$.

![Figure 3. The operation of forward searching on a simple and undirected graph.](image-url)
Algorithm 2. To extract the fewest-turn-path graph (G*).
Input: HG.
Output: The fewest-turn-path graph G*.
1. Initialization: delete all vertexes except T in layer N of HG.
2. For (layer i = n–1, i > 1, i–)
3. Delete vertexes in layer i if they are not zero-level reachable point of vertexes in layer i + 1.
4. Add an edge to link two zero-level reachable points between layer i and i + 1.

RJun(R1) = {G, T}, RJun(R2) = {D, F}, RJun(R3) = {A, C, E}, Jun(R4) = {S, B}, RJun(R5) = {E, F, T}, RJun (R6) = {B, C, D, G}, RJun(R7) = {S, A}.

An example of Process 1: forward searching
Figure 3(b–k) is the corresponding status description for displaying forward searching in detail. In the status description, Q is the FIFO queue to manage gray vertexes, hf is a hierarchical flag in Q, and HG is a hierarchical graph of zero-level reachable points. At the first step shown in Figure 3(b), paint S gray and place it at the tail of Q, assign S in Q to hf, and initialize HG to be null. Then in Figure 3(c), remove the head of Q (i.e. S) to the tail of HG and paint S black, paint A and B gray and place it at the tail of Q, and assign B in Q to hf. Since A and B are zero-level points of S which was just removed from Q, they are found and painted gray. Here, there is no order between A and B. In addition, the removal of hf in Q (i.e. S) results in the alteration of hf which was assigned to the last element of Q. Figure 3(d–k) works as the same way. In Figure 3(k), HG contains endpoint (T) and the forward searching terminates. Here, comma (,) in HG is a tag to divide layers.

An example of Process 2: backward cleaning
An example of backward cleaning is shown in Figure 4. Figure 4(a) is the HG returned from forward searching, here HG = {S, AB, CEDG, and FT}. The first layer contains only S while the last layer includes T. The procedure of backward cleaning works as Figure 4(b–e). First, starting cleaning from the last layer and deleting vertexes except T, the last layer contains only T, as shown in Figure 4(b). Second, E and G are linked to T, respectively, while C and D can be removed from layer 3, since E and G are zero-level reachable points of the last layer but C and D are not. As a result in Figure 4(c), there are E and G left in layer 3. Third, similar to layer 3, A is linked to E and B to G in layer 2, as shown in Figure 4(d). Finally, S is attached to every vertex in layer 2 and the G* is obtained as Figure 4(e).

2.3. Computing the fewest-turn-and-shortest path
The G* obtained from Section 2.2 has three characteristics. (1) It is a simple, directed, unweighted, and connected graph. (2) It contains all fewest-turn-paths. (3) Each path from S to T is a fewest-turn path. Obviously, the length of each edge in G* can be computed easily. Thus, G* will be a weighted graph if taking sides as weight, as shown in Figure 5. It is easy to obtain the shortest path from all fewest-turn paths in graph G* by using any shortest path algorithm. The classic Dijkstra algorithm is used here to compute the fewest-turn-and-shortest path.
Table 1. The comparison of path length and turns’ number between FTSP and SP provided by Baidu Map.

| Group ID | Starting point $S$ | Endpoint $T$ | Shortest path in Baidu map | Fewest-turn-and-shortest path |
|----------|-------------------|-------------|----------------------------|------------------------------|
|          | $X$(m) $Y$(m)     | $X$(m) $Y$(m) | Length (km) | Number of turns | Length (km) | Number of turns |
| 1        | 13493372.19 3637199.34 | 13495830.38 3637770.40 | 3.3 | 4 | 3.4 | 2 |
| 2        | 13491783.65 3634726.60 | 13490340.91 3638061.17 | 3.6 | 1 | 3.6 | 1 |
| 3        | 13493372.19 3637199.34 | 13496777.19 3635669.84 | 4.2 | 4 | 4.7 | 2 |
| 4        | 13491783.65 3634726.60 | 13492653.9 3638658.28 | 4.4 | 4 | 5.4 | 2 |
| 5        | 13491703.32 3635128.17 | 13494314.03 3637223.05 | 4.4 | 4 | 4.3 | 2 |
| 6        | 13493372.19 3637199.34 | 13491047.07 3634359.34 | 4.5 | 2 | 4.6 | 1 |
| 7        | 13493372.19 3637199.34 | 13492435.83 3633434.81 | 4.7 | 3 | 4.7 | 1 |
| 8        | 13493372.19 3637199.34 | 13497622.80 3634017.35 | 5.6 | 4 | 5.7 | 3 |
| 9        | 13489558.89 3639844.46 | 13489865.67 3634736.92 | 5.6 | 3 | 6.0 | 2 |
| 10       | 13491703.32 3635128.17 | 13492505.91 3640798.84 | 6.3 | 5 | 7.2 | 2 |
| 11       | 13496252.22 3635700.67 | 13492646.81 3639435.19 | 6.4 | 5 | 6.8 | 3 |
| 12       | 13489558.89 3639844.46 | 13495026.58 3637840.25 | 6.4 | 3 | 7.8 | 2 |
| 13       | 13496252.22 3635700.67 | 13490747.7 3636736.95 | 6.5 | 4 | 6.4 | 2 |
| 14       | 13489558.89 3639844.46 | 13494312.97 3636754.79 | 6.6 | 3 | 9.3 | 2 |
| 15       | 13491703.32 3635128.17 | 13493829.12 3640784.18 | 7.0 | 3 | 8.4 | 2 |
| 16       | 13489558.89 3639844.46 | 13493729.58 3635212.96 | 7.1 | 5 | 7.0 | 2 |
| 17       | 13496252.22 3635700.67 | 13492505.75 3640788.11 | 7.8 | 5 | 7.8 | 2 |
| 18       | 13496252.22 3635700.67 | 13493258.78 3638787.62 | 7.5 | 5 | 7.6 | 4 |
| 19       | 13496252.22 3635700.67 | 13490235.23 3638574.80 | 7.7 | 5 | 8.3 | 2 |
| 20       | 13496252.22 3635700.67 | 13489979.99 3633235.96 | 7.2 | 5 | 7.6 | 2 |
| 21       | 13496252.22 3635700.67 | 13489024.02 3637370.74 | 8.1 | 6 | 8.4 | 3 |
| 22       | 13496252.22 3635700.67 | 13490020.71 3639949.93 | 8.9 | 5 | 10.1 | 2 |

Note: The coordinate system of $S$ and $T$ is WGS 1984 Web Mercator Auxiliary Sphere.
3. Correctness proof of algorithm

In the process of backward cleaning, a conclusion, each path from starting point (S) to endpoint (T) in G* is a fewest-turn path and the number of turns is equal to N−2, is presented. To prove this conclusion, we must first prove that, in the HG of zero-level reachable points, the number of turns in a fewest-turn path from S to v equals to i−2. For Vv∈layer i (i ≥ 2) and all its corners exist in HG. Since vertex set in G* is a subset of HG, the fewest-turn path algorithm is true if this theorem can be proved. Before presenting the proof of this theorem, there are two lemmas to be proved.

Lemma 1

In the HG of zero-level reachable points, suppose that v1∈layer N1, v2∈layer N2. If v1 and v2 are zero-level reachable points for each other, then |N1−N2| ≤ 1.

Proof by contradiction Suppose that |N1−N2| ≥ 2. Let N1 ≤ N2, then v2 is being searched after v1 because v2 is zero-level reachable point of v1. Thus, v2∉layer (N1 + 1) necessarily and N2 = N1 + 1. Obviously, N2−N1 = 1 and |N1−N2| ≥ 2 are contradictory. Therefore, |N1−N2| ≤ 1.

This lemma reveals that two zero-level reachable points are in the same layer, or in two adjacent layers respectively.

Lemma 2

Suppose that the HG of zero-level reachable points have N layers. If ∀v∈layer i (i > 2), then 3v∉layer (i−2) to set up the number of turns in fewest-turn path from v1 to v is equal to 1 and its corner exists in layer (i−1).

Proof According to Lemma 1, it is impossible that v1 and v are zero-level reachable point for each other. Thus, there exists a turn, at least, from v1 to v. As all vertexes in layer i are zero-level reachable point of vertexes in layer (i−1), 3v∉layer (i−1) to set up v2 → v is the fewest-turn path from v2 to v. Similarly, there exists a vertex in layer (i−2) as a zero-level reachable point of v2. Suppose it is v1, then 3v∉layer (i−1) to set up v1 → v2 is the fewest-turn path from v1 to v2. Therefore, exist a path v1 → v2 → v from v1 to v. Thus, 3v∉layer (i−2) to set up the number of turns in fewest-turn path from v1 to v equal to 1 and its corner v2∉layer (i−1).

Theorem (correctness of forward searching)

Let G=(V, E, RN Jun) be a simple and undirected graph. Suppose that S is starting point and T endpoint. Applying forward searching on G, generate a HG of zero-level reachable points with N layers. In HG, if v∉layer i (i ≥ 2), then, number of turns in a fewest-turn path from S to v equal to i−2 and this fewest-turn path exists in HG.

Proof For i = 2, according to the principle of forward searching, all vertexes in layer 2 are zero-level reachable point of S. Therefore, S → v is the fewest-turn path from S to v and the number of turns is equal to 0. Conclusion establishes.

For i > 2, by Lemma 2 shows that 3v∉layer (i−2), 3v∉layer (i−1) to set up the fewest-turn path from v−1 to v−1 is v−2 → v−1 → v and the number of turns is equal to 1.

Use Lemma 2 repeatedly, a fewest-turn path from S to v is S → v2 → … → v−3 → v−2 → v−1 → v where v2, …, v−3, v−2, and v−1 locates in layer 2, …, i−3, i−2, and i−1 in sequence.

4. Experiments and discussion

To verify the effectiveness of the presented algorithm, an experiment is designed to compare the routing result of the proposed algorithm with the shortest path suggested by Baidu Map. The Dijkstra shortest path algorithm is used to compute the shortest path among all fewest-turn paths.

A data-set of road network used here is located in downtown of Songjiang Zone, Shanghai, China, which can be downloaded from Open Map Street (www.openstreetmap.org). The experiment chose 22 groups of starting points and endpoints, and recorded corresponding length and the number of turns in the fewest-turn-and-shortest path and the shortest path of Baidu Map, respectively. These results are listed in Table 1. We selected three groups (ID 20–22) and exhibited their path results in Figure 6.

Figure 6 shows the differences of routing selection between SP and FTSP, where SP is the abbreviated for the shortest path provided by Baidu Map and FTSP for the fewest-turn-and-shortest path. For example in group 20, the FTSP contains only two turns but the SP contains twice more than the former. The length of FTSP is...
Figure 7. The comparison of the length and number of turns between SP and FTSP.

6% longer than the SP. In Figure 6, S is the starting point of group 20–22, while E_{20}, E_{21}, and E_{22} are the endpoints of group 20–22, successively. And a cross “×” denotes a turn of a path.

Figure 7 illustrates the comparison of the length and number of turns between SP and FTSP. The value of twill cylinder is the ratio of length of SP to FTSP, while the value of gray cylinder is the ratio of number of turns in SP to FTSP. As revealed in Figure 7, the ratio of length is approximately equal to 1 which means their path length is almost equal. The ratio of number of turns is not significantly equal to 1 and mainly distributes from 1.5 to 2.5, which means that the number of turns in the former is times than that of the latter.

5. Conclusions

In this paper, we developed an algorithm to plan the fewest-turn-and-shortest path, which can achieve a balance between distance travelled and simplicity of path. A simpler path was obtained with its length nearly as short as that of the shortest path. The results indicate that the fewest-turn-and-shortest path is superior to the shortest path from the consideration of the length and turns factors. The superiority could be even more obvious when using the fewest-turn-and-shortest path algorithm into more complicated road network structures, and there is still possibility to improve the performance of our proposed algorithm, which will be studied in our future work.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Nos. 41471332 and 41101354), the National High Technology Research and Development Program of China (863 Program) (No. 2013AA12A302), the Fundamental Research Funds for the Central Universities (No. ZYGX2011J077), and the Fund of China Scholarship Council.

Notes on contributors

Yan Zhou is an associate professor at School of Resources and Environment, University of Electronic Science and Technology of China. Her research interests include spatial analysis, high-performance spatial data management, and 3D GIS technology and application.

Weisheng Wang received his BS degree in Electronic and Information Engineering from Shanghai University of Engineering Science, in 2013. He is currently a master student of School of Resources and Environment, University of Electronic Science and Technology of China. His research is focused on spatial analysis.

Di He received his BS degree in Electronic and Information Engineering from University of Electronic Science and Technology of China, in 2014. He is currently a master student of School of Resources and Environment, University of Electronic Science and Technology of China. His research interest is 3D GIS technology and application.

Zhe Wang received her BS degree in Communication Engineering from Qingdao University of Science and Technology, in 2014. She is currently a master student of School of Resources and Environment, University of Electronic Science and Technology of China. Her research interest is high-performance spatial data management.

References

(1) Golledge, R. Path Selection and Route Preference in Human Navigation: A Progress Report. In Spatial Information Theory A Theoretical Basis for GIS: Frank, A., Kuhn, W., Eds.; Springer Berlin Heidelberg: Semmering, 1995; pp 207–222.
(2) Golledge, R.G.; Garling, T. Cognitive Maps and Urban Travel. In Handbook of Transport Geography and Spatial Systems: Hensher, D.A., Button, K.J., Haynes, K.E., Stopher, P.R., Eds.; Elsevier: Amsterdam, 2004; pp 501–512.
(3) Streeter, L.A.; Vitello, D.; Wonsiewicz, S.A. How to Tell People where to Go: Comparing Navigational Aids. Int. J. Mar. Mach. Stud. 1985, 22 (5), 549–562.
(4) Streeter, L.A.; Vitello, D. A profile of drivers’ map-reading abilities. Hum. Factors 1986, 28 (2), 223–239.
(5) Duckham, M.; Kulik, L. “Simplest” paths: automated route selection for navigation. In Spatial Information Theory. Foundations of Geographic Information Science: Kuhn, W., Worboys, M., Timpl, S., Eds.; Springer Berlin Heidelberg: Kartause Ittingen, 2003; pp 169–185.
(6) Denis, M.; Pazzaglia, F.; Cornoldi, C.; Bertolo, L. Spatial Discourse and Navigation: An Analysis of Route Directions in the City of Venice. Appl. Cognitive Psychol. 1999, 13 (2), 145–174.
(7) Turner, A. The Role of Angularity in Route Choice: An Analysis of Motorcycle Courier Gps Traces. In Spatial Information Theory: Hornsby, K., Claramunt, C., Denis, M., Ligozat, G., Eds.; Springer Berlin Heidelberg: Aber Wrac’h, 2009; pp 489–504.
(8) Shapiro, J.; Waxman, J.; Nir, D. Level Graphs and Approximate Shortest Path Algorithms. Networks 1992, 22 (7), 691–717.
(9) Liu, B. Using Knowledge to Isolate Search in Route Finding. Proceedings of the 14th International Joint Conference on Artificial Intelligence—Volume 1, Morgan Kaufmann: Montreal, Quebec, Canada, 1995; pp 119–124.
(10) Jiang, B.; Liu, X.T. Computing the Fewest-Turn Map Directions Based on the Connectivity of Natural Roads. Int. J. Geog. Inf. Sci. 2011, 25 (7), 1069–1082.