ROUTING WITH MINIMUM NUMBER OF LANDMARKS

Jun Luo* **and Rong Peng†** **and Chenglin Fan‡** **and Jinxing Hu§**

Shenzhen Institutes of Advanced Technology
Chinese Academy of Sciences, China

KEY WORDS: Landmark, algorithms, routing

ABSTRACT:

Routing problem has been studied for decades. In this paper, we focus on one of the routing problems: finding a path from source to destination on road network with the guidance of landmarks. People use landmarks to identify previously visited places and reoriented themselves in the environment. When people give direction instructions for other people, they also like to refer to landmarks. In this sense, we want to find a path such that it visits as many landmarks as possible but also the distance of the path is as short as possible. However, in some situations, the wayfinder may not want to see as many landmarks as possible along the way. For example, the wayfinder drives a car from source to destination. He probably doesn't want to use many landmarks to guide his driving since it's not convenient to switch from one landmark to the other landmark frequently. But he still want to have at least one landmark to be seen at any point along the way. Therefore, the problem becomes: find a path P from s to t such that the driver can see at least one landmark at any point along P and the number of landmarks the driver can stick to is minimized. There are two cases: (a) The same landmark in different road segments counts twice. (b) The same landmark in different road segments counts once. We give the optimal solutions for those two problems by using modified Dijkstra's shortest path algorithm and modified Bellman-Ford algorithm.

1 INTRODUCTION

Routing problem has been studied for decades. The first routing problem could be traced back to 1959 when Dantzig and Ramser proposed vehicle routing problem (Dantzig and Ramser, 1959). Routing problem is very important not only in the field of transportation, but also in the field of logistics, distribution, TCP/IP networks, wireless sensor networks and so on (Golden et al., 2008, Campbell et al., 1997, Campbell et al., 2002, Huitema, 1995, Al-Karaki and Kamal, 2004). The classic routing problem is the shortest path problem that is given the road network and finding the shortest path from source to destination.

In this paper, we focus on one of the routing problems: finding a path from source to destination on road network (Car and Frank, 1993, Gaisbauer and Frank, 2008). A good wayfinding system provides precise and enough indicators of where the wayfinder current location is and how to get to the destination from his/her current location. One of the important indicator for human wayfinding is landmark (Jacob et al., 1999, Lovelace et al., 1999, Peebles et al., 2007, Raubal and Winter, 2002, Elias, 2003, Michon and Denis, 2001).

Landmarks are defined as entities that are salient and easily distinguishable from their surrounding background. People use landmarks to identify previously visited places and reoriented themselves in the environment. When people give direction instructions for other people, they also like to refer to landmarks. In this sense, we want to find a path such that it visits as many landmarks as possible but also the distance of the path is as short as possible. However, in some situations, the wayfinder may not want to see as many landmarks as possible along the way. For example, the wayfinder drives a car from source to destination. He probably doesn't want to use many landmarks to guide his driving since it's not convenient to switch from one landmark to the other landmark frequently. But he still want to have at least one landmark to be seen at any point along the way.

Suppose we already know the road network N with n nodes, m land marks $M_i(i = 1, ..., m)$, the road segments that could be seen by M_i are painted by color $C_i(i = 1, ..., m)$ (see Figure 1). Actually, the road network should be directed since the same landmark could be seen on one direction at one place but couldn't be seen

*jun.luo@sub.siat.ac.cn
†rong.peng@siat.ac.cn
‡cl.fan@sub.siat.ac.cn
§jinxing.hu@sub.siat.ac.cn

on the other direction at the same place. However, this will not change the complexity of our algorithms. Therefore we assume the road network is not directed in the remaining part of this paper. Suppose the road network can be segmented. Each segment has unit length. Each color covers either the whole segment or none of the segment. Then we can count how many colors cover each road segment.

Given source and destination s, t in N, the problem becomes:

Problem 1 $\mathbf{MinLandmarks}$. *Find a path P from s to t such that the driver can see at least one landmark (or can be associated with at least one color) at any point along P and the number of landmarks (or colors) the driver can stick to is minimized. There are two cases: (a) The same landmark in different road segments counts twice. (b) The same landmark in different road segments counts once.*

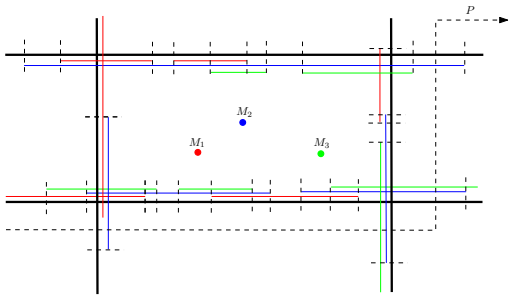

Figure 1: Illustrating the two cases for problem 1. For the first case, the number of times the driver sees the landmark M_1 (red color) along path P is 3 while in the second case, that is 1.

Shortest path problem can be divided into two categories: single-objective shortest path problem (SSPP) and multi-objective shortest path problem (MSPP). The objective of SSPP is only one: minimize the distance from source to destination. Although the objective of $\mathbf{MinLandmarks}$ problem is not to minimize the distance from source to destination, it can be converted to single-objective shortest path problem. There are abundant algorithms for single-objective shortest path problem from the classic Dijkstra's algorithm to the latest evolutionary algorithm (Cormen et al., 2001, S. Baswana and Neumann, 2009).

2 MINIMIZE THE NUMBER OF LANDMARKS

In this section, we solve the $\mathbf{MinLandmarks}$ problem defined in section 1. Again, we use the color scheme introduced in section 1 such that each landmark is assigned a color and the road segments that are visible to that landmark are also painted by the color of that visible landmark. Furthermore, we insert *virtual nodes* on two endpoints of each colored road segments. We call the original nodes of the road network N are *real nodes* (see figure 2). We use *nodes* to refer both *virtual nodes* and *real nodes*. If the virtual node is coincide with real node, then the virtual node is deleted. There are two observations of this new graph.

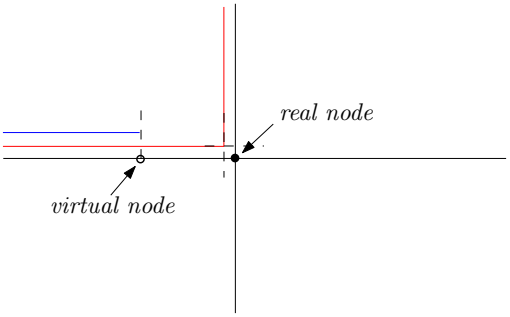

Figure 2: Illustration of *virtual nodes* and *real nodes*.

Observation 1 *The colors cover the edge between* nodes *do not change.*

Observation 2 *Without loss of generality, we assume the endpoints of the different color road segments are not in the same places. Therefore, for a virtual node, there are only two edges adjacent to it and the number of colors covering those two edges is only different by one.*

Since one landmark could be seen by different disconnected road segments, the different road segments with same color could be seen more than once along a certain path. Depending on how we count the same color road segments, it leads to two different objective functions thus leads to two different algorithms.

2.1 The same landmark in different road segments counts twice

In this section, we discuss the scenario that along some path if we see the same landmark on two disconnected road segments, the same landmark is treated as two different landmarks. In other words, we count the landmark or the color twice.

Our algorithm is similar to Dijkstra's single source shortest path problem. The input is the graph $G = (V, E)$,

Algorithm MIN-LANDMARK-TWICE(G, w, s)
1. INITIALIZE-SINGLE-SOURCE(G, s)
2. $S \leftarrow \emptyset$
3. $Q \leftarrow V[G]$
4. **while** $Q \neq \emptyset$
5. **do** $u \leftarrow$ EXTRACT-MIN(Q)
6. $S \leftarrow S \cup \{u\}$
7. **for** each vertex $v \in Adj[u]$
8. **do** RELAX(u, v, w)

Algorithm INITIALIZE-SINGLE-SOURCE(G, s)
1. $C1[s], C2[s] \leftarrow \emptyset$
2. $counter[s] \leftarrow 0$
3. **for** each node $u \in V \backslash s$
4. **do** $C1[u], C2[u] \leftarrow \emptyset$
5. $counter[u] \leftarrow \infty$

Algorithm RELAX(u, v, w)
1. $C1' \leftarrow C1[u] \cap w$
2. $C2' \leftarrow w - C1'$
3. **if** $w = \emptyset$
4. **then** exit
5. **else if** $C1' = \emptyset$
6. **then** $counter' \leftarrow counter[u] + 1$
7. $C1' \leftarrow C2'$
8. $C2' \leftarrow \emptyset$
9. **if** $counter' < counter[v]$
10. **then** $counter[v] \leftarrow counter'$
11. $C1[v] \leftarrow C1'$
12. $C2[v] \leftarrow C2'$
13. **if** $counter' = counter[v]$
14. **then** $C1[v] \leftarrow C1' \cup C1[v]$
15. $C2[v] \leftarrow C2' \cup C2[v] - C1[v]$

where G is the road network with virtual nodes, V are the nodes and E are the edges between nodes. However, the weight w of each edge $(u, v) \in E$ is not the distance between u, v. The weight w is actually the list of colors that cover edge (u, v). For each node u, we maintain two sets $C1, C2$ and one integer variable $counter$. $C1, C2$ store the colors that are visible just before node u. $counter$ records the minimum number of colors needed to cover the optimal path from s to u so far.

Actually, $C1[v]$ denotes the latest possible colors people stick to on the optimal path from source to node v, the colors in $C1[v] \cup C2[v]$ are the colors covering the edge (u, v). We should stick to one of the color in $C1[v]$ for the edge (u, v). Choosing which one to stick depends on which color lasts the longest but we can not decide that up to node v unless there is only one color left in $C1[v]$. The colors in $C2[v]$ are the colors might be used as guidance color later and will be moved to $C1[v]$ only after $C1[v]$ is empty. Of course, some road segments may not be covered by any colors. In that case, we may not find a path satisfying our requirement. We consider this case in our algorithm.

The algorithm MIN-LANDMARK-TWICE looks exactly the same as Dijkstra's algorithm. However, the three subroutines are different. The first two are straightforward. We explain the third subroutine RELAX(u, v, w) in detail.

In line 1, the colors appear in both $C1[u]$ and w are assigned to the temporary color list $C1'$ because we want to stick to the colors in $C1[u]$ as long as possible for traveling edge (u, v). In line 2, the colors appear in w but not in $C1[u]$ are put into another temporary color list $C2'$ because currently we don't need to stick to those colors but we probably need them later. Line 3 to 6 deal with the situation if $C1'$ is empty. That means all colors in $C1[u]$ disappear in w. We have to switch to the colors in w and the counter needs to increase by one. Line 7 to 10 just substitute $counter[v], C1[v], C2[v]$ with the new values $counter', C1', C2'$ if the new counter is smaller than the old one. If the new counter is the same as the old one, line 11 to 13 just merge $C1[v], C2[v]$ with the new values $C1', C2'$ and take off the colors appearing in updated $C1[v]$ from updated $C2[v]$. The correctness of line 7 to 13 is given as follows:

Algorithm EXTRACT-MIN(Q)
1. output the node $u \in Q$ such that $counter[u]$ is the minimum

Lemma 1 *For a path passing through nodes v_1, v_2, v_3 consecutively, $counter[v_3]$ is larger than $counter[v_2]$ only by one at most, that means $counter[v_2] \leq counter[v_3] \leq counter[v_2] + 1$.*

Proof. Suppose the color lists of edge (v_1, v_2) and (v_2, v_3) are w_1 and w_2, if w_1 and w_2 are the same, which could happen when v_2 is a real node, then the color we stick to on w_1 can be still used on (v_2, v_3), thus $counter[v_3] = counter[v_2]$. If w_1 and w_2 are different, then if the color we stick to in w_1 is still appear in w_2, thus we can continue to use that color, that means $counter[v_3] = counter[v_2]$. Otherwise we just stick to a new color in w_2, that means $counter[v_3] = counter[v_2] + 1$. Thus the lemma follows. ∎

Lemma 2 *For a node v, we only need to keep the smallest $counter[v]$ and corresponding $C1$ and $C2$.*

Proof. If v is a virtual node, from observation 2, we know there are only one incoming edge of v, then there is only one value of $counter[v]$.

If v is a real node, there could be multiple incoming edges. Without loss of generality, we assume there are two incoming edges, (u_1, v) and (u_2, v) and one outgoing edge (v, x). Let the color lists for (u_1, v), (u_2, v),(v, x) be w_1, w_2, w_3 respectively and $counter[v]_1, counter[v]_2$, $counter[x]_1, counter[x]_2$ be counter numbers from u_1, u_2 to v and then to x respectively. From lemma 1, we know $counter[v]_1 \leq counter[x]_1 \leq counter[v]_1 + 1$ and $counter[v]_2 \leq counter[x]_2 \leq counter[v]_2 + 1$. There are three cases:

- $counter[v]_1$ and $counter[v]_2$ are different by one such that $counter[v]_2 = counter[v]_1+1$(if $counter[v]_1 = counter[v]_2 + 1$, the proof is symmetric). In worst case, $counter[x]_1 = counter[x]_2$ that means the colors we could stick to on edge (u_1, v) (which are actually the colors in $C1[v]$) are disappear and we have to switch to the colors on edge (v, x). According to the algorithm RELAX, $C1[x]_1 = w_3 \supseteq C1[x]_2$. Therefore, $C1$ will be empty earlier if the path goes through u_2. Because the color counter increases only when $C1$ is empty, the path going through u_1 is better than the path going through u_2.
- $counter[v]_1$ and $counter[v]_2$ are different more than one. Suppose $counter[v]_2 \geq counter[v]_1+2$. According to the proof of above case, the counter for the path going through u_1 will never be larger than the counter for the path going through u_2. Thus we only need keep smaller one.
- $counter[v]_1 = counter[v]_2$. According to the algorithm RELAX, $C1[v]_1$ and $C1[v]_2$ are merged into $C1[v]$. Suppose if we keep $C1[v]_1$ and $C1[v]_2$ separately and $C1[v]_1$ becomes empty first, then $C1[v]$ becomes empty when $C1[v]_2$ becomes empty. That means we implicitly follow the optimal path going through $C1[v]_2$.

∎

The proof of the correctness of our algorithm is the same as that of Dijkstra's algorithm except we use $counter[v]$ instead of $d[v]$. The running time of our algorithm is also similar to that of Dijkstra's algorithm except in relax step, the transactions of intersection and union of two lists $C1$ and $C2$ take extra $O(m)$ time where m is the number of landmarks. So the total running time is $O(|V|^2 + |E|m) = O(n^2m)$.

Theorem 1 *For the road network with n vertices and m landmarks, we can find an optimal path from s to t in $O(n^2m)$ time for the* **MinLandmarks** *problem if the same landmark counts twice when it is seen twice at two disconnected road segments.*

2.2 The same landmark in different road segments counts once

For this problem we use different data structures. Original input is the road network graph $G = (V, E)$ with each edge covered by different color road segments. We augment G to $G' = (V, E')$ as follows: for each edge $(u, v) \in E$, we compute all the combinations of colors that could cover the whole edge. Each combination is represented by an edge from u to v (see figure 3). Then each edge (u, v) in G could be augmented to many edges in G' and each edge in G' is associated with a color list $C[u, v]_i$ where $1 \leq i \leq k$ and k is the number of combinations of colors that could cover the whole edge (u, v). Let $|C[u, v]_i|$ denote the number of colors in the list and $(u, v)_i$ denote the ith augmented edge of (u, v).

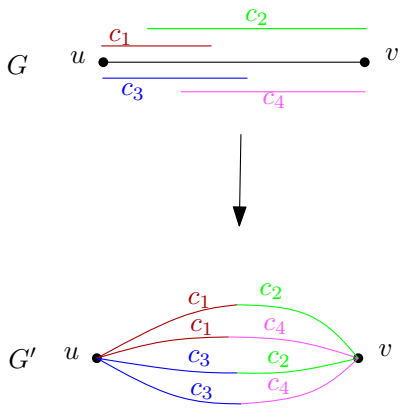

Figure 3: Augmentation from G to G'.

This problem is more complicated than counting twice case and we can not use Dijkstra's algorithm. This is because the optimal path from s to t passing through node v may not consist of the optimal path from s to v and the optimal path from v to t. For example, in figure 4, the optimal path from s to t consists of P_1 and P_3 which are

covered by three colors c_1, c_2, c_3. While the optimal path from s to v is P_2 which is covered by two colors c_4, c_5 and the optimal path from v to t is P_4 which is covered by two colors c_6, c_7. To solve this problem, it seems that we have to record all the color lists for all possible paths form s to v. Fortunately, we don't need to do that. We can use the Bellman-Ford algorithm with different data structure. Let $C_i[v]$ be the ith color list for node v that could cover some paths from s to v.

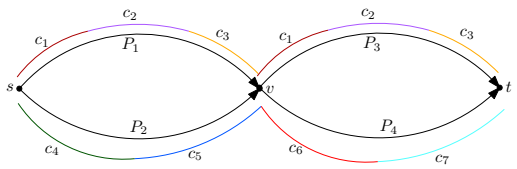

Figure 4: The example shows that the optimal path from s to t passing through node v may not consist of the optimal path from s to v and the optimal path from v to t.

Lemma 3 *If the optimal path from s to v is $p =< v_0, v_1, ..., v_k >$ where $v_0 = s$ and $v_k = v$, the color list corresponding p is $C_{opt}[v]$, then after kth passes over the edges of G' in* MIN-LANDMARK-ONCE, *$C_{opt}[v]$ is one of the color lists of v.*

Proof. Actually, $C_{opt}[v] = C[v_0, v_1]_{i_1} \cup C[v_1, v_2]_{i_2} \cup ... \cup C[v_{k-1}, v_k]_{i_k}$ where $C[v_{j-1}, v_j]_{i_j}$ is the color list for the i_jth augmented edge of edge (v_{j-1}, v_j). After first pass over the edges of G', we can get the list $C[v_0, v_1]_{i_1}$ for v_1 and after second pass over the edges of G', we can get the list $C[v_0, v_1]_{i_1} \cup C[v_1, v_2]_{i_2}$ for v_2, and so on. We can get $C[v_0, v_1]_{i_1} \cup C[v_1, v_2]_{i_2} \cup ... \cup C[v_{k-1}, v_k]_{i_k}$ for v_k. Thus the lemma follows. ∎

After the algorithm MIN-LANDMARK-ONCE, we can get color lists $C_1[v], C_2[v], ..., C_k[v]$ for each node v. According to lemma 3, we know that $C_i[v]$ is $C_{opt}[v]$ if the number of colors of $C_i[v]$ is the smallest among all color lists of v. However, we can not report the optimal path from s to v since we does not provide any backtrack scheme in the algorithm and data structure. Actually, we only need to add a pointer for each color list $C_i[v]$ of

Algorithm MIN-LANDMARK-ONCE (G',$C[u, v]$,s)
1. INITIALIZE-SINGLE-SOURCE(G', s)
2. **for** $i \leftarrow 1$ to $|V(G')| - 1$
3. **do for** each edge $(u, v) \in E(G')$
4. **do** RELAX($u, v, C[u, v]$)

Algorithm INITIALIZE-SINGLE-SOURCE(G, s)
1. **for** each node $u \in V$
2. **do** $C[u] \leftarrow \emptyset$

Algorithm RELAX($u, v, C[u, v]$)
1. **for** each color list $C_[u]$ of u
2. **do** $C[v] = C[u] \cup C[u, v]$

v. This pointer points to the color list $C_j[v']$ that generates $C_i[v]$. To report the optimal path from s to t, we first find $C_{opt}[t]$ and get the pointer for $C_{opt}[t]$. Suppose the pointer points to $C_i[v]$, then the predecessor of t is v and we backtrack the path from $C_i[v]$ recursively until we reach $C[s]$.

The running time of this algorithms is $O(|V(G)| \cdot |E(G')| \cdot l \cdot m) = O(n^3 lm)$ since each edge relaxation needs $O(l)$ times of two color lists union operations and each union takes $O(m)$ time where l is the maximum number of color lists for one node. The running space is $O(|V(G)| \cdot l \cdot m) = O(nlm)$ since each color list needs $O(m)$ space.

Theorem 2 *For the road network with n vertices and m landmarks, we can find an optimal path from s to t in $O(n^3 lm)$ time and $O(nlm)$ space for the* **MinLandmarks** *problem if the same landmark counts once when it is seen twice at two disconnected road segments, where l is the maximum number of color lists for one node.*

3 DISCUSSION

In this paper we presented $O(n^2 m)$ time algorithm for the **MinLandmarks** problem when the same landmark counts twice and $O(n^3 lm)$ time and $O(nlm)$ space algorithm when the same landmark counts once where n is the number of vertices of road network and m is the number of landmarks and l is the maximum number of color lists for one node. For the latter case, we know that the running time and space of the optimal algorithm are all related to l. In worst case, $l = O(C_1^m + C_2^m + ... + C_m^m) = O(m^m)$. If m is constant, that optimal algorithm is a polynomial time and space algorithm. Otherwise, that is exponential time and space algorithm which is unacceptable. Thus it's worth investigating whether there exists an approximation algorithm with polynomial running time and space of n, m.

The other interesting open problem is to finding a path such that it visits as many landmarks as possible but also the distance of the path is as short as possible. This problem is much harder than **MinLandmarks** problem and

actually is a multiobjective shortest path problem which has been proved to be NP-complete. Therefore proposing an approximation algorithm for this problem is also a challenge.

REFERENCES

Al-Karaki, J. N. and Kamal, A. E., 2004. Routing techniques in wireless sensor networks: a survey. IEEE Wireless Communications 11(6), pp. 6–28.

Campbell, A., Clarke, L. and Savelsbergh, M., 2002. Inventory routing in practice. In: D. V. P. Toth (ed.), The Vehicle Routing Problem, SIAM monographs on discrete mathematics and applications, pp. 309–330.

Campbell, A., Clarke, L., Kleywegt, A. and Savelsbergh, M., 1997. The inventory routing problem. In: T. Crainic and G. Laporte (eds), Fleet Management and Logistics, Kluwer Academic Publishers, pp. 95–113.

Car, A. and Frank, A. U., 1993. Hierarchical street networks as a conceptual model for efficient way finding. In: Proceedings of the EGIS'93, pp. 134–13913.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C., 2001. Introduction to Algorithms, Second Edition. The MIT Press.

Dantzig, G. and Ramser, J., 1959. The truck dispatching problem. Management Science 6(1), pp. 80–91.

Elias, B., 2003. Extracting landmarks with data mining methods. In: COSIT 2003: Proceedings of the International Conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, pp. 375–389.

Gaisbauer, C. and Frank, A. U., 2008. Wayfinding model for pedestrian navigation. In: Proceedings of the 11th AGILE International Conference on Geographic Information Science 2008, pp. 59–68.

Golden, B., Raghavan, S. and Wasil, E., 2008. The Vehicle Routing Problem: Latest Advances and New Challenges. Springer.

Huitema, C., 1995. Routing in the Internet. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Jacob, R., Marathe, M. and Nagel, K., 1999. A computational study of routing algorithms for realistic transportation networks. J. Exp. Algorithmics 4, pp. 6.

Lovelace, K. L., Hegarty, M. and Montello, D. R., 1999. Elements of good route directions in familiar and unfamiliar environments. In: COSIT '99: Proceedings of the International Conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, Springer-Verlag, London, UK, pp. 65–82.

Michon, P.-E. and Denis, M., 2001. When and why are visual landmarks used in giving directions? In: COSIT 2001: Proceedings of the International Conference on Spatial Information Theory, Springer-Verlag, London, UK, pp. 292–305.

Peebles, D., Davies, C. and Mora, R., 2007. Effects of geometry, landmarks and orientation strategies in the drop-off orientation task. In: COSIT 2007: Proceedings of the International Conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, pp. 390–405.

Raubal, M. and Winter, S., 2002. Enriching wayfinding instructions with local landmarks. In: GIScience '02: Proceedings of the Second International Conference on Geographic Information Science, Springer-Verlag, London, UK, pp. 243–259.

S. Baswana, S. Biswas, D. D. T. F. P. K. and Neumann, F., 2009. Computing single source shortest paths using single-objective fitness functions. In: Proceedings of ACM International Workshop on Foundations of Genetic Algorithms (FOGA 2009).