Deep Learning-Based Alternative Route Computation

Alex Zhai Google
Dee Guo Google
Kostas Kollias Google
Sreenivas Gollapudi Google
Daniel Delling Google

Abstract

Algorithms for the computation of alternative routes in road networks power many geographic navigation systems. A good set of alternative routes offers meaningful options to the user of the system and can support applications such as routing that is robust to failures (e.g., road closures, extreme traffic congestion, etc.) and routing with diverse preferences and objective functions. Algorithmic techniques for alternative route computation include the penalty method, via-node type algorithms (which deploy bidirectional search and finding plateaus), and, more recently, electrical-circuit based algorithms. In this work we focus on the practically important family of via-node type algorithms and aim to produce high quality alternative routes for road networks using a novel deep learning-based approach that learns a representation of the underlying road network.

We show that this approach can support natural objectives, such as the uniformly bounded stretch, that are difficult and computationally expensive to support through traditional algorithmic techniques. Moreover, we achieve this in a practical system based on the Customizable Route Planning (CRP) hierarchical routing architecture. Our training methodology uses the hierarchical partition of the graph and trains a model to predict which boundary nodes in the partition should be crossed by the alternative routes. We describe our methods in detail and evaluate them against previously studied baselines, showing quality improvements in the road networks of Seattle, Paris, and Bangalore.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION

Computing good routes between two points in a road network is a fundamental problem at the core of popular navigation platforms used by hundreds of millions of people every day. To a first approximation, users are interested in finding the route that minimizes travel distance (or time), in which case this problem reduces to the classical shortest path problem in graph theory.

In the real world, however, the "goodness" of a route depends on many additional factors. Whether to accommodate a user's driving preferences, or to support a combined objective function (e.g. trade off between travel time and carbon emissions), or to be robust against changes in road conditions (such as a road closure), real-world navigation platforms often produce a larger set of candidate routes than a single "best" one.

The problem of finding these candidate routes is known as *alternative (or alternate) route computation* and has received significant attention in the literature (e.g., de la Barra and Anez [1993], Bader et al. [2011], Paraskevopoulos and Zaroliagis [2013], Cam [2009], Luxen and Schieferdecker [2012], Abraham et al. [2013], Kobitzsch [2015], Sinop et al. [2021]). Existing approaches typically model alternate route computation as a graph optimization problem that accounts for both diversity (the alternates are not too similar to each other) and quality (e.g. each alternate route is still reasonably fast). However, computing exact optimal solutions for these problems is intractable for general graphs, so heuristics are always required in practice.

One important measure of a route's quality is what we term the *uniform stretch*[1], defined as the maximum factor by which a subpath of an alternate route is longer than the corresponding shortest path (see Section 2.2.1 for more details). In other words, requiring low uniform stretch means that if we suggest the user take a detour from the shortest path, it should not be too significant of a detour (see e.g. Figure 1). Although uniform stretch captures well our intuitions about what makes a good alternate route, it is considered very expensive to

[1]Based on the "uniformly bounded stretch" introduced in Abraham et al. [2013]

compute (let alone optimize) in traditional algorithms (Abraham et al. [2013]).

1.1 Our Contributions

In this work, we propose a new approach based on *learning* road network representations using deep learning models. Instead of crafting and tuning heuristic algorithms, we directly learn a succinct representation of the road network that implicitly encodes multiple high-quality routes between every pair of nodes. The alternate routes we generate are via-paths: shortest paths from source to destination that go through a particular other node (called a via-node). Given a road network, we train a model that predicts good via-nodes for any source/destination pair.

Our method is able to efficiently produce alternate routes with low uniform stretch. By doing the heavy lifting at model training time, we sidestep the algorithmic complexities of computing uniform stretch at query time. Additionally, we use the Customizable Route Planning (Delling et al. [2017]) routing architecture, which enables fast searches using precomputed shortcuts. As a result, our method is practical for real-time use cases.

We evaluate our method on real road networks for the metropolitan areas of Seattle, Paris, and Bangalore. The primary metric we consider is *non-cuspy diversity* (defined formally in Section 2.2.2), which measures the diversity of alternate routes returned but excludes routes with high uniform stretch (such "cuspy" routes would be unlikely to satisfy the user). Existing via-node algorithms that address cuspiness are based on local optimality heuristics, which consider whether subpaths of the via-path are shortest paths (Abraham et al. [2013], Kobitzsch [2013]). We compare against two such algorithms as baselines, the standard plateau method and the T-test introduced in Abraham et al. [2013] (explained in Section 4.3). Our experiments show that the model-based approach outperforms both baselines.

Finally, we note that our general framework has two important advantages over traditional algorithmic approaches:

- *Flexible and generic.* Rather than come up with new algorithmic ideas for every new use case, deep learning models are able to optimize for a wide range of objectives using the same framework.
- *Adapts to the data.* Rather than apply the same rules to every part of the road network, a learned representation can naturally encode routing patterns specific to certain sub-regions.

Thus, we believe this framework has the potential to be applied more broadly to other routing problems, where instead of uniform stretch, we can optimize for almost any other imaginable objective (e.g. simplicity of directions, user preferences, etc.).

1.2 Related Work

1.2.1 Alternative Routes Generation

Early approaches for alternative routes generation were based on using different objective functions, such as best travel-time, minimum distance, road quality, scenery and so on (Ben-Akiva et al. [1984]). However, this approach is not scalable, as even for a small set of features, there will be exponentially many objective functions. Moreover, many of the potential criteria are not independent of each other.

A second approach is the k-shortest path problem (Yen [1971]). This approach has been described as less effective (Bader et al. [2011]) for alternative route generation in road networks due to similarity of the produced routes. A number of recent works have attempted to correct this issue by incorporating explicit diversity metrics into their problem formulations and algorithms, with a focus on efficiently computing and optimizing overlap between candidate paths (Häcker et al. [2021], Chondrogiannis et al. [2020], Liu et al. [2018], Luo et al. [2022]).

The penalty method is another approach based on repeated Dijkstra searches that penalize edges already used in previous searches. It was first introduced by de la Barra and Anez [1993], with various subsequent improvements (Bader et al. [2011], Paraskevopoulos and Zaroliagis [2013]).

Via-node methods (Cam [2009], Luxen and Schieferdecker [2012], Abraham et al. [2013]), particularly the plateau method, are the most popular approaches for alternative route generation. The goal is to find a single intermediate node (the via-node) and take the shortest path from source to destination that goes through the via-node. The key question then becomes the identification of good via-nodes. In the plateau method, via-nodes are chosen according to high overlap between the forward and reverse search trees in a bi-directional Dijkstra search. For a more in-depth description of both plateau and penalty methods, we refer the reader to Kobitzsch [2015].

Recently, Sinop et al. [2021] proposed the use of electrical flow networks for generating alternative routes. Their algorithm models the road network as a network of resistors and lets a unit of current travel from the origin to the destination. Flow decomposition is then used to produce a set of paths.

In most of the above literature, the only measure of an alternate path's quality is its length. Abraham et al. [2013] define a more comprehensive set of criteria for determining the quality of an alternative route, including the uniform stretch that we focus on in this work. As mentioned earlier, existing algorithms that address uniform stretch (Abraham et al. [2013], Kobitzsch [2013]) are based on local optimality heuristics such as the plateau method or a local search. Note however that they are geared toward finding a single (or very few) alternates.

Finally, we mention the work of Li et al. [2020], which studies the incremental update of the alternative routes as the user is in motion.

1.2.2 Neural Net-Based Models

Neural net-based models have been applied in the context of road networks for various applications, most prominent of which is predicting the average speed of road segments. These works are often based on graph neural networks, which account for nearby road segments when predicting traffic speed. We refer the reader to Jiang and Luo [2021] for a comprehensive overview.

Along similar lines, deep learning techniques have also been used for diverting traffic after events. For example, Saha et al. [2020] use deep learning to reroute traffic after accidents on freeways, and Perez-Murueta et al. [2019] use it to divert traffic from congested zones. The work in Raj et al. [2020] designs a deep learning-based traffic advisory system. For further pointers in this area, we point the reader to the review in Patil [2022].

The above literature focuses on using deep learning to process signals such as traffic data, but not for the actual generation of routes. Some works that use deep learning models in a route planning context include Kool et al. [2019], Ma et al. [2021], Vinyals et al. [2015], Bello et al. [2016], Khalil et al. [2017], Kaempfer and Wolf [2018], Deudon et al. [2018], Nazari et al. [2018]. However, in all these works the focus is really on solving combinatorial optimization problems (e.g. traveling salesperson and multi-vehicle routing problems), which are abstracted away from the geometry of the road network.

In this paper, we use neural networks that directly encode structural information about an entire metro-sized road network, with an application to generating alternative routes. We are not aware of any previous work that is directly comparable to ours in the machine learning literature.

2 PRELIMINARIES

In this section we review how the CRP architecture works, establish notation, and define our metrics of interest.

2.1 CRP Preprocessing

We model a road network as a graph with nodes V and weighted edges E, where weights represent the cost (typically distance or travel time) to traverse each edge. Additionally, we preprocess the graph following the Customizable Route Planning (Delling et al. [2017], henceforth "CRP") routing architecture, which hierarchically partitions the graph into pieces that share few boundary nodes in order to support very efficient searches. Below, we give a brief review of how CRP works and establish notation for our setting.

2.1.1 Hierarchical Clustering

Given a graph, the first step of CRP is to partition the nodes V into a small number of disjoint subsets $P_1, \ldots, P_k \subset V$ such that $V = \bigcup_{i=1}^{k} P_i$. Let B be the set of "boundary" nodes, i.e. nodes incident to an edge that crosses between different P_i's. The goal is to have P_i roughly equally sized and for B to be as small as possible. Intuitively, we want the P_i to correspond to major geographical "regions" of the graph, and B to correspond to major highways and bridges that you must go through to get from one region to another. This is usually done using min-cut algorithms.

Then, for some predetermined number of levels L, the process is repeated recursively for each P_i up to a depth L. The final result is L levels of partitionings, where for each level i, we have a partitioning $\mathcal{P}_i = \{P_{i,1}, P_{i,2}, \ldots\}$ of V, and $\mathcal{P}_i$ is a refinement of $\mathcal{P}_{i+1}$. We also define $B_i \subset V$ to be the set of boundary nodes for $\mathcal{P}_i$.

For a cluster $P \in \mathcal{P}_i$, we write $B(P) := P \cap B_i$ for its set of boundary nodes, and for a node $v \in V$, we write $P_i(v)$ for its partition at level i.

2.1.2 Shortcut Edges and Routing

The second step of CRP is to precompute shortcut costs for each partition P (at each level) between each pair of nodes in $B(P)$. This enables efficient shortest path searches as follows: given a source and target pair $s, t \in V$, we can perform an ordinary Dijkstra search, except that we always take the highest level of precomputed shortcut that is "permissible".

To be precise, a level-i shortcut from a boundary node $v \in B_i$ (i.e. a shortcut from v to another node in $B(P_i(v))$) is *permissible* if $P_i(v) \neq P_i(s)$ and $P_i(v) \neq P_i(t)$. Intuitively, this is saying that from the point of

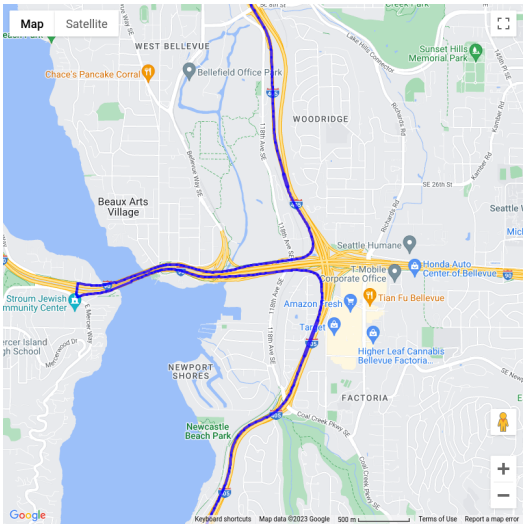


Figure 1: An example of a cuspy path. The path starts on a highway and turns onto another highway, only to return to the original highway.

view of search from s to t, entering and exiting $P_i(v)$ can be treated as a "black box", and we don't need to inspect any internal nodes of $P_i(v)$.

Thus, a CRP-based search from s to t will generally use higher and higher level shortcuts as it leaves s and then lower and lower level shortcuts as it approaches t. In what follows (unless specified otherwise), whenever we refer to a graph search, it is assumed that we are doing a search based on CRP shortcuts as described above.

2.2 Evaluation Metrics

2.2.1 Uniform Stretch

The *stretch* of a path is defined as the ratio of its cost divided by the optimal cost. Following the notion of "uniformly bounded stretch" introduced in Abraham et al. [2013], we define the *uniform stretch* of a path as the maximum stretch of any of its subpaths. The uniform stretch is our primary measure of the quality of an alternate route—our objective is to find diverse alternate routes with low uniform stretch. For example, if we suggest to a user a path that has uniform stretch greater than 10, then there is some subpath where we are suggesting the user to take 10 times as long as they normally would, which is unlikely to be desirable. In particular, this measure helps to rule out low-stretch but "cuspy" paths that can often be generated by naive via-node algorithms (see for example Figure 1).

As observed in Abraham et al. [2013], it seems difficult to efficiently compute the exact uniform stretch, and they propose a "T-test" approximation approach based on searching in the neighborhood of a via-node. We evaluate this approach in Section 4, but our focus is on offline computation of uniform stretch that we use to train a model (see Section 3.2.4).

Although efficiency is no longer a primary concern for offline computation, we note that there is a way to estimate uniform stretch that is much faster than a naive approach of checking each subpath. Given a path P and a threshold u, we can run a Dijkstra search with all of the weights of edges in P multiplied by $\frac{1}{u}$. Then, the uniform stretch of P is below u if and only if the resulting optimal path is our original path P. We can then run this repeatedly for different values of u (perhaps in a binary search) to get an estimate of the uniform stretch. Note however that this is still not nearly fast enough to run on more than a handful of paths at query time.

2.2.2 Non-cuspy Diversity

To measure the overall quality of a set of alternates, we measure the diversity of paths while not counting "bad" paths. The *diversity* of a set of alternates is defined as the sum of costs of distinct edges appearing in the paths divided by the optimal path cost. For a given uniform stretch threshold u, we define the *non-cuspy diversity* to be the diversity excluding paths that have uniform stretch above u. In most places we use $u = 2.0$.

We then measure the overall quality by computing the *non-cuspy diversity* of the top n alternates produced by the different algorithms, with varying values of n.

3 OUR APPROACH

3.1 CRP-based Via-node Alternates

We first describe the simple algorithmic component of our alternate route generation procedure. It follows the template of many via-node algorithms, consisting of three main steps:

1. **Candidate generation.** First, we perform a forward CRP search from the origin and a reverse CRP search from the destination, keeping track of the resulting shortest path trees. All meeting points of the two searches are initially considered as candidate via-nodes (the via-path through each candidate via-node can be easily reconstructed from the search trees). We prune out the candidates whose via-paths have stretch above 1.5.

2. **Scoring and ranking.** We then rank the quality of candidate via-nodes according to some scoring mechanism. In our case, the quality is determined by a model prediction (see Section 3.2 below).

In particular, the model predicts whether each via-node has uniform stretch above some specified threshold. If yes, the via-node is scored as negative infinity and effectively removed from consideration. Remaining candidate via-nodes are ranked in increasing order of their path length.

We also compare against several other scoring procedures as baselines.

3. **Selection.** Finally, we select a set of the requested size among the ranked/scored candidates. We used a simple greedy procedure: we took the paths one by one from highest quality to lowest, but omitted any that had too high "overlap" (above some threshold θ) with the paths selected so far, progressively increasing θ if not enough paths could be found.

   Here, "overlap" of two paths p_1 and p_2 is defined to be the total length of edges in both paths divided by the smaller of the lengths of p_1 and p_2. We chose (somewhat arbitrarily) thresholds of $\theta = 0.75, 0.85, 0.95$, which produced reasonable results.

The overall procedure is summarized in pseudocode below.

Algorithm 1 CRP-based Alternates Generation

Input: Origin s, destination t, number of alternates n, via-node quality scoring function $Q : V \to \mathbb{R}$
Output: List A of paths from s to t representing the selected alternates

```
function GET-ALTERNATES(s, t, n, Q)
    ▷ Gather all candidate via-nodes from bidirectional search ◁
    C_f ← nodes in forward CRP search from s
    C_r ← nodes in reverse CRP search from t
    C ← C_f ∩ C_r
    ▷ Filter high stretch candidates and sort according to quality (best-to-worst) ◁
    C ← {x : x ∈ C, stretch(viapath(x)) ≤ 1.5}
    SORT-DESCENDING(C, Q)
    ▷ Build up the selected set of alternates ◁
    A ← ∅
    for θ = 0.75, 0.85, 0.95 do
        for x in C do
            p ← viapath(x)
            if overlap(p, p') ≤ θ, ∀p' ∈ A then
                A ← A ∪ {p}
            if |A| ≥ n then
                return A
    return A
```

3.2 Via-node Prediction Model

We now describe in detail the model used to filter out via-paths with high uniform stretch. Although our model is technically a single neural network, it consists of two conceptually distinct parts: an embedding representation of the road network and a prediction module, detailed below.

3.2.1 Embedding Representation

A key property of real-world road networks is that desirable routes tend to all go through the same small set of major roads. We account for this phenomenon by building our model's representation of the road network on top of the CRP partitioning described in Section 2.1. Recall that we have a hierarchy of L levels, with the i-th level consisting of partitions $\mathcal{P}_i$ and boundary nodes B_i. We then learn a vector embedding

$$e : V \sqcup \left(\bigsqcup \mathcal{P}_i\right) \sqcup \left(\bigsqcup B_i\right) \to \mathbb{R}^d,$$

which assigns a d-dimensional vector to each node, partition, and boundary node in the CRP graph.[2] By having separate embeddings at each CRP level, we can encode graph information at different spatial resolutions.

3.2.2 Prediction Module

We train a prediction module that takes as input a triple $(s, t, v) \in V \times V \times V$, where s and t are origin and destination nodes, and v is a via node. The module then outputs a prediction of whether v is a good via-node for going from s to t.

The prediction module consists of two components. The first is a query encoding module $q : \mathbb{R}^{d \times (2L+2)} \to \mathbb{R}^m$ that encodes the query (s, t) into an m-dimensional vector. Its input is the embedding vectors of s and t concatenated with the embeddings of their CRP partitions at each level, i.e.

$$\begin{aligned} q(s,t) := q(&e(s), e(P_1(s)), \ldots, e(P_L(s)),\\ &e(t), e(P_1(t)), \ldots, e(P_L(t))), \end{aligned}$$

where in a slight abuse of notation we think of q also as a function of s and t.

The second component is a classification head $c : \mathbb{R}^{m+d} \to \mathbb{R}$ that takes in the query encoding and via-node embedding and outputs a probability that the via-node is "good". The end-to-end calculation is thus

$$\hat{\mathbb{P}}(\text{good via}) = c(q(s,t), e(v)).$$

[2] Note the use of disjoint union. The same node may appear in multiple B_i, but it is assigned separate embeddings per level.

Our approach allows flexibility in defining what is considered "good", but for our experiments, a via-node was considered good if its via-path had uniform stretch at most 2.

3.2.3 Architecture Details

We use $L = 6$ and $d = m = 64$. For the embedding e and query encoder q, we leverage BERT (Devlin et al. [2019]), a transformer architecture that is widely used in natural language processing. Customarily, it takes as input a sequence of text tokens, where each token has a learned embedding. Instead of text tokens, we use graph nodes and partitions with a fixed sequence length of $2L + 2$. Our version of BERT uses 4 layers, 8 attention heads, and intermediate dimension 256. BERT produces encoded outputs of the same dimension as the embeddings at each input position; we read off the final output of q from the first position.

For the classification head c, we simply use a two-layer multilayer perceptron with hidden dimension 128 and output dimension 2, followed by softmax binary classification.

3.2.4 Training Procedure

We randomly sampled 10 million origin/destination pairs and computed a CRP bidirectional search for each. We then used the via-node candidates found by the bidirectional search as training examples, using a path search as described in Section 2.2 to determine whether the resulting via-paths had uniform stretch above our threshold of 2. The model was trained for approximately 50 epochs until convergence.

3.2.5 Resource Usage

Even though the training stage is a resource-intensive process and the memory footprint of the model is significant, both are well within the realm of practicality. Specifically, in our experiments, the training time was in the order of tens of hours, and the model used 1.2 GB of memory for a 1.5 million node network.

4 EXPERIMENTS

We evaluated on data from a popular navigation engine in 3 metro areas: Seattle, Paris, and Bangalore. Each road network consisted of between 1.5M and 2M nodes (see Table 1 for detailed CRP statistics).

We sampled a query set of 100 random origin/destination pairs for each metro and evaluated the alternates produced by each of four via-node ranking methods:

- The standard plateau heuristic where the size of a route's plateau (cost of the part of the route that belongs to both the forward and backward trees) is compared with the total route cost.
- The T-test heuristic (Abraham et al. [2013]) where routes that have a locally suboptimal stretch around the via node are rejected.
- A high quality (but slow and impractical) benchmark where alternates are thoroughly evaluated in terms of their uniform stretch before being returned.
- The deep learning-based scoring approach that we propose in this work.

We produce n alternate routes ($n = 5, 10, 15, 20, 25, 30$) from each ranking method using the algorithm described in Section 3.1. We used the same parameters for each method, with the only difference being the scoring function Q in Algorithm 1. The alternates were then evaluated according to the noncuspy diversity metric described in Section 2.2.2 with uniform stretch threshold $u = 2.0$.

4.1 Ranking Procedures

We now explain in more detail how each one of the algorithms that we evaluate handles the ranking stage of the search algorithm.

4.2 Plateau Baseline

The *plateau* of a via-path is the part of the path that is included in both the forward and the backward shortest path trees (e.g., in all routes the via node is by definition in the plateau, and in the shortest path, the whole route is the plateau). To find the plateau, one can begin at the via-node and start moving forward/backward in the path and test whether each hop is included in both trees produced by the bidirectional search. Note that since we use CRP-based searches, in our setting these hops are CRP shortcuts. Thus, the plateau sizes we find are lower approximations of the true plateau size.

Once the plateau sizes are calculated, the alternates are ranked based on the difference between the cost of the alternate and the size of the plateau, lower values being considered better (e.g., for the shortest path the value is 0 since the whole shortest path belongs to both trees of the bidirectional search and the plateau equals the complete route).

4.3 T-test Baseline

In this algorithm, we filter out some bad candidates by applying the "T-test" introduced in Abraham et al.

	Seattle		Paris		Bangalore	
	# partitions	# boundary nodes	# partitions	# boundary nodes	# partitions	# boundary nodes
Level 0		1647889		1635219		1965248
Level 1	4183	34176	4616	43871	4839	41441
Level 2	1148	11695	1247	16662	1277	14777
Level 3	161	2371	167	4562	160	3487
Level 4	34	753	37	1799	41	1497
Level 5	8	188	9	772	10	545
Level 6	3	80	2	183	2	153

Table 1: Number of CRP partitions and boundary nodes at each level for the graphs we used.

[2013]. Briefly, for a given parameter $T \in [0, 1]$, the *T-test* is performed on a via-node v as follows: take the via-path P corresponding to v and take the subpath of P centered around v of cost approximately $2T$ times the cost of P. The test is passed if this subpath is a shortest path. (We refer to Abraham et al. [2013] for precise details.) After removing all candidates that do not pass the T-test, we rank the remaining via-nodes by the same scoring as the plateau method.[3]

4.4 Uniform Stretch Filter Benchmark

In this algorithm, we perform an expensive exact computation on each candidate route to determine if its uniform stretch exceeds the specified threshold. We then exclude any path with uniform stretch exceeding the threshold and rank the rest by ascending total cost. Thus, this algorithm is guaranteed never to return a cuspy route.

Due to the expensive computation on every single candidate via-node, this algorithm is not practical. However, we include it to show the headroom for other methods.

4.5 Model Prediction

In this method, we use the model described in Section 3.2 to predict for each via-node in the candidate set whether its via-path has uniform stretch exceeding the threshold u. We then exclude any via-node predicted to exceed the uniform stretch threshold and rank the rest by ascending cost, as above.

4.6 Results

Figure 2 shows the non-cuspy diversity (cuspiness threshold $u = 2.0$) averaged over 100 query samples in each of the 3 metro areas. We find that the model prediction method outperforms both the plateau and T-

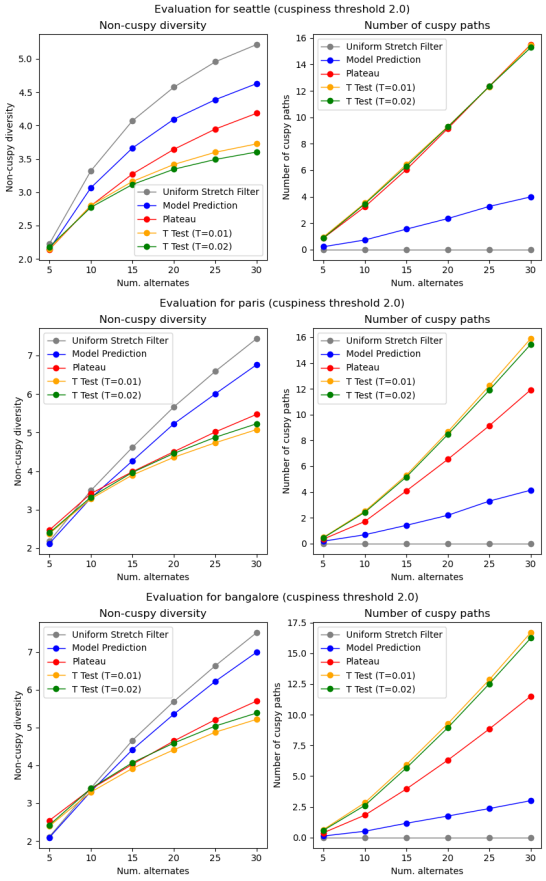


Figure 2: Diversity and number of cuspy paths for Seattle, Paris, and Bangalore metros, with uniform stretch threshold 2.0.

[3]We note that Abraham et al. [2013] recommends a variant of the plateau ranking function that is more suited towards finding a single alternate route. We experimented with several variants and did not observe significant differences, so we just use the plateau ranking function for simplicity.

Ranking method	Scoring time (ms)	Path expansion time (ms)	Total time (ms)
Uniform Stretch Filter	34186.10	169.35	34378.31
Model Prediction	6.15	179.70	208.19
Plateau	0.50	188.23	214.61
T-test ($T = 0.01$)	738.18	194.67	958.43
T-test ($T = 0.02$)	816.35	191.97	1033.24

Table 2: Average latencies across three metros for generating 30 alternates.

test baselines ($T = 0.01, 0.02$)[4] on non-cuspy diversity. Furthermore, it returns significantly fewer cuspy paths than the baselines. Perhaps surprisingly, we see that the T-test method actually performs worse than the plateau method on our metrics, despite involving extra steps to filter out bad routes. To investigate further, we directly measured the effectiveness of the T-test as a signal for detecting cuspy routes.

Recall that for a given query, the initial bidirectional search returns a set of candidate via-nodes. Each candidate via-node is considered either cuspy if its uniform stretch exceeds the threshold u or non-cuspy otherwise. Given a predictor of non-cuspiness, we can measure its precision (fraction of predicted non-cuspy nodes that were actually non-cuspy) and recall (fraction of non-cuspy nodes that were predicted as such) on this set. We compared precision/recall curves for the following predictors:

1. **Model prediction.** A node is predicted as non-cuspy if its model-predicted probability of being good is above a threshold p, where we varied p in equally spaced increments from 0 to 1.

2. **Plateau.** A node is predicted as non-cuspy if the ratio of its plateau size to its via-path cost is above a threshold r, where we varied r in equally spaced increments from 0 to 0.2.

3. **T-test.** A node is predicted as non-cuspy if it passes the T-test. We varied T in equally spaced increments from 0 to 0.2.

Our precision and recall results are shown in Figure 3. As can be seen in the plot, our learned model is a much stronger predictor of non-cuspiness than either heuristic method. Although the T-test does remove bad candidates, it also removes good ones, and for larger values of T ($T \approx 0.1$), we found that often only a few candidates passed the T-test. This is consistent with the original usage of the T-test, which was in the context of finding only a single (or very few) alternates (Abraham et al. [2013]).

[4]Best-performing T values among a range we tested from 0 to 0.2.

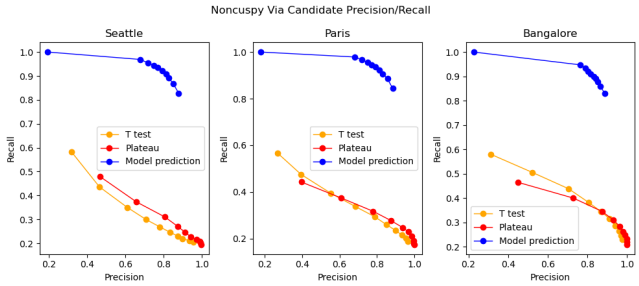


Figure 3: Precision and recall for detecting non-cuspy candidate via-nodes. The precision and recall were measured individually on each of 100 sample queries, and these precision/recall values were then averaged over the query set.

We also evaluated the algorithms at a cuspiness threshold of $u = 1.5$, shown in Appendix 7.1, Figure 4. At this threshold, our model still matches or outperforms both baselines in non-cuspy diversity and returns significantly fewer cuspy paths. The difference can be observed qualitatively in Appendix 7.2 Figure 6, which shows a query in the Seattle metro area. The model prediction-based alternates do not have significant cusps, while the plateau method returns several clearly undesirable routes.

Finally, Table 2 shows the average time it took to produce 30 alternates, averaged across our queries from all three metros. The scoring time column measures the time spent evaluating each candidate via-node, which is the only part that varies between the different ranking methods. Path expansion time refers to the time spent expanding the CRP shortcuts once the via-nodes are selected.

The running time of our model-based approach and the plateau baseline are both slightly over 200ms, with the vast majority of that time spent in path expansion. In particular, both scoring steps ran in well under 10ms, fast enough for practical applications. Note that we did not optimize the path expansion, since it was shared across all methods we evaluated.

Both the T-test and uniform stretch filter ran significantly slower due to having to non-trivially process

each candidate via-node. Typically queries had hundreds of such candidates, even after filtering out those with stretch above 1.5 (see Appendix 7.1, Figure 5). Our implementation of the T-test could have been optimized in various ways, but we did not do so given its relatively low performance on evaluation metrics.

5 CONCLUSION

In this work, we propose a new approach to finding alternative routes in road networks based on a neural network model that learns a good representation of the underlying graph. Using our model, we are able to produce routes with low uniform stretch, something which is expensive to achieve using traditional algorithms.

Our work opens up possibilities for future research. Continuing along the lines of alternates generation, one can explore other training objectives besides uniform stretch. For example, we could change the model to output an embedding vector $f(v) \in \mathbb{R}^d$ for each via-node candidate v. Then, we could train the model so that $f(v)$ and $f(v')$ are close if and only if the via-paths of v and v' have high overlap. This would provide a fast way of clustering candidate paths without having to make expensive path overlap computations.

Another future direction is to optimize the model. Although our model is already practical for metro-sized graphs (or even larger), we believe there is potential to scale much further. The embeddings for each individual node account for most of the model size. We could imagine a model that instead operates on only level 1 boundary nodes (around 40x fewer, see Table 1) and relies on local searches to connect the origin and destination to the level 1 boundary.

Finally, we believe that the paradigm of training a road network representation can be a general way to incorporate almost any kind of additional information when suggesting a route (whether as an alternate or as a main route). By augmenting the training procedure with richer data, we could train a model that suggests routes that account for traffic and traffic variability, vehicle types, individual user preferences, and more.

6 ACKNOWLEDGEMENTS

We thank our colleagues Pranjal Awasthi and Zhengdao Chen for fruitful discussions on topics related to this work.

References

Camvit: Choice routing. `http://www.camvit.com`, 2009.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. Alternative routes in road networks. *ACM J. Exp. Algorithmics*, 18, 2013.

Roland Bader, Jonathan Dees, Robert Geisberger, and Peter Sanders. Alternative route graphs in road networks. In Alberto Marchetti-Spaccamela and Michael Segal, editors, *Theory and Practice of Algorithms in (Computer) Systems*, pages 21–32, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19754-3.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940, 2016. URL `http://arxiv.org/abs/1611.09940`.

M Ben-Akiva, M J Bergman, Andrew J Daly, and Rohit Ramaswamy. Modeling inter-urban route choice behaviour. In *International Symposium on Transportation and Traffic Theory*, pages 299–330. VNU Press, 1984.

Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, Ulf Leser, and David B. Blumenthal. Finding k-shortest paths with limited overlap. *VLDB J.*, 29(5):1023–1047, 2020. doi: 10.1007/s00778-020-00604-x. URL `https://doi.org/10.1007/s00778-020-00604-x`.

Tomas de la Barra and B Perezand J Anez. Multidimensional path search and assignment. In *PTRC Summer Annual Meeting (SAM)*, 1993.

Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Customizable route planning in road networks. *Transp. Sci.*, 51(2):566–591, 2017.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the TSP by policy gradient. In Willem Jan van Hoeve, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings*, volume 10848 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2018. doi: 10.1007/978-3-319-93031-2_12. URL `https://doi.org/10.1007/978-3-319-93031-2_12`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short*

Papers), pages 4171–4186. Association for Computational Linguistics, 2019.

Christian Häcker, Panagiotis Bouros, Theodoros Chondrogiannis, and Ernst Althaus. Most diverse near-shortest paths. In Xiaofeng Meng, Fusheng Wang, Chang-Tien Lu, Yan Huang, Shashi Shekhar, and Xing Xie, editors, *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*, pages 229–239. ACM, 2021. doi: 10.1145/3474717.3483955. URL https://doi.org/10.1145/3474717.3483955.

Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *CoRR*, abs/2101.11174, 2021. URL https://arxiv.org/abs/2101.11174.

Yoav Kaempfer and Lior Wolf. Learning the multiple traveling salesmen problem with permutation invariant pooling networks. *CoRR*, abs/1803.09621, 2018. URL http://arxiv.org/abs/1803.09621.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6348–6358, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html.

Moritz Kobitzsch. An alternative approach to alternative routes: Hidar. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, volume 8125 of *Lecture Notes in Computer Science*, pages 613–624. Springer, 2013. doi: 10.1007/978-3-642-40450-4_52. URL https://doi.org/10.1007/978-3-642-40450-4_52.

Moritz Kobitzsch. *Alternative Route Techniques and their Applications to the Stochastics on-time Arrival Problem*. PhD thesis, Karlsruhe Institute of Technology, 2015.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Lingxiao Li, Muhammad Aamir Cheema, Mohammed Eunus Ali, Hua Lu, and David Taniar. Continuously monitoring alternative shortest paths on road networks. *Proc. VLDB Endow.*, 13(11):2243–2255, 2020. URL http://www.vldb.org/pvldb/vol13/p2243-li.pdf.

Huiping Liu, Cheqing Jin, Bin Yang, and Aoying Zhou. Finding top-k shortest paths with diversity. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1761–1762. IEEE Computer Society, 2018. doi: 10.1109/ICDE.2018.00238. URL https://doi.org/10.1109/ICDE.2018.00238.

Zihan Luo, Lei Li, Mengxuan Zhang, Wen Hua, Yehong Xu, and Xiaofang Zhou. Diversified top-k route planning in road network. *Proc. VLDB Endow.*, 15(11):3199–3212, 2022. URL https://www.vldb.org/pvldb/vol15/p3199-luo.pdf.

Dennis Luxen and Dennis Schieferdecker. Candidate sets for alternative routes in road networks. In Ralf Klasing, editor, *Experimental Algorithms - 11th International Symposium, SEA 2012, Bordeaux, France, June 7-9, 2012. Proceedings*, volume 7276 of *Lecture Notes in Computer Science*, pages 260–270. Springer, 2012.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 11096–11107, 2021.

MohammadReza Nazari, Afshin Oroojlooy, Lawrence V. Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9861–9871, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/9fb4651c05b2ed70fba5afe0b039a550-Abstract.html.

Andreas Paraskevopoulos and Christos D. Zaroliagis. Improved alternative route planning. In Daniele Frigioni and Sebastian Stiller, editors, *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2013, September 5, 2013, Sophia Antipolis, France*, volume 33 of *OASICS*, pages 108–122. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

Priyadarshan Patil. Applications of deep learning in traffic management: A review. *International Journal of Business Intelligence and Big Data Analytics*, 5(1):16–23, Jan. 2022. URL https://research.tensorgate.org/index.php/IJBIBDA/article/view/26.

Pedro Perez-Murueta, Alfonso Gómez-Espinosa, Cesar Cardenas, and Miguel Gonzalez-Mendoza. Deep learning system for vehicular re-routing and congestion avoidance. *Applied Sciences*, 9(13), 2019. ISSN 2076-3417. doi: 10.3390/app9132717. URL https://www.mdpi.com/2076-3417/9/13/2717.

Pratyush Raj, M Sravan Kumar, and Priyanka Dwivedi. An embedded deep learning based traffic advisory system. In *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pages 1–5, 2020. doi: 10.1109/ICRAIE51050.2020.9358364.

Rajib Saha, Mosammat Tahnin Tariq, and Mohammed Hadi. Deep learning approach for predictive analytics to support diversion during freeway incidents. *Transportation Research Record*, 2674(6):480–492, 2020. doi: 10.1177/0361198120917673.

Ali Kemal Sinop, Lisa Fawcett, Sreenivas Gollapudi, and Kostas Kollias. Robust routing using electrical flows. In Xiaofeng Meng, Fusheng Wang, Chang-Tien Lu, Yan Huang, Shashi Shekhar, and Xing Xie, editors, *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*, pages 282–292. ACM, 2021.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971. ISSN 00251909, 15265501.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Please see Sections 2 and 3.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No] We did not provide analysis of Algorithm 1 as it is not central to the point of the paper. However, we are happy to provide a more detailed analysis if requested.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] We provide descriptions of the experiments and the steps needed to complete them but not actual code. We have ran our experiments on an actual system which includes proprietary data and code.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] We have given most important training details in 3.2. We are happy to provide further details if requested.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No] We have provided clear definitions of metrics used in Section 2.2. We did not include error bars because we felt they were not very informative and would hurt the presentation of the plots.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No] We have not shared details about the computing infrastructure used because it includes proprietary information.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

7 APPENDIX

7.1 Additional Figures

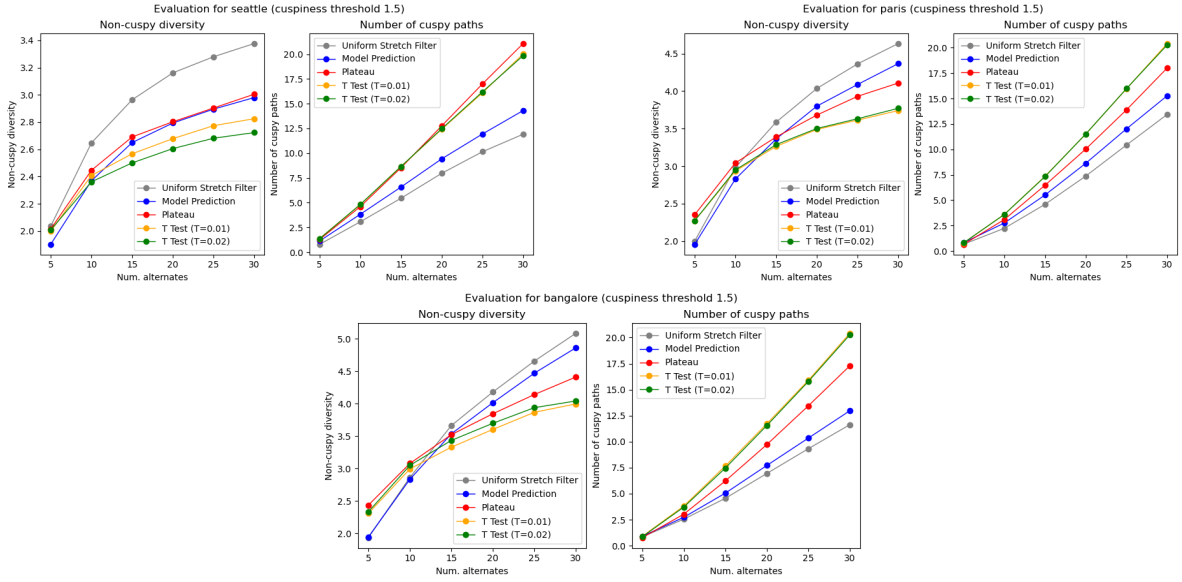


Figure 4: Diversity and number of cuspy paths for Seattle, Paris, and Bangalore metros, with uniform stretch threshold 1.5. Note that the uniform stretch filter is still applied at a 2.0 threshold, so it produces some paths with uniform stretch above 1.5.

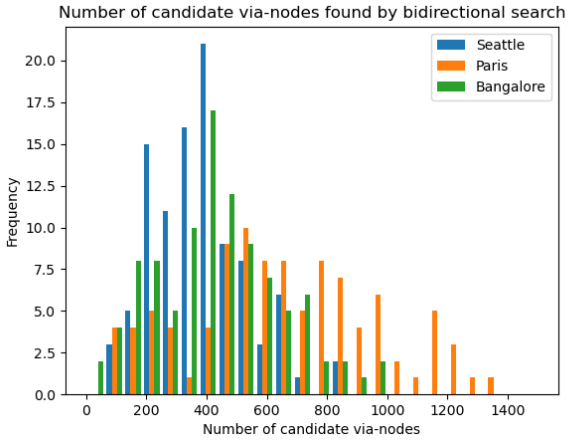
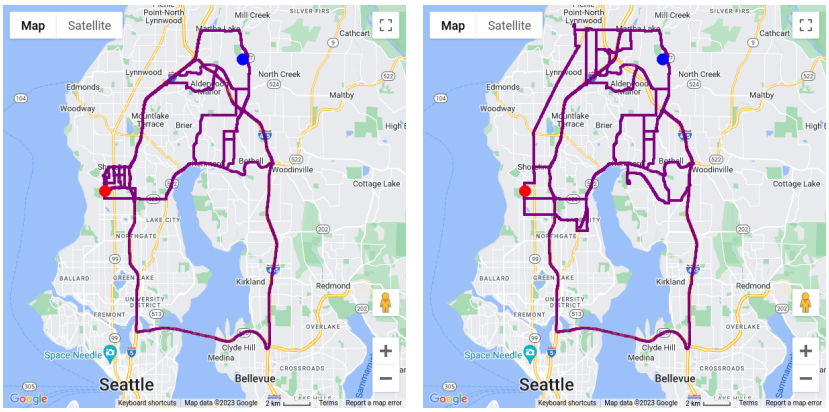


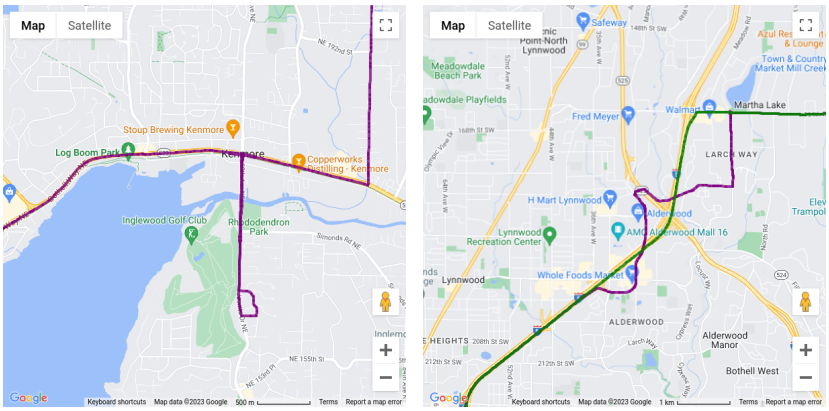
Figure 5: The distribution of the number of potential via-nodes found by bidirectional search.

7.2 Qualitative Examples



(a) model prediction (b) plateau

Figure 6: Example alternates generated for a query in Seattle. The model-scored results (left) included no cuspy paths, while the plateau (right) results include noticeable cusps.



(a) (b)

Figure 7: Close-up of some bad paths returned by the plateau method in Figure 6. In (a), the path makes an unnecessary detour from the highway to go through a via-node. In (b), the returned path (purple) takes a very circuitous route compared to the main path (green).