Erschienen in: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. - New York, NY : ACM Press, 2015. - 68. - ISBN 978-1-4503-3967-4
http://dx.doi.org/10.1145/2820783.2820858

Alternative Routing: k-Shortest Paths with Limited Overlap

Theodoros Chondrogiannis [#] Panagiotis Bouros [†] Johann Gamper [#] Ulf Leser [†]

[#]Faculty of Computer Science
Free University of Bozen-Bolzano, Italy
{tchond,gamper}@inf.unibz.it

[†]Department of Computer Science
Hunboldt-Universität zu Berlin, Germany
{bourospa,leser}@informatik.hu-berlin.de

ABSTRACT

Shortest path computation is a fundamental problem in road networks with application in various domains in research and industry. However, returning only the shortest path is often not satisfying; users are also interested in alternative paths which might be longer but have other advantages, e.g., less frequent traffic congestion. In this paper, we formally introduce the *k-Shortest Paths with Limited Overlap* (k-SPwLO) problem seeking to recommend k alternative paths which are (a) as short as possible and (b) sufficiently dissimilar based on a user-controlled similarity threshold. We propose two algorithms that examine the paths from a source s to a target t in increasing order of their length and progressively construct the result set. The baseline algorithm BSL builds upon a standard algorithm for computing k-Shortest Paths, followed by a filter step. The OnePass algorithm considers the overlap constraint in each expansion step while traversing the network. We evaluate the performance of both algorithms on real road networks and show that OnePass always outperforms BSL.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

Keywords

shortest path, similarity, alternative routing

1. INTRODUCTION

Computing the shortest path between two locations of a road network is a fundamental problem that has attracted a lot of attention both in research and in industry. State-of-the-art methods [1, 2] compute the shortest path in linear time, even for world-scale road networks. However, determining solely the overall shortest path is not sufficient in many real-world scenarios. Most commercial route planning and navigation systems offer alternatives longer than the shortest path but with other desirable properties, e.g., less traffic. Another important scenario arises in emergency situations such as natural disasters or terrorist attacks, where different evacuation plans need to be determined and communicated. In both examples, it is important that the alternatives are not only short, but also significantly different from each other.

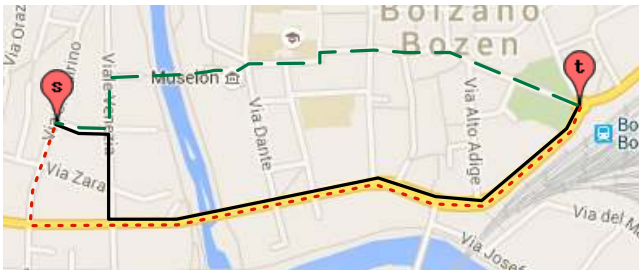


Figure 1: Motivational Example.

A direct approach for computing alternative paths is the k-Shortest Paths problem [3, 4]. However, in most cases the returned paths share a large number of edges and thus they are not valued as true alternatives by users. Consider Figure 1, which shows three distinct paths from location s to t in the city center of Bolzano. The solid/black line indicates the shortest path from s to t. The dotted/red line shows the next path by length which is very similar to the shortest path. Finally, the dashed/green line shows a path which is longer than the dotted/red path but significantly different from the shortest path as they traverse different parts of the city's road network. In applications like the ones discussed above, it is the dashed/green path that can be considered a good and useful alternative to the shortest path.

The problem of alternative routing has been studied before. However, previous works differ substantially in the underlying similarity definition and the algorithmic approach they take. Given a source node s and target node t, [5] incrementally computes a set of k dissimilar alternative paths. First, the shortest path from s to t is added to the result set. Then, an extension of Yen's algorithm [3] is adopted to progressively produce paths that obey a given similarity constraint (shared length). In contrast to this method, [6, 7, 8] first compute a large set of candidate paths and then filter candidates with respect to some similarity constraints (e.g., their length or the nodes they cross). Another approach is to repeatedly run a shortest path algorithm on a network that is progressively changed by adding penalties to edge weights. The penalties are derived from the overlap of the actual path with previously computed paths [9, 10]. Finally, the alternative routing problem has also been approached as skyline query on multi-criterion networks [11, 12, 13].

In contrast to our approach in this paper, the above studies do not provide any guarantees regarding the quality of the alternative

SIGSPATIAL '15 November 03-06, 2015, Bellevue, WA, USA

DOI: http://dx.doi.org/10.1145/2820783.2820858

paths, i.e., being as short as possible. In addition, most of the current methods do not compute alternative paths that are dissimilar to each other, rather only the similarity to the shortest path is considered. For any $k > 2$, this typically produces unsatisfactory results.

In this paper, we introduce a novel definition of alternative paths. Our focus is to recommend a set of k paths (including the shortest path p_0) such that every path in the result is (a) sufficiently dissimilar to all its predecessors (based on a user-specified similarity threshold) and (b) as short as possible. We formalize this form of alternative routing as the *k-Shortest Paths with Limited Overlap* (k-SPwLO) problem and propose two evaluation algorithms, termed BSL and OnePass. Both algorithms examine the paths in increasing order of their length from the source node and terminate as soon as k paths within the similarity and length constraints are identified. The baseline algorithm, BSL, first computes a (necessarily large) set of candidate shortest paths and applies similarity filtering in a second step. In contrast, the OnePass algorithm traverses the network once and expands only those paths that qualify the similarity constraint. Our experiments on real road networks show that OnePass always outperforms BSL.

The rest of the paper is organized as follows. Section 2 introduces our approach on alternative routing and formally defines the problem of k-shortest paths with limited overlap. Section 3 describes two algorithms for the problem at hand, which are experimentally evaluated in Section 4. Finally, Section 5 concludes the paper.

2. PROBLEM DEFINITION

Let N denote a set of nodes that represent road intersections. A road network is a *directed graph* $G(N, E)$, where $E \subseteq N \times N$ is the set of edges, (n_x, n_y), each representing a road segment that connects nodes n_x and n_y. A weight function $w : E \rightarrow \mathbb{R}^+$ assigns to each edge (n_x, n_y) a weight w_{xy}, which captures the cost of moving from n_x to n_y, e.g., travel time or distance. A *(simple) path*, $p(s \rightarrow t)$, from node s to t (or just p if s and t are clear from the context) is a connected and cycle-free sequence of edges $p(s \rightarrow t) = \langle (s, n_x), \ldots, (n_y, t) \rangle$. The length $\ell(p)$ of a path p equals the sum of the weights for all contained edges, i.e.,

$$\ell(p) = \sum_{\forall (n_x, n_y) \in p} w_{xy}. \quad (1)$$

Next, we introduce the concept of an alternative path. Consider a set of paths P from a source node s to a target node t on a road network $G(N, E)$. We call any path $p(s \rightarrow t)$ alternative to P if p is sufficiently dissimilar to every path $p' \in P$. The similarity of a path p to another p' is determined by their overlap ratio:

$$Sim(p, p') = \frac{\sum_{(n_x, n_y) \in p \cap p'} w_{xy}}{\ell(p')}, \quad (2)$$

where $p_i \cap p$ denotes the set of edges shared by p and p'. For the overlap ratio we have $0 \leq Sim(p, p') \leq 1$, where $Sim(p, p') = 0$ holds if p shares no edge with p' and $Sim(p, p') = 1$ if $p \equiv p'$. Since we consider only simple paths, i.e., cycle-free, the similarity between different paths is strictly lower than 1. Using the above similarity measure, we formalize the concept of alternative paths in the following definition.

DEFINITION 1 (ALTERNATIVE PATH). *Let P be a set of paths from s to t and $\theta \in [0, 1)$ be a similarity threshold. A path p is alternative to P* iff *(a) p is also from s to t and (b) $\forall p_i \in P : Sim(p, p_i) \leq \theta$.*

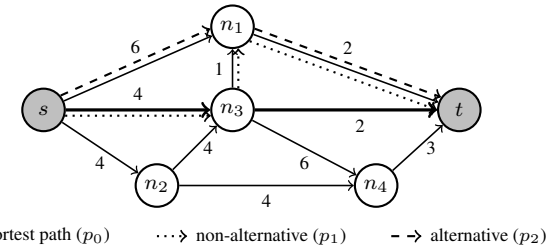


Figure 2: Example of a road network and alternative routing: shortest path $p_0\langle (s, n_3)(n_3, t) \rangle$, non-alternative path $p_1\langle (s, n_3)(n_3, n_1)(n_1, t) \rangle$, alternative path $p_2\langle (s, n_1)(n_1, t) \rangle$

Note that our definition of similarity in Equation (2) is asymmetric. We briefly discuss the intuition behind this choice. The concept of alternative paths is always defined between one *candidate* alternative path and a given set of alternative paths. Let p be a candidate path and $P = \{p_i, p_j\}$ a set of alternative paths. The goal is to determine whether p is an alternative path to set P, i.e., sufficiently dissimilar to paths p_i, p_j. In other words, set P is the fixed reference in this test and hence, the similarity of any candidate path depends on its overlap to p_i and p_j, i.e., the nominator of Equation (2).

Consider the road network in Figure 2. The shortest path from node s to t is p_0 with length $\ell(p_0) = 6$. Assume that set P contains only the shortest path, i.e., $P = \{p_0\}$. The figure also shows paths p_1 and p_2 as candidate alternative paths to set P; p_1 shares only edge (s, n_3) with p_0 and so, $Sim(p_1, p_0) = w_{s,n_3}/\ell(p_0) = 4/6 = 0.67$, while $Sim(p_2, p_0) = 0$ as paths p_0 and p_2 share no edge. Assuming a similarity threshold $\theta = 0.5$, only p_2 is in fact alternative to set P.

We now introduce the problem of *k-Shortest Paths with Limited Overlap* (k-SPwLO). Given a s and a target t, the goal is to recommend a set of k paths from s to t, sorted by length in increasing order such that (a) the shortest path $p_0(s \rightarrow t)$ is always included, (b) every path is dissimilar to its predecessors with respect to a similarity threshold θ, and (c) all k paths are as short as possible. Intuitively, this task can also be seen as progressively recommending k paths to the user starting from the shortest path p_0. Every path p_i recommended next is alternative to the set of paths already recommended and as short as possible. We formalize the k-SPwLO problem in the following definition.

DEFINITION 2 (k-SPWLO PROBLEM). *Given a source node s, a target node t, and a threshold θ, a query k-SPwLO (s, t, θ, k) returns a set $P_{LO} = \{p_0, \ldots, p_{k-1}\}$ of k paths from s to t, such that:*

- *p_0 is the shortest path from s to t,*
- *$\forall p_i, p_j \in P_{LO}$ with $i \neq j$: $Sim(p_i, p_j) \leq \theta$, and*
- *$\forall p \notin P_{LO}$:* either *$\ell(p) \geq \ell(p_i)$ holds $\forall p_i \in P_{LO}$* or *$\exists p_i \in P_{LO}$ with $\ell(p_i) \leq \ell(p)$ and $Sim(p, p_i) > \theta$.*

The first bullet of Definition 2 guarantees that the shortest path $p_0(s \rightarrow t)$ is always recommended. The second bullet assures that the recommended paths in P_{LO} are sufficiently dissimilar to each other. Finally, the third bullet guarantees that P_{LO} contains the shortest among all paths qualifying the previous constraints.

Consider again the road network of Figure 2 and the k-SPwLO$(s, t, 0.5, 2)$ query, i.e., $\theta = 0.5$ and $k = 2$. The result

Algorithm 1: BSL

Input: Road network $G(N, E)$, weight function W, source node s, target node t, number of results k, similarity threshold θ
Output: Set P_{LO} of k paths

```
1  initialize P_LO ← ∅;
2  while P_LO contains less than k paths and p_c not null do
3      p_c ← NextShortestPath(G, s, t);          ▷ Use Yen's alg.
4      if Sim(p_c, p_i) ≤ θ for all paths p_i ∈ P_LO then
5          add p_c to P_LO;                      ▷ Update result set
6  return P_LO;
```

set P_{LO} contains the shortest path $p_0 = \langle (s, n_3), (n_3, t) \rangle$ and the path $p_2 = \langle (s, n_1)(n_1, t) \rangle$ since $Sim(p_2, p_0) = 0 \leq 0.5$. Notice that although the path $p_1 = \langle (s, n_3)(n_3, n_1)(n_1, t) \rangle$ is shorter than p_2, the path p_1 is not recommended as it is too similar to p_0, i.e., $Sim(p_1, p_0) = 0.67 > 0.5$. In other words, with the recommended paths being dissimilar to each other, the result set $\{p_0, p_2\}$ is more attractive and valuable to the user compared to $\{p_0, p_1\}$.

3. PROPOSED ALGORITHMS

A naïve approach for computing k-SPwLO queries is to iterate over all paths from the source node s to the target node t and to compute their pairwise overlap ratio. However, such a solution is impractical as the computation of all possible paths from s to t is a $\#P$-complete problem [14]. Due to this examination order, the algorithms manage to consider in practice only a subset of all paths from s to t and hence, early terminate the search.

3.1 A Baseline Solution

Algorithm 1 illustrates our baseline algorithm, denoted BSL. It constructs the result set P_{LO} by examining paths from s to t in length order. More specifically, the shortest path $p_0(s \rightarrow t)$ is first added to P_{LO} (Line 1). Then, we enter a loop, where Yen's algorithm is invoked (Line 3) to generate the next path p_c from s to t in increasing length order. Note that Yen's algorithm does not run from scratch each time it is called, but continues from its previous state. For every path p_c, BSL checks whether the path is alternative to the already recommended paths in P_{LO} (Line 4). If this is the case, p_c is added to P_{LO}. BSL proceeds to the next path in line until P_{LO} contains k paths or all possible paths from s to t are examined.

3.2 The OnePass Algorithm

To boost k-SPwLO queries, we employ a pruning criterion based on the following observation. Let $p(s \rightarrow n)$ be a path from source node s to a node n and $p_i(s \rightarrow t) \in P_{LO}$ be an already recommended path. Assume that p is extended to reach the target t, resulting in path $p'(s \rightarrow t)$. As p' contains all edges shared by p and p_i, the similarity of p' to p_i is greater or equal to the similarity of p to p_i, i.e., $Sim(p', p_i) \geq Sim(p, p_i)$. Hence, given a threshold θ, if there exists a path $p_i \in P_{LO}$ such that $Sim(p, p_i) \geq \theta$, path p can be safely discarded as all extensions p' of p will also violate the similarity constraint. Lemma 1 captures this pruning criterion.

LEMMA 1. *Let P_{LO} be the set of already recommended paths. If p is an alternative path to P_{LO} with respect to a threshold θ, then $Sim(p', p_i) \leq \theta$ holds for every subpath p' of p and any $p_i \in P_{LO}$.*

Lemma 1 also shows the monotonicity of the similarity function. Let p_i be a path in P_{LO}, p be a path not in P_{LO}, and p' be a path that extends p, i.e., p is a subpath of p'. According to Lemma 1, the similarity $Sim(p', p_i)$ can only be greater or equal to $Sim(p, p_i)$, and it will not decrease under any circumstances.

Algorithm 2: OnePass

Input: Road network $G(N, E)$, weight function W, source node s, target node t, number of results k, similarity threshold θ
Output: Set P_{LO} of k paths

```
1  initialize P_LO ← shortest path p_0(s → t);
2  initialize min-heap H with source node s;
3  while P_LO contains less than k paths and H not empty do
4      [p_c, ℓ(p_c), V_Sim(p_c)] ← H.pop();          ▷ Current path
5      n_c ← the last node in p_c;
6      if n_c is target node t then
7          add p_c to P_LO;                          ▷ Update result set
8          foreach label [p_q, ℓ(p_q), V_Sim(p_q)] in H do
9              update V_Sim(p_q);
10             if Sim(p_q, p_i) > θ, ∀p_i ∈ P_LO then      ▷ Lemma 1
11                 remove [p_q, ℓ(p_q), V_Sim(p_q)] from H;
12     else
13         foreach outgoing edge (n_c, n) ∈ E do
14             create path p ← p_c ∘ (n_c, n);        ▷ Expand p_c
15             compute V_Sim(p);
16             if ∀p_i ∈ P_LO : Sim(p, p_i) ≤ θ then      ▷ Lemma 1
17                 H.push([p, ℓ(p), V_Sim(p)]);
18 return P_LO;
```

We next present OnePass, a label-setting algorithm which traverses the road network and expands every path from source s that qualifies the pruning criterion of Lemma 1. The paths are examined in increasing length order. OnePass keeps a label $[p, \ell(p), V_{Sim}(p)]$ for each distinct path $p(s \rightarrow n)$, where vector $V_{Sim}(p)$ records the similarity of p to every path in P_{LO}, i.e., $V_{Sim}(p)[i] = Sim(p, p_i)$. Each time a new path to the target node t is added to P_{LO}, an update procedure takes place for all remaining incomplete paths $p(s \rightarrow n)$, which updates $V_{Sim}(p)$ and checks p against Lemma 1. OnePass terminates when either k paths are recommended or all paths from s to t that qualify Lemma 1 are examined.

Algorithm 2 illustrates the pseudocode of OnePass. The algorithm uses a min-heap $\mathcal{H}$ (initialized with source s) to traverse the road network. The heap organizes the labels of the incomplete paths according to their length. T The result set P_{LO} is initialized with the shortest path, p_0, from s to t (Line 1). From Line 3–17, OnePass examines the contents of $\mathcal{H}$ until either k paths are recommended or the heap is depleted. At each iteration, the label $[p_c, \ell(p_c), V_{Sim}(p_c)]$ of the shortest path $p_c(s \rightarrow n_c)$ is deheaped from $\mathcal{H}$ (Line 4). If the end node n_c of the path p_c is the target t, a new path is added to P_{LO} (Line 7). Furthermore, for each label $[p_h, \ell(p_h), V_{Sim}(p_h)]$ in $\mathcal{H}$, the similarity vector $V_{Sim}(p_h)$ is updated, i.e., compute the similarity of p_h to the newly recommended path p_c and determine whether p_h qualifies the pruning criterion of Lemma 1. If $Sim(p_h, p_c) > \theta$, then p_h can be safely discarded[1]. If the end node n_c is not the target t, OnePass expands the current path p_c, considering all outgoing edges (n_c, n) (Lines 13–17), provided that the new path $p \leftarrow p_c \circ (n_c, n)$ qualifies the pruning criterion of Lemma 1 (Line 16).

3.3 Optimization

The performance of both BSL and OnePass can be improved by employing a lower bound, $\underline{d_N}(n, t)$, for the network distance $d_N(n, t)$ of a node n to the target t, i.e., the length of the shortest path from n to t. By using the lower bound $\underline{d_N}(n, t)$, the algorithms traverse the network in an A^* fashion. One approach for

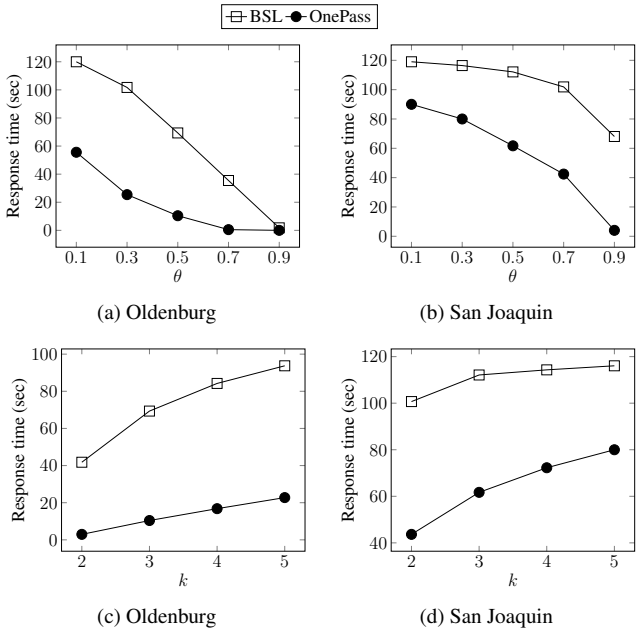
[1]In practice, our OnePass implementation performs lazy updates; vector $V_{Sim}(p_h)$ is updated and the pruning criterion of Lemma 1 is checked every time a $[p_h, \ell(p_h), V_{Sim}(p_h)]$ label is deheaped.

deriving such bounds is to precompute offline the distance of all nodes from/to a set of selected *landmark* nodes. A lower bound $\underline{d_N}(n,t)$ is then computed based on the triangle inequality [15]. An alternative approach to compute $\underline{d_N}(n,t)$ is to first reverse the edges of the road network and then run Dijkstra's algorithm from target t to every node n of the network [16]. Without loss of generality, our implementations of BSL and OnePass adopt the second approach as determining the best method for $\underline{d_N}(n,t)$ is out of the scope of this paper.

4. EXPERIMENTAL EVALUATION

For our experiments we used the road networks for the city of Oldenburg (6,105 nodes, 14,058 edges) and the city of San Joaquin (18,263 nodes, 47,594 edges). To assess the performance of our algorithms, we measure the average response time over 1,000 queries (i.e., pairs of nodes), varying (a) the number of requested paths k and (b) the similarity threshold θ. On each test, we vary one of the parameters while fixing the other to its default value (3 for k and 0.5 for θ). Due to the high execution time of BSL, we consider a timeout of 120 secs for each query. All algorithms were implemented in C++ and the tests run on a Quad-Core Intel Xeon X5550 (2.67GHz) with 48GB of RAM running Ubuntu Linux.

Figure 3 reports the response time of our algorithms while varying parameters θ and k. We observe that both BSL and OnePass run faster as the similarity threshold θ increases (Figure 3(a), (b)), and slower as the number of requested paths k increases (Figure 3(c), (d)). As expected, BSL is impractical due to the large number of paths examined in length order, i.e., the paths returned by Yen's algorithm. On the other hand, OnePass is faster than BSL because a number of incomplete paths can be pruned using Lemma 1. Our tests also unveiled an important trade-off. As θ increases, the pruning power of Lemma 1 deteriorates, i.e., more incomplete paths qualify the pruning condition, and OnePass constructs more paths (supporting measurements omitted due to lack of space). But at the same time, the next recommended path will be naturally determined earlier and hence, the total execution time drops.



(a) Oldenburg (b) San Joaquin

(c) Oldenburg (d) San Joaquin

Figure 3: Performance comparison while varying similarity threshold $\theta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for $k = 3$, and requested paths $k \in \{2, 3, 4, 5\}$ for $\theta = 0.5$. Timeout is set to 120 seconds.

5. CONCLUSIONS

We studied the problem of alternative routing on road networks. Our goal was to recommend k paths that are sufficiently dissimilar to each other and as short as possible. We formalized this task as the k-Shortest Path with Limited Overlap problem and designed two evaluation algorithms. Our experimental analysis showed that the OnePass algorithm is always faster than BSL.

Future work includes the study of various optimizations in our proposed algorithms as well as approximation solutions. Furthermore, we plan to extend the definition of alternative routing by considering additional constraints and criteria besides the overlap between the paths and their length.

6. REFERENCES

[1] H. Bast, D. Delling, A. Goldberg, M. Müller, T. Pajor, P. Sanders, D. Wagner, and R Werneck. Route Planning in Transportation Networks. Technical report, 2014.

[2] L. Wu, X. Xiao, D. Deng, G. Cong, and A. D. Zhu. Shortest Path and Distance Queries on Road Networks: An Experimental Evaluation. In *VLDB*, pages 406–417, 2012.

[3] J. Y. Yen. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, 1971.

[4] E.Q.V. Martins and M.M.B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR*, 1(2):121–133, 2003.

[5] Y.J. Jeong, T.J. John Kim, C.H. Park, and F.K. Kim. A Dissimilar Alternative Paths-search Algorithm for Navigation Services: A Heuristic Approach. *KSCE Journal of Civil Engineering*, 14(1):41–49, 2009.

[6] Choice Routing. Technical report, Cambridge Vehicle Information Technology Ltd., 2005.

[7] R. Bader, J. Dees, R. Geisberger, and P. Sanders. Alternative Route Graphs in Road Networks. In *ICST TAPAS*, volume 6595 LNCS, pages 21–32, 2011.

[8] I. Abraham, D. Delling, A.V. Goldberg, and R.F. Werneck. Alternative Routes in Road Networks. *Journal of Experimental Algorithmics*, 18(1):1.1–1.17, 2013.

[9] A. Akgun, E. Erkut, and R. Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121:232–246, 2000.

[10] Y Lim and H Kim. A Shortest Path Algorith for Real Road Network based on Path Overlap. *Journal of EASTS*, 2005.

[11] D. Delling and Wagner D. Pareto Paths with SHARC. In *SEA*, pages 125–136, 2009.

[12] H.P. Kriegel, M. Renz, and M. Schubert. Route Skyline Queries: A multi-preference Path Planning Approach. In *ICDE*, pages 261–272, 2010.

[13] M. Shekelyan, G. Jossé, and M. Schubert. Linear path skylines in multicriteria networks. In *ICDE*, 2015.

[14] Valiant L.G. The Complexity of Enumeration and Reliability Problems. *Siam Journal of Computing*, 8(3):410–421, 1979.

[15] A.V Goldberg and C. Harrelson. Computing the shortest path: *A* search meets graph theory. In *ACM-SIAM SODA*, pages 156–165, 2005.

[16] D. Sacharidis and P. Bouros. Routing directions: keeping it fast and simple. In *ACM SIGSPATIAL*, pages 164–173, 2013.