Shortest-Path Diversification through Network Penalization: A Washington DC Area Case Study

Dan Cheng
George Mason University
dcheng4@gmu.edu

Olga Gkountouna
George Mason University
ogkounto@gmu.edu

Andreas Züfle
George Mason University
azufle@gmu.edu

Dieter Pfoser
George Mason University
dpfoser@gmu.edu

Carola Wenk
Tulane University
cwenk@tulane.edu

ABSTRACT

Traditional navigation systems compute the quantitatively shortest or fastest route between two locations in a spatial network. In practice, a problem resulting from all drivers using the shortest path is the congregation of individuals on routes having a high in-betweenness. To this end, several works have proposed methods for proposing alternative routes. In this work, we test solutions for traffic load-balancing by computing diversified routes proposing variants of the penalty method using the road network of the Washington DC metropolitan area as a case study. Our experimental evaluation shows that the tested Penalty-based approaches allow to significantly balance the load of a spatial network, compared to existing k-shortest path algorithms, and compared to a naive baseline that randomly changes the weights of the network at each shortest-path computation.

CCS CONCEPTS

• **Information systems** → Data analytics; • **Applied computing** → **Transportation**.

KEYWORDS

k-Shortest Paths, Diversification, Penalization, Load balancing, Road Network, Traffic

ACM Reference Format:
Dan Cheng, Olga Gkountouna, Andreas Züfle, Dieter Pfoser, and Carola Wenk. 2019. Shortest-Path Diversification through Network Penalization: A Washington DC Area Case Study. In *12th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS'19), November 5, 2019, Chicago, IL, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357000.3366137

1 INTRODUCTION

Efficient transportation solutions are of paramount importance in our society. According to the latest INRIX Global Traffic Scorecard [9], drivers in the United States wasted $305 billion while stuck in traffic in 2017 alone. This number is derived from more than 11 billion liters of wasted fuel per year, and 6.9 billion of man-hours stuck in traffic per year [21]. In many geographic areas, human mobility is dominated by individualized transportation solutions. Here efficient navigation solutions play a key role. Applications such as Google Maps derive their service quality from crowdsourcing the traffic situation in real time. However, while sensing the crowd to come up with traffic-sensitive shortest paths, the service then actually targets individuals and not the collective user base (drivers) with its solutions. Such micro-level shortest-path solutions might lead to systemic problems such as additional traffic jams caused by users of the same application. While providers have addressed this problem, shortest-path solutions still do not have a macro-level perspective across applications when generating solutions.

Our goal with this work is to look at this key aspect of individualized transportation and see how we can improve it for all rather than just for a few users. We evaluate approaches for shortest-path computation methods that go "the extra mile" to provide greater variety without compromising too much on path length (travel time). Our approach is motivated by the observation that, for a given source-target pair, there are often a number of different near-shortest paths that actually do not differ too much in length. Such a method would facilitate load balancing and help minimize the aggregate travel time for all users by better utilizing the overall road network capacity.

The problem solved in this work is to find a set of paths between a designated pair of vertices in a given directed weighted spatial network. Our challenge is to diversify these paths, thus minimizing the redundancy of edges between these paths, but while also keeping the average length of these paths low. To distribute traffic, a classic method of diversifying routes between source-target pairs is the k-shortest path. However, as also shown by our experiments, the k-shortest path will often make very minor detours, or even incur loops, that do not improve traffic conditions in practise.

We show that our penalty-based approache can achieve better and more diverse results. Rather than changing the shortest-path algorithm, we adapt the underlying spatial network, to incentivize the use of road segments that would have a higher associated cost. In particular, the penalization approach increases the weight of road segments that have been used on previous routes. As the original weight of a segment of the road network, we use the minimum expected travel time, which we calculate as the length of the segment divided by its speed limit. On the other hand, we use as a baseline

IWCTS'19, November 5, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6967-1/19/11…$15.00
https://doi.org/10.1145/3357000.3366137

an approach that changes the weight of all the road segments of the network randomly before each shortest path computation.

The contributions of this paper include the following:

(1) We formally describe the problem of shortest path diversification.
(2) In addition to naive solutions, we evaluate an approach based on network penalization. This approach can be parameterized to control the trade-off between length of the paths, and their diversity.
(3) We perform qualitative and quantitative experimental evaluations using the Washington DC road networkas a case study, showing that the penalty-based approach is capable of vastly diversifying the choice of shortest path compared to state-of-the-art solutions.

The remainder of this paper is organized as follows. In Section 2, we survey existing work on k-shortest path computation and load balancing on road networks. Then, Section 3 formally describe the problem of shortest path diversification. All the tested solutions, including a naive solution and the penalization based solution, are presented in Section 4. Our experimental evaluation, both qualitatively and quantitatively is presented in Section 5, and we conclude the paper in Section 6.

2 RELATED WORK

A related research problem to path diversification is the k-shortest path problem (KSP) [3, 12, 25], which is about finding the k shortest paths from a source vertex s to a target vertex t in a directed weighted graph G for an arbitrary natural number k. The set $K_{sp}$of k shortest path is defined as the k-set (having k elements) of paths between s to t, such that there exists no other path $\pi \notin K_{sp}$ that is shorter than any path in K_{sp}.

We argue that solutions of the k-shortest path problem are not sufficient to diversify paths between s and t in a meaningful way. In practise, the k shortest paths only exhibit minor differences, such as cutting a corner, or take an additional loop in a roundabout. Variants of the KSP problem have been studied in the literature that constrain paths to be simple [15, 25], thus having no repeat vertices which means that no state is repeated along any solution path. As shown in our experimental evaluation in Section 5, the 500 shortest paths avoiding loops between random source and target vertices in the road network of the Washington DC metro area, have more than 99% overlap in terms of edges, thus making them impractical for applications such as traffic load balancing.

A strict way to enforcing diversity between paths is defined by the problem of k-shortest disjoint paths [22, 23]. An efficient algorithm to compute edge-disjoint and vertex-disjoint paths was proposed in [6]. These algorithms are motivated by communication networks, where the goal is to maximize the probability that any of the paths reaches the destination. However, these solutions are inapplicable in a road network, where vertices are connected sparsely: For any source vertex of degree n there must not be more than n disjoint paths to a destination vertex. In an extreme case, where the source vertex is located in a dead end (a vertex of degree one), then there exist only one disjoint path, thus allowing for no diversification via disjoint paths.

Another related field is the problem of load balancing in communication networks [5, 14, 20], for which reconfiguration of the network is used to relax the load on a single vertex or edge. While theoretically intriguing to temporarily close roads of a road traffic network to balance the load among the network, it is not practically possible. Our approach differs, as we do not change the topology of the network, but we adjust the weights of edges used for navigation, in order to force shortest-path algorithms to diverge.

Several works have been proposed to address the problem of finding alternative routes in road networks [1, 2, 4, 7, 8, 16–19]. First, [7] formalized the problem of finding k-Shortest Paths with Limited Overlap (k-SPwLO). Their goal is to recommend k alternative paths which are as short as possible, and sufficiently dissimilar based on a user-controlled similarity threshold. They proposed a baseline solution and a faster OnePass algorithm. This problem of finding top-k shortest paths with diversity (KSPD) has later been shown to be a NP-hard problem [18]. The work of [2] treats path enumeration as a "p-dispersion" problem to generate spatially dissimilar routes, by applying a heuristic that searches for an initial k-shortest paths solution and performs a local search based on minimizing similarity to improve this initial solution. The notion of 'admissible paths' is introduced in [1] to find "reasonable" alternative routes that feel natural to the driver. Based on this work, [16, 19] use the concept of 'candidate via nodes' to further speed up the computation. Most related to our work, the notion of an 'alternative graph' that contains the set of reasonable alternative routes is introduced in [4]. This work uses k-shortest paths and a variation of the penalty method proposed in this work. Recent works reduce the execution time of the penalty method, using dynamic runtime adjustments [17] and based on customizable route planning [10].

In this work, the goal is not to propose new algorithms for shortest path diversification and load balancing in road network. Rather, our goal is to leverage existing related work to study how different solutions for this problem affect the distribution of traffic in the road network of the Washington DC area.

3 PROBLEM DEFINITION

In this section we formalize our definition of shortest path diversification. We first define a spatial network, as follows.

Definition 3.1 (Spatial Network). A spatial network is a directed graph Let $G = (V, E)$ with $n := |V|$ vertices and $m := |E|$ edges where each edge $(u \in V, v \in V) \in E$ has a non-negative weight $f(u, v)$.

Further, we define a path between two vertices in G as follows:

Definition 3.2 (Paths). Let $s, t \in V$ be two distinct vertices of graph G. A path $P_{s,t}$ from a source vertex s to a sink vertex t in G is a sequence of vertices $(s = v_1, v_2, \ldots, v_q = t)$, where $(v_i, v_{i+1}) \in E, i = 1, \ldots, q-1$.

The *length* $l(P_{s,t})$ of path $P_{s,t}$ is defined as the sum of weights of all its edges: $l(P_{s,t}) = \sum_{i=1}^{q-1} f(v_i, v_{i+1})$. The number of edges in $P_{s,t}$ is denoted by $|P_{s,t}|$. A path $P_{s,t}$ is called *simple* if no vertex is repeated, i.e., if for any $v_i, v_j \in P_{s,t}$ it holds that $i \neq j \rightarrow v_i \neq v_j$.

The set of all simple paths from s to t in $G = (V, E)$ is denoted by $\Pi_{s,t}$. The *shortest path* from s to t, denoted by $P_{s,t}$, is a simple

path from s to t in G that has the minimum length among all paths in $\Pi_{s,t}$.

As the aim of this work is to find a set of paths between two vertices s and t that are diverse, we first proceed to define a measure of diversity of a set of paths. We define the *diversity* of a set of paths $\pi_{s,t}$ as the total length (weight) of all edges that are traversed by any path in $\pi_{s,t}$, normalized by the length of the shortest path, formally:

Definition 3.3 (Diversity). Let $\pi_{s,t}$ be a set of simple paths in G between vertices s and t, we can define the *diversity of* $\pi_{s,t}$ as

$$\mathcal{D}(\pi_{s,t}) = \frac{\sum\limits_{e \in \mathcal{S}_{\pi_{s,t}}} f(e)}{l(P_{s,t})} - 1 \ ,$$

where $\mathcal{S}_{\pi_{s,t}} = \bigcup\limits_{P \in \pi_{s,t}} \{e \in P\}$ the set of all edges traversed by any path in $\pi_{s,t}$.

If there is no path diversity ($\mathcal{D} = 0$) in $\pi_{s,t}$, then all paths coincide with the shortest path. Otherwise, non-overlapping edges of diverse paths will result in a positive diversity measure.

Having a diversity measure, we can now define the k-diversified path problem as follows.

Definition 3.4 (k-Diversified Shortest Paths). Given source and target vertices s and t, we want to find a set $\pi_{s,t}$ consisting of any k paths for which all of the following conditions are met:

(1) Each path $P \in \pi_{s,t}$ is a simple path from s to t in G.
(2) The sum of the lengths of all paths $\sum\limits_{P \in \pi_{s,t}} l(P)$ is minimized.
(3) The *diversity* $\mathcal{D}(\pi_{s,t})$ is maximized.

There is no unique solution to this problem as it depends on the trade-off between the total weight of the set of all paths and the overall diversity of the result. Any solution that is not dominated in terms of both total weight and diversity is an acceptable solution. It is then up to the final user to choose their best preference. If we represented the full set of possible solutions as points on the 2-dimensional space of total weight vs. diversity, the set of all acceptable solutions would be the result of a *skyline* query of low weight and high diversity [24].

4 METHODOLOGY

To address the challenge of computing diversified shortest paths, we consider a range of iterative approaches that are based on manipulating the edge weights of the road network graph during subsequent runs of the SP algorithm. The methods differ how we adjust the weights, i.e., based on (i) randomizing weights or by (ii) penalizing the weights that already contributed to shortest paths. The experimental study in Section 5 compares the various instances of algorithms of each group under different quality criteria.

4.1 Edge Weight Randomization

A first, naive approach to diversify the paths between two locations is to randomize the weights of the underlying spatial network to lure traditional shortest path algorithms into deviating from the shortest path. Each time a shortest path is computed, we simply add unbiased random noise to the weights of edges. Thus, for each path computation, the weight of each edge is the result of a random variable, having the original weight as its expected value. On the randomized spatial network, we employ a traditional shortest path algorithm.

For this baseline approach, we use two parameters, δ and τ to control the degree of randomization.

Definition 4.1 (Edge Weight Randomization). Let $G = (V, E)$ be a spatial network. Path diversification using edge weight randomization adds an unbiased, normally distributed weight to each edge e to the network. This is done via the function $f_{rand}(e)$ defined as follows:

$$f_{rand}(e) = max(f(e) + N(0, f(e)^2 \cdot \delta^2), \tau) \ , \quad (1)$$

where $N(0, f(e)^2 \cdot \delta^2)$ is a Gaussian distributed random variable having zero mean, and having a variance of $f(e)^2 \cdot \delta^2$. The rationale for choosing this variance is to obtain a standard deviation of $f(e) \cdot \delta$. For instance, for $\delta = 0.1$, we expect 68% of the area under the curve to be within $f(e) \pm 0.1 f(e)$. The parameter τ is a small constant that is used to ensure that the Gaussian error does not yield negative or zero-cost edges.

ALGORITHM 1: Graph Randomization (GR)

```
Data: Graph G = (V, E, f); source vertex s; target vertex t; number
      of paths k; noise parameter δ
Result: Set of k paths π_{s,t} = {P^1, P^2, ..., P^{k-1}}
1  π_{s,t} := ∅;
2  for e ∈ E do
3  |  w_e := f(e); //initialization of weights
4  end
5  for i = 0, ... k − 1 do
6  |  P^i := Dijkstra(V, E, w_e; s, t);
7  |  π := π ∪ {P^i};
8  |  for e ∈ E do
9  |  |  w_e := max(f(e) + N(0, f(e)^2 · δ^2), τ);
10 |  end
11 end
12 return π_{s,t};
```

The pseudo-code of the Graph Randomization approach is presented in Algorithm 1. Initially the result trajectory set π is empty. The weight w_e for each edge $e \in E$ is initialized based on the original weight function f of the graph G in lines 2-4. The loop in lines 5-11 computes each of the k paths iteratively. In the first iteration the actual shortest path $P^1_{s,t}$ is computed, based on the original weights of the edges of the graph, by invoking Dijkstra's algorithm in line 6. The resulting path is added to the set π in line 7. Then the weights of all edges in G are updated in the lines 8-10 using the formula given by Equation 1. Then the same process is repeated for the next path, which is calculated by calling Dijkstra's algorithm again, this time based on the randomized weights of the edges. After k iterations, the result set π, consisting of k paths, is returned and the algorithm terminates.

A variation of this approach, is to only randomize the edges that belong to the previous path, instead of all the edges of the graph. This would require line 8 of Algorithm 1 to be modified to only consider all edges $e \in P^i$ instead. We term this approach Path Randomization (PR).

It is worth noting that the result of the weight randomization approach is non-deterministic. Different executions of the same code on the same graph may yield different sets of paths and the diversity of each result is not guaranteed.

We note that the Edge Weight Randomization may not be able to diversify the shortest path. In particular, when the shortest path is much shorter than the second shortest path, the shortest path may be returned an infinite number of iterations.

Theorem 4.2. *Let $G = (V, E, f)$ be a finite spatial network with positive integer weights, i.e., $f : E \rightarrow \mathbb{Z}^+$, and $\Pi_{s,t}$ denote the set of simple shortest paths between vertices s and t in G. The Edge Weight Randomization method may return the same path an infinite number of times in a row.*

Proof. Let P^1 be the path returned in the first iteration. Since the weights follow the same distribution in each iteration, there is a non-zero probability p that the same path may be chosen in the second iteration, i.e., it is possible that $P^1 = P^2$. Now, let P^k be the path returned in iteration k. Again, since the weights follow the same distribution in each iteration, there is a non-zero chance $p_r > 0$ that the same path may be chosen in iteration P^{k+1}. By mathematical induction, we conclude that there is a probability $p_r > 0$ to obtain the same result as in the previous iteration $i \in \mathbb{N}$. Since weights are randomized independently in each iteration, the probability of having the same result after k iterations is $p_r^k > 0$. □

In the following section, we describe a path penalization method that memorizes the number of times each edge has been visited before, and penalized edges depending on this number of make paths using other edges more attractive. We further show that this approach is guaranteed to find a new path after a finite number of iterations.

4.2 Edge Weight Penalization

The penalization approach tries to introduce diversity to the set of computed shortest paths by penalizing the weights of edge once they contributed to a shortest path.

In its first iteration, this method calculates the actual shortest path $P^1_{s,t}$ as $\{s = v^1_1, v^1_2, \ldots, v^1_q = t\}$. For the second iteration, we adjust the weights of all edges (v^1_i, v^1_{i+1}) and $i = 1, \ldots, q-1$ as $f'(v^1_i, v^1_{i+1})$, by increasing their respective weights to $f' = f \cdot (1+p)$. We refer p as the penalization factor.

The pseudo-code of our penalization based approach is presented in Algorithm 2. The initialization of the original weights of every edge in E is the same as that of Algorithm 1. The penalization method differs in lines 8-10, where only the edges of the current path P^i are penalized by a (fixed deterministic) factor p. Note that the penalization is cumulative; if an edge has been used by a previous path and its original weight has been increased by $p \cdot w_e$, then it will again be penalized by an additional $p \cdot w'_e$ based on its current weight w'_e if it is used again in a subsequent path.

This approach guarantees that, regardless of the choice of the penalization factor p, the chosen path will change after a finite number of iterations.

Theorem 4.3. *Let $G = (V, E, f)$ be a finite spatial network with positive integer weights, i.e., $f : E \rightarrow \mathbb{Z}^+$, and $\Pi_{s,t}$ denote the set of simple shortest paths between vertices s and t in G. If $|\Pi_{s,t}| > 1$, then for any $p > 0$, it takes a finite number of iterations of the Path Penalization method for the shortest path to differ by at least one edge.*

ALGORITHM 2: Path Penalization (PP)

Data: Graph $G = (V, E, f)$; source vertex s; target vertex t; number of paths k; penalization factor p
Result: Set of k paths $\pi_{s,t} = \{P^1, P^2, \ldots, P^k\}$

```
1  π_{s,t} := ∅;
2  for e ∈ E do
3  |  w_e := f(e); //initialization of weights
4  end
5  for i = 0, ... k − 1 do
6  |  P^i := Dijkstra(V, E, w_e; s, t);
7  |  π_{s,t} := π_{s,t} ∪ P^i;
8  |  for e ∈ P^i do
9  |  |  w_e := w_e · (1 + p);
10 |  end
11 end
12 return TR;
```

Proof. Let P^i be the shortest path computed during an iteration i and let $f^i : E \rightarrow \mathbb{Z}^+$ be the edge-weight function after these i iterations. For a path, or a set of edges, P, let $l^i(P)$ denote the total length of its edges, after i iterations. In each subsequent iteration of Algorithm 2 that results in the same shortest path, all edges along P^i have their weight increased by a factor of $(1 + p)$. Now, assume that the same path P^i is chosen for k iterations, i.e., assume that $P^i = P^{i+1} = \ldots = P^{i+k}$. This means that the total length of P^i will have increased to $(1 + p)^k \sum_{e \in P^i} f^i(e)$. Let $P_x \in \Pi_{s,t}$ denote another simple path such that $P_x \neq P^i$. Such path must exist, as we assumed that $\Pi_{s,t}$ has at least two elements. Let $P^i_x \subset P_x$ denote the set of edges shared by P^i and P_x. Since any shortest path in $\Pi_{s,t}$ is simple, we know that P^i must contain at least one edge that is not in P_x, i.e., $P_x \setminus P^i_x \neq \emptyset$. After $i + k$ iterations, the length of P_x is given by

$$l^i(P_x) = l^i(P_x \setminus P^i_x) + (1 + p)^k \cdot l^i(P^i_x)\,, \tag{2}$$

while the length of P_i is given by

$$l^i(P^i) = (1+p)^k \cdot l^i(P^i_x) = (1+p)^k \cdot l^i(P^i_x) + (1+p)^k l^i(P^i \setminus P^i_x)\,. \tag{3}$$

Combining (2) and (3) we obtain

$$l^i(P_x) < l^i(P^i) \quad \Leftrightarrow \quad l^i(P_x \setminus P^i_x) < (1 + p)^k l^i(P^i \setminus P^i_x) \tag{4}$$

$$\Leftrightarrow \quad \log_{1+p} \frac{l^i(P_x \setminus P^i_x)}{l^i(P^i \setminus P^i_x)} < k\,. \tag{5}$$

Since we assume that all edges have finite weights, there must exist a finite k such that $l^i(P_x) < l^i(P^i)$. Thus, P^i is no longer the shortest path, since there exists at least one path, namely P_x that is shorter. Let P_y be the shortest path after k iterations. Again, since all paths in $\Pi_{s,t}$ are simple, there must be at least one edge in P_y that is not in P^i. □

Figure 1: The distribution of speed limits of the network

Theorem 4.3 guarantees that this path penalization based approach will eventually yield a new path, unless there is only one simple path between source and target vertices.

5 EXPERIMENTS

In this section, we describe experimental evaluation, qualitative and quantitative, comparing the penalization based approach against various baseline solutions. All the approaches were implemented in python and tested on a 16GB Ram Intel Xeon CPU 3.0GHz Dell workstation, running Windows 10.

5.1 Data

To evaluate the aforementioned approaches, we used a real transportation network. In particular, we used the road network of the Washington DC Metropolitan Area, which we obtained from Open Street Map [13]. For every edge e we assign its weight as $f(e) = \frac{l_e}{v_e}$, where l_e is the length and v_e the speed limit of the edge. This weight equals the minimum *travel time* of that edge, and simulates free-flow travel at maximum speed. Figure 1 presents a visualization of the distribution of speed limits of every road on our network. For our experiments we randomly generated a set of 25 location pairs to be used as sources and targets.

5.2 Method Overview

We have included the following shortest-path computation methods in our experimental analysis:

- **PP**: We evaluate the Path Penalization (PP) based approach (as described in Section 4.2) for varying values of the penalization factor $p \in \{0.01, 0.03, 0.1, 0.3\}$.
- **PP** $p \to \infty$: A variant of the PP method with large path penalties $W >> \max_{e \in \mathcal{E}} f(e)$. This simulates the k-shortest *disjoint* path problem[22, 23], where an edge cannot be included in two paths, and hence an edge is removed once it has been used in a shortest path.
- **GR**: A baseline method that randomizes the weights of all graph edge after each shortest path iteration.
- **PR**: A baseline variant that randomizes the weights of only those edges that belonged to the path computed in the previous iteration.
- **k-Shortest Paths**: Original k-shortest path method. We used an implementation [1] of Yen's algorithm [25].

[1] https://github.com/beegeesquare/k-shortest-path

Unless stated otherwise, we use the standard setting of $k = 100$ paths. For the "Penalization $p \to \infty$" approach, W is a very big number, treated as 'infinity'. This approach sets any visited edge to the same weight W to ensure that all 'infinities' are treated equal. At the same time, W is a finite number to avoid problems of connectivity when no more paths between source and target exist after edges are removed. For example, there may be less than k edges connected to the source or to the target vertex. In this case, the algorithm is forced to minimize the number of incurred W-edges first, then minimizing the remaining shortest path last. In the extreme case that all edges have been visited, the next path will be the one with the fewest number of edges, regardless of their original weight.

5.3 Quality Criteria

To assess the quality of the various algorithms in terms of sets of shortest paths they produce we use three specific criteria: *Entropy*, the *Redundancy* and *Diversity*. While diversity has already been defined in Section 3, we give definitions for the other measures as follows.

First, we define the *load* of an edge, which we will require for other definitions in this section.

Definition 5.1 (Edge Load). Let $\pi = \{P^1, ..., P^n\}$ be a set of paths in G. For any edge $e \in E$ its *load* is defined as the number of paths traversing e:

$$\mathcal{L}_\pi(e) = |\{P \in \pi \mid e \in P\}|$$

For each edge $e \in E$, the *normalized load* $\mathcal{L}_\pi(e)/|\pi|$ is between 0 and 1.

Definition 5.2 (Load Balance Entropy). Given a set of paths π in a road network graph G, we want to measure how distributed the paths are with respect to the road network. We refer to this quality as the *entropy of the load distribution.*

$$\mathcal{E}_\pi = -\sum_{e \in G} \frac{\mathcal{L}_\pi(e)}{|\pi|} \cdot \log_2 \frac{\mathcal{L}_\pi(e)}{|\pi|} .$$

If a set of paths is not diverse and they considerably overlap, then for each edge e the normalized load $\mathcal{L}_\pi(e)/|\pi|$ is either 0 (if no path traverses e) or 1 (if all paths traverse e). In other words, when the diversity is lowest then the entropy is 0. On the other hand, if the normalized load is evenly distributed in the network then the entropy is higher. Note that by definition of $\mathcal{L}$, it always holds that $\mathcal{L}_\pi(e) \leq |\pi|$, therefore, the argument of the logarithm is always less than or equal to 1, and therefore the entropy is always non-negative.

Definition 5.3 (Redundancy). For a given set of paths π in G, we denote $\mathcal{S}_\pi = \bigcup_{P \in \pi} \{e \in P\}$ the set of all edges traversed by any path in π. We define the *redundancy* of a set of paths as:

$$\mathcal{R}_\pi = \frac{\sum_{P \in \pi} |P|}{|\mathcal{S}_\pi|}$$

Intuitively, *redundancy* is an indicator of the degree of popular edges in a set of paths. Specifically, it is the fraction of the total number of edges of all paths, divided by the total number of unique edges of all paths. This quantity represents the average utilization of

edges that appear in at least one trajectory. If all paths are identical then $\mathcal{R} = |\pi|$, the total number of paths. Whereas, if there is no overlap and each edge is used at most once in a path, then $\mathcal{R} = 1$.

5.4 Qualitative Evaluation

Before providing a quantitative assessment of our results in terms of the metrics defined above, we provide a few visual examples of what kind of paths the different methods produce. Figure 2 shows a map excerpt of the Washington DC metropolitan area. The six maps visualize different k-shortest paths results for the same source and target vertex pair, but in each case based on different penalization and randomization parameters. Figure 2(a) shows the shortest path calculated by Dijkstra's algorithm [11], while Figure 2(b) shows the top 100 shortest paths given by Yen's algorithm [25]. We observe that, the resulting 100-shortest paths are all extremely similar to the shortest path. We see that Yen's k-shortest paths do not introduce any meaningful diverse paths, but only small local detours to the actual shortest path. As such, this example serves as a good motivation for our work.

Figures 2(c) through 2(h) show paths computed by alternative approaches aiming at increasing the diversity of the paths. Assuming the paths are comparable in terms of length (using travel time as a metric as described in Section 5.3), then this would allow us from a traffic management point of view, to distribute the traffic load in a more meaningful way. In each figure, the actual shortest path is shown as a black dashed line, while diversified paths are shown in blue. Using transparency, stacked paths appear darker (more opaque) and indicate more popular routes. Figure 2(c) shows a set of paths computed by the GR method (randomized graph), which re-assigns random weights to all edges of the road network in between iterations. Figure 2(d) shows the 100 shortest paths obtained by the PR method, a path randomization approach which randomizes only those edges that belong to the $(i-1)$-st shortest path when calculating the i-th shortest path. Both randomization approaches marginally increase the diversity of generated paths and most paths still reuse a large portion of the actual shortest path. It must be noted that the results of these two approaches are non-deterministic, i.e., different runs produce slightly different results. However the aforementioned figures are representative, as the degree of diversity was similar for different runs although the individual paths were not quite identical.

Figures 2(e)-2(h) shows the 100 shortest paths resulting from the PP diversification approach using varying penalization factors. As the penalization factor increases, the resulting set of 100 paths becomes more diverse. For Figure 2(e), every used edge was penalized by increasing its weight by 1%. After each new path calculation we penalize all its edges before computing the next path. Penalization is accumulative. Figures 2(f) and 2(g) show the results of the same approach for penalization factors of 3% and 10% respectively.

Figure 2(h) illustrates the results of the penalization based approach when the penalization factor approaches infinity, which is similar to removing an edge once it has been used in any path. The problem with such a method is that there may be connectivity issues after a few iterations. For instance, the number of edges incident to the source or target may be less than k. Instead of setting the weights to infinity (i.e., removing edges from the graph), we choose a very large weight instead, e.g., $W >> \max_e(w_e)$. This severely discourages the use of this edge in a path, but does not remove completely in case of connectivity issues. For example, if the only way from a source to a target is by using a bridge, then this road network edge must be used, even if its weight has been set to W. This approach also allows for the extreme case in which all edges have been used in a path already. Here the next path will be the one with the fewest number of edges, since all edges have the same weight (W). As can be seen in Figure 2(h), this approach has the largest diversity, covering most of the road network. However, many of the paths become much longer than the original shortest path incurring unreasonable detours.

5.5 Experimental Results

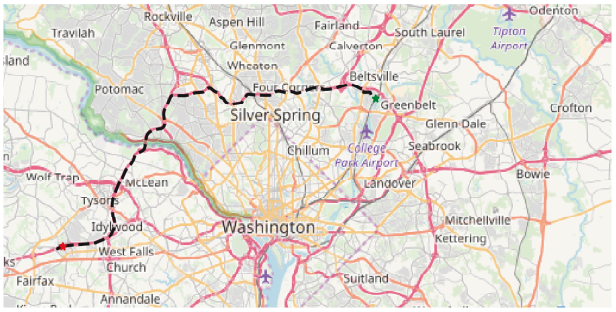
In the following, we show the results of our quantitative studies, measuring diversity and path length for all approaches described in Section 5.2, using all quality criteria described in Section 5.3.

5.5.1 Travel Time Evaluation. Figure 3 shows the travel time for each of the 100 paths computed by the different approaches illustrated for a single source and target vertex pair in Figure 2. The $k = 100$ shortest paths computed using the original k shortest-path algorithm [25] have almost identical travel times. The example of Figure 2(b) shows that the 100 paths overlap considerably, and hence have no diversity. The two randomization approaches, GR and PR, also fail to achieve diversity, as the travel times of the paths in their result sets are quite close to the original shortest path.

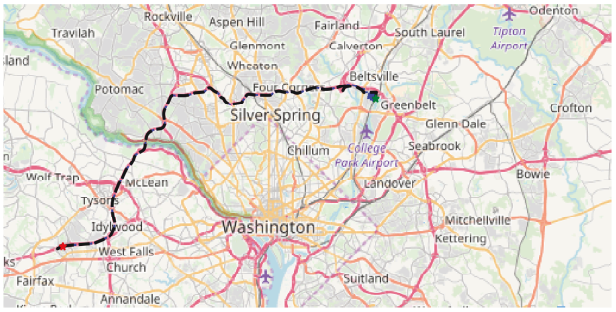
The *path penalization* approaches (PP) on the other hand exhibit a higher variance in the total lengths (travel times) of the computed paths. This variance increases with the number of iterations. For larger penalization factors the average path length increases and so does the variance between subsequent iterations (spikes). For p=0.01, i.e., a 1% increase in edge weight, the path travel times increase moderately in comparison to the actual shortest path. For p=0.3, the paths on average double in length after about 60 iterations. In the extreme case of the $p \to \infty$ approach, the travel times of paths grow up to 3 times larger than the shortest path after 30 iterations. Also, the variance increases significantly. Following 74 iterations, it plateaus since all edges have been visited and all alternative paths have been exhausted. Following this, the shortest path will always be the one with the fewest number of edges (minimum "hops"), which is not necessarily the actual shortest path.

5.5.2 Comparison of Different Source Target Pairs. What follows is a comparison of the various methods in terms of path diversity, entropy, redundancy and path length (travel time). For those experiments we computed k=100 paths each for 25 random sets of source and target vertices (cf. Figure 4). Figure 4(a) shows the *diversity* results (logarithmic scale). Diversity is low for the original k-shortest path algorithm. Both randomization methods GR and PR generate comparable results, which are close to one order of magnitude larger than those of the original method. The penalization methods generate the most diverse results. Even small penalization factors (p=0.01) provide for an increase in diversity by 82.2% on average, compared to the randomization approaches. Larger penalization factors result in even more significant gains. Finally, the $p \to \infty$ achieves 99.86% more diversity than the randomization approaches.

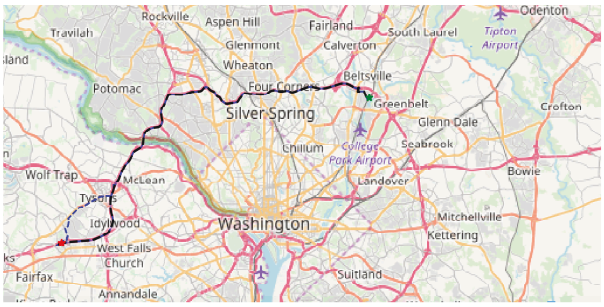
Figure 4(b) shows the results for measured entropy. The entropy of a set of paths is another indicator of path diversity. The results are similar to diversity, but have some interesting differences.



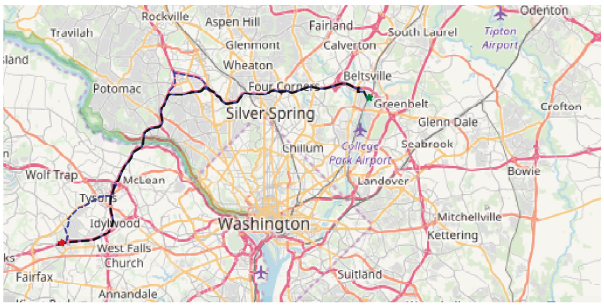
(a) Shortest path by Dijkstra's algorithm



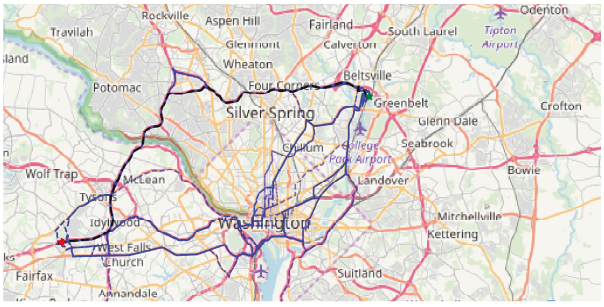
(b) 100-Shortest paths by Yen's algorithm



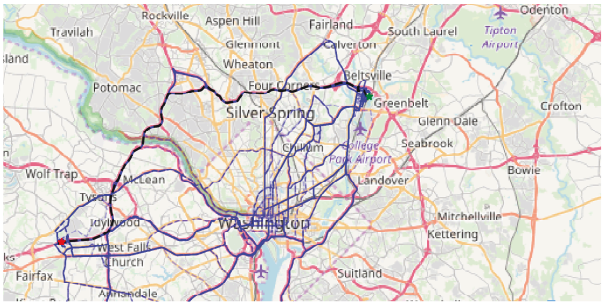
(c) 100 paths by total graph randomization ($\delta = 0.1$)



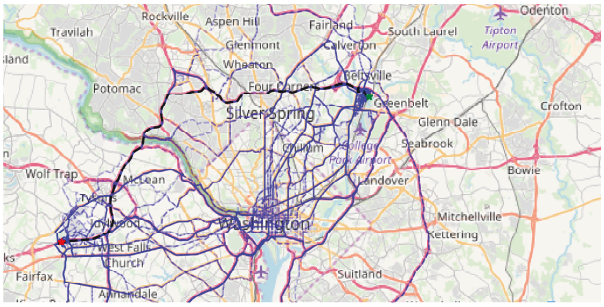
(d) 100 paths by previous-path randomization ($\delta = 0.1$)



(e) 100 paths by penalization ($p = 0.01$)



(f) 100 paths by penalization ($p = 0.03$)



(g) 100 paths by penalization ($p = 0.1$)

(h) 100 paths by penalization ($p \to \infty$)

Figure 2: Diversity of 100 shortest paths computed between a fixed source-destination pair using different approaches.

The two randomization approaches achieve marginally higher entropy compared to the original method. The penalization methods manage to increase the entropy by at least 75% even with a small penalization factor of 1%. Again, larger factors result in higher entropy. The $p \to \infty$ approach achieves an entropy comparable to the p=0.3 penalization in some cases. The reason here is that after it plateaus and has visited most edges, it will always return the path of the lowest number of edges, as shown in the example of Figure 3. This causes the edges of this "last" frequently visited path to have a significantly higher load and decreases the overall entropy.

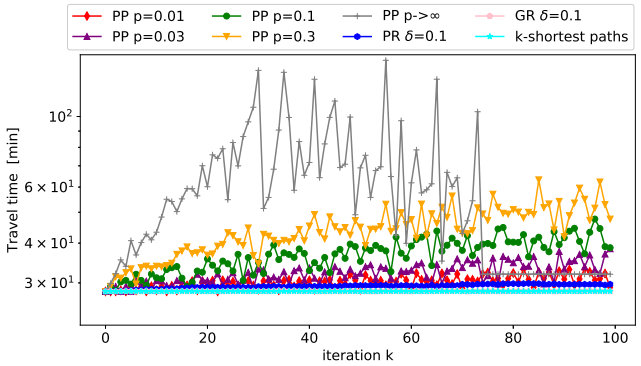


Figure 3: Travel time of each of the k=100 paths

The *redundancy* measurements are shown in Figure 4(c) in a scale from 1 to 100. We observe a trend similar to the inverse of diversity as the more diverse a set of paths is, the lower is its redundancy. The original 100-shortest paths are highly redundant, i.e., close to 100. Randomization methods GR and PR achieve a redundancy of 35 and 39, respectively. All penalization methods generate path sets of $\mathcal{R}$=12 and below. The $p \to \infty$ approach has the lowest redundancy for all the source-target pairs, ranging from 1.1% to 1.5%.

Figure 4(d) shows the average *path length* (travel time). The randomization methods produce paths with lengths close to that of the original method (withing 2%). The penalization methods for small p values, generates paths that have high diversity and relatively small paths lengths. *For p=0.01 the average path length is increased by only 5.7% (compared to the original method), while the result is 98% more diverse.* The extreme case of penalization $p \to \infty$ achieves high diversity at the cost of a very long paths.

5.5.3 Path Length versus Diversity. The trade-off between travel time and path diversity is illustrated in Figure 5. This figure relates the quality measures to resulting path length. Overall, we are trying to find a set of "shortest" paths that is diverse and close to the length of the actual shortest path.

Figure 5(a) plots *diversity with respect to path length*. We observe two extremes, (i) the original k-shortest path method produces the shortest but not at all diverse paths and (ii) the $p \to \infty$ penalization method produces the most diverse but also longest paths. Randomization methods perform close to the original method. However, if we want to find the solution with the best trade-off between diversity and length, we have to identify the set of points that belong to the skyline [24] of high diversity and low travel time. This would include all points that are not dominated by any other point in both dimensions. Since PR is worse than GR in terms of both diversity and travel time, PR is *dominated* by GR. The penalization methods PP are not dominated by any other approach. Different values of p achieve a different trade-off between diversity and length. As p increases we gain in terms of higher diversity but lose in terms of a greater path length. This relationship appears to be linear for the penalization based approach, as shown in Figure 5(a).

Figure 5(b) shows the corresponding relationship between *entropy and path length*. It is shown that none of the tested solutions is dominated by any other approach. Increasing the p results in longer paths, but also in increased entropy, which can be translated as a better distribution of paths in the road network (load balance).

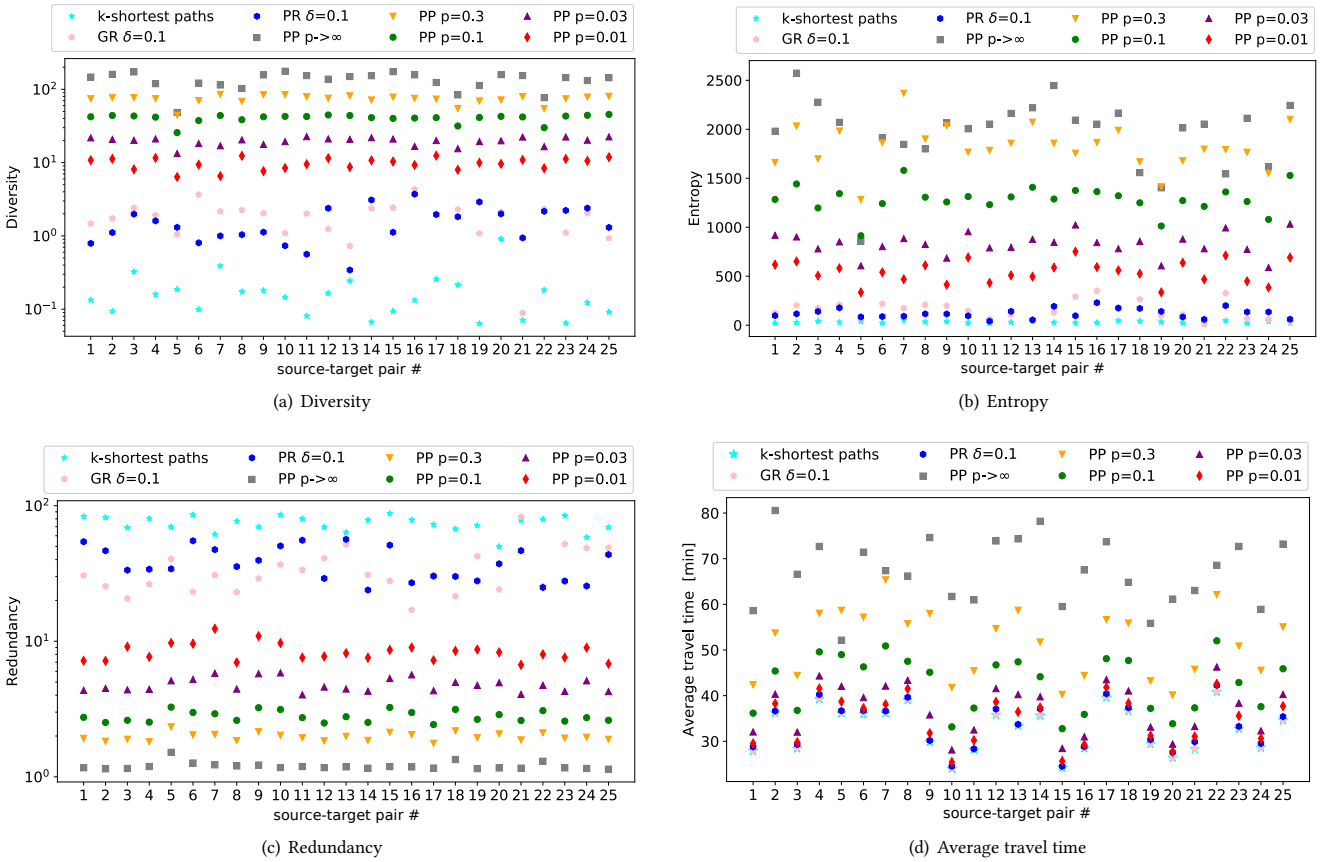
The relationship between average *path length and redundancy* is presented in Figure 5(c). Looking at this graph as a more traditional skyline, we want to find the points that have low values for both these quality metrics. All PP results are part of the skyline, as they are not dominated in both dimensions by any other point. *The approach that appears to substantially lower the redundancy of its result, while keeping a reasonable average travel time is* PP *for penalization factors p of 0.01 and/to 0.03.*

5.5.4 Scalability. Figure 6 shows the behavior of all approaches for different values of k. The first three graphs report the average values of our quality metrics for 25 source-target pairs. Figure 6(a) shows how the average diversity increases with k. While, the original method is constant, the two randomization approaches are slow growing, so do the penalization methods grow almost linearly with k. The $p \to \infty$ methods plateaus at k=90 due to the saturation effect of having visited all edges. Note that, by definition the diversity cannot decrease when adding more paths to the result set. If we keep computing the same path from some iteration and on, the diversity will remain stable, which explains the behavior of the $p \to \infty$ curve.

Figure 6(b) shows a sublinear Entropy growth. All methods are outperformed by the penalization variants PP by at least two orders of magnitude. The entropy of PP $p \to \infty$ increases rapidly and then decreases after k=90 due to the above mention saturation effect and eventually always visiting the path with the fewest edges.

Figure 6(c) shows the redundancy behavior. Note that the y-axis uses a logarithmic scale. All methods shows similar rates of growth. We clearly observe the k-shortest path approach having the largest increase in redundancy, followed by the curves of PR and GR. The redundancy of these approaches is approximately linear. The penalization methods outperform all others and manage to maintain low values of redundancy $\mathcal{R}$<10, for any k. Again, the $p \to \infty$ approach exhibits interesting behavior. Initially, it shows almost no redundancy, as it avoid picking the same edge more than once unless the graph topology forces it to pick a duplicate. Once this approach converges into the shortest number of hops, the redundancy remains low, as most of the graph has already been explored in the first iterations.

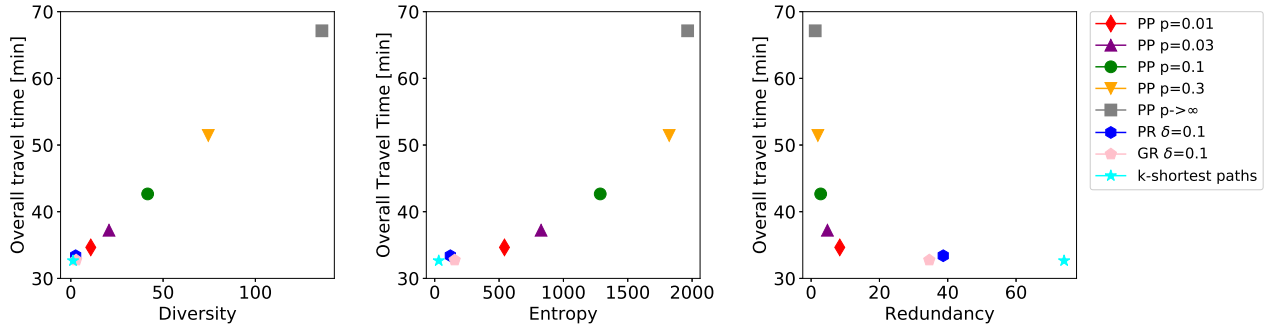
Finally, Figure 6(d) shows the path length (travel time) for every iteration up to 500 paths, averaged over the 25 random source-target pairs. Note that in this last graph, every measurement corresponds to the k^{th} path, not the set of the first k paths that were used for the previous three graphs. The path length of the 500-shortest paths remains almost stable for all iterations, showing that these 500 paths do not differ substantially from each other. The path lengths of the randomization approaches remain low and relatively close to the original shortest paths, without experiencing significant variance. An interesting observation is the path randomization approach progressively yielding longer paths. The reason is that the most commonly chosen paths chosen after randomization have progressively more random noise added. While this noise is unbiased, it increases the variance of the weight of the randomized edges. Consequently, some edges may get very large weights, but extremely low weights are impossible due to the constraint of having positive weights. This adds a slight bias towards making the average edge larger, thus creating a trend of preferring new edges. We also see



(a) Diversity (b) Entropy

(c) Redundancy (d) Average travel time

Figure 4: Diversity, Entropy, Redundancy and Path length for 25 random source-target vertex pairs.



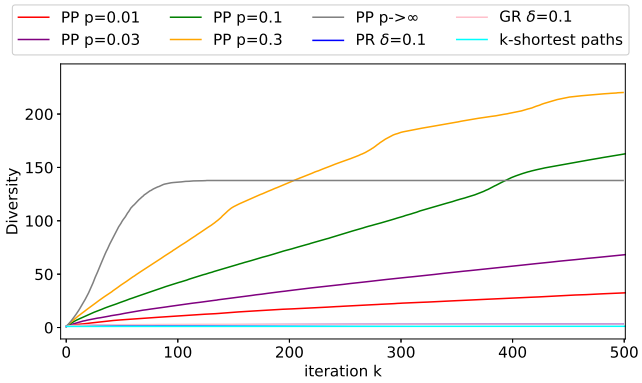
(a) Path length (travel time) vs. Diversity (b) Path length (travel time) vs. Entropy (c) Path length (travel time) vs. Redundancy

Figure 5: Average path length (travel time) vs. Diversity, Entropy and Redundancy, for k=100 shortest paths.

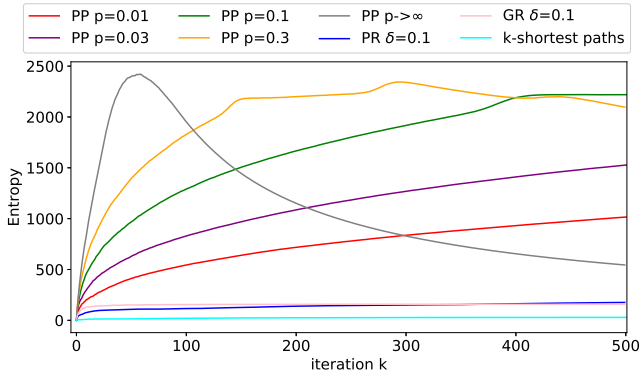
that, for all approaches, the travel times is strongly correlated to the entropy. This result is intuitive, as longer paths are more likely to explore new areas of the network, thus increasing the entropy of the distribution of visited edges.

6 CONCLUSIONS

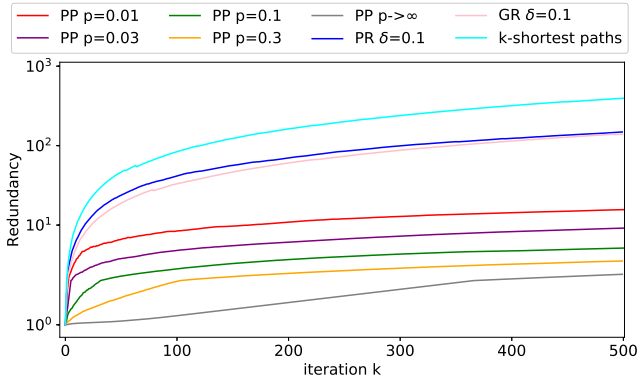
This work addresses the meaningful diversification of possible paths from a source to a target location. Solving this problem has applications for commuters, who want to reduce the monotony of their daily travel patterns, for fleet navigation, where the task is to move a large number of vehicles through the network, and load balancing, which tries to reduce the traffic of bottlenecks in the road network. First, we concluded that methods that compute (i) k-shortest paths and (ii) the k disjoint shortest paths are too strict, either offering too little diversity in practice, or involving unreasonable detours. To identify a compromise, we test variations of the penalty method, an approach that iteratively penalizes edges that have been used in previous paths. Our approach allows one to balance the trade-off between path length (travel time) and diversity using a single parameter p. We show theoretically that this approach is guaranteed to find new paths. In our experimental evaluation, we first show that different choices of p affect the network space explored by



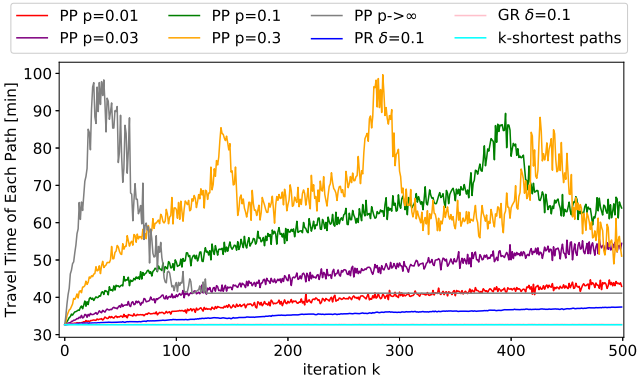
(a) Diversity vs. k



(b) Entropy vs. k



(c) Redundancy vs. k



(d) Path length (travel time) of each path

Figure 6: Diversity, Entropy, Redundancy and Path length (travel time) for varying k.

our paths, while existing solutions fail to achieve reasonable diversity. In our qualitative evaluation, we show that this trend prevails for randomly chosen source and target locations, while incurring reasonable run-times.

REFERENCES

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Alternative routes in road networks. In *International Symposium on Experimental Algorithms*, pages 23–34. Springer, 2010.

[2] V. Akgün, E. Erkut, and R. Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121(2):232–246, 2000.

[3] H. Aljazzar and S. Leue. K*: A heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence*, 175(18):2129–2154, 2011.

[4] R. Bader, J. Dees, R. Geisberger, and P. Sanders. Alternative route graphs in road networks. In *International Conference on Theory and Practice of Algorithms in (Computer) Systems*, pages 21–32. Springer, 2011.

[5] M. E. Baran and F. F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power delivery*, 4(2):1401–1407, 1989.

[6] R. Bhandari. *Survivable networks: algorithms for diverse routing*. Springer Science & Business Media, 1999.

[7] T. Chondrogiannis, P. Bouros, J. Gamper, and U. Leser. Alternative routing: k-shortest paths with limited overlap. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 68. ACM, 2015.

[8] T. Chondrogiannis, P. Bouros, J. Gamper, and U. Leser. Exact and approximate algorithms for finding k-shortest paths with limited overlap. In *20th International Conference on Extending Database Technology: EDBT 2017*, pages 414–425, 2017.

[9] G. Cookson. Inrix global traffic scorecard (2017). INRIX Research, February, (published February 2018).

[10] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning. In *International Symposium on Experimental Algorithms*, pages 376–387. Springer, 2011.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[12] D. Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[13] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, October-December:12–18, 2008.

[14] M. Kashem, V. Ganapathy, and G. Jasmon. Network reconfiguration for load balancing in distribution networks. *IEE Proceedings-Generation, Transmission and Distribution*, 146(6):563–567, 1999.

[15] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12(4):411–427, 1982.

[16] M. Kobitzsch. An alternative approach to alternative routes: Hidar. In *European Symposium on Algorithms*, pages 613–624. Springer, 2013.

[17] M. Kobitzsch, M. Radermacher, and D. Schieferdecker. Evolution and evaluation of the penalty method for alternative graphs. 2013.

[18] H. Liu, C. Jin, B. Yang, and A. Zhou. Finding top-k shortest paths with diversity. *IEEE Transactions on Knowledge and Data Engineering*, 30(3):488–502, 2017.

[19] D. Luxen and D. Schieferdecker. Candidate sets for alternative routes in road networks. In *International Symposium on Experimental Algorithms*, pages 260–270. Springer, 2012.

[20] J. P. Rohrer, A. Jabbar, and J. P. Sterbenz. Path diversification: A multipath resilience mechanism. In *Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop on*, pages 343–351. IEEE, 2009.

[21] D. Schrank, B. Eisele, T. Lomax, and J. Bak. Urban Mobility Scorecard. *The Texas A&M Transportation Institute and INRIX*, 2015.

[22] J. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974.

[23] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[24] E. Tiakas, A. N. Papadopoulos, and Y. Manolopoulos. Skyline queries: An introduction. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6, July 2015.

[25] J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.