Beyond Shortest Paths: Node Fairness in Route Recommendation

Antonio Ferrara
antonio.ferrara@centai.eu
CENTAI, Turin, Italy
TU Graz, Austria

David García-Soriano
david.garcia.soriano@upc.edu
Universitat Politècnica de Catalunya
Serra Húnter Fellow, Barcelona, Spain

Francesco Bonchi
bonchi@centai.eu
CENTAI, Turin, Italy
Eurecat, Barcelona, Spain

ABSTRACT

Traditionally, route recommendation systems focused on minimizing distance (or time) to travel between two points. However, recent attention has shifted to other factors beyond mere length. This paper addresses the challenge of ensuring a fair distribution of visits among network nodes when handling a high volume of point-to-point path queries. In doing so, we adopt a *Rawlsian* notion of individual-level fairness exploiting the power of randomization. Specifically, we aim to create a probabilistic distribution over paths that maximizes the minimum probability of any eligible node being included in the recommended path.

A key idea of our work is the notion of *forward paths*, i.e., paths where travelling along any edge decreases the distance to the destination. In unweighted graphs forward paths and shortest paths coincide, but in weighted graphs forward paths provide a richer set of alternative routes, involving many more nodes while remaining close in length to the shortest path. Thus, they offer diversity and a wider basis for fairness, while maintaining near-optimal path lengths. We devise an algorithm that extracts a directed acyclic graph (DAG) containing all the forward paths in the input graph, with the same computational runtime as solving a single shortest-path query. This avoids enumerating all possible forward paths, which can be exponential in the number of nodes. We then design a flow problem on this DAG to derive the probabilistic distribution over forward paths with the desired fairness property, solvable in polynomial time through a sequence of small linear programs.

Our experiments on real-world datasets validate our theoretical results, demonstrating that our technique provides individual node satisfaction while maintaining near-optimal path lengths. Moreover, our experiments show that our method can handle networks with millions of nodes and edges on a commodity laptop, and scales better than the baselines when there is a large volume of path queries for the same source and destination pair.

PVLDB Reference Format:
Antonio Ferrara, David García-Soriano, and Francesco Bonchi. Beyond Shortest Paths: Node Fairness in Route Recommendation. PVLDB, 18(9): 3230 - 3242, 2025.
doi:10.14778/3746405.3746440

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Ambress92/MaxMinFairForwardPaths.

Proceedings of the VLDB Endowment, Vol. 18, No. 9 ISSN 2150-8097.
doi:10.14778/3746405.3746440

1 INTRODUCTION

Route recommendation systems have become essential tools in everyday life for navigation, deliveries, and trip planning [7, 8, 28, 35, 37, 45]. In every smartphone or personal device, apps such as Google Maps help users find the most effective route from a source to a destination by driving, public transportation, or walking. Such point-to-point queries can be solved by classic shortest-path algorithms [2, 14, 31, 40, 44] by converting the physical space into a graph structure, where nodes represent locations, edges represent road segments, and the edge weights can be used to represent travel costs in terms of time, distance, or other criteria.

Modern route recommendation systems are able to take into consideration other factors beyond the physical space (e.g., traffic status, delays in public transportation, varying user preferences, weather, etc.), thus recommending routes beyond the mere utilitarian value of shortest paths. Researchers have also studied methods to recommend routes that maximize human factors such as enjoyment, perceived safety, and overall positive experience of users, while minimizing the risk of crime, accidents, or pollution [33, 39]. While most of this research focuses on the system's benefit for the users, scant attention has been paid to ensuring fair exposure of the items forming the recommendation, i.e., nodes or points-of-interest (POIs), which may or may not be part of the recommended route. In this paper we tackle a novel problem thus far overlooked in the literature: *how to guarantee a fair distribution of visits among the nodes of the network, when providing route recommendations.*

Consider the following motivating example:

Example 1 (Node individual fairness.). *The tourism office of the municipality of Florence develops an app to help tourists navigate the city center and move between the key attractions and historic landmarks. Shortly after the release of the app, an ice-cream parlor, located in a strategic position for tourists flowing between the Central Train Station and the Uffizi Gallery, suffers a sudden drop in sales. Upon investigation, they realize that all tourists moving between the two landmarks were being routed through the same shortest path, greatly reducing the visibility of the ice-cream parlor, which was located along a slightly different path. At the same time, a newly opened ice-cream parlor located on the recommended shortest path experiences a surge in sales.*

The notion of *item fairness*, which concerns whether the recommendation allocates exposure to items fairly, is well established in the literature on fairness in recommender systems [5, 12, 42]. However, it has received very little attention in the context of route recommendations. Some previous work considered a *group-level* notion of fairness in point-to-point shortest path queries on *vertex-colored graphs*, requiring that paths pass through a balanced number of nodes of different colors [3, 4]. The problem we introduce in this paper, however, departs significantly from previous formulations, as we focus on *individual fairness* for the nodes, requiring that they are *equitably covered* by the recommendations. To the best of our knowledge, we are the first to address this requirement.

Related to our proposal is the notion of *diversity* of the recommendation. In fact, it is well known that improvements in the items' individual fairness are likely to increase diversity [42], due to the larger number of different items covered by some recommendations [27]. However, the converse need not hold: increasing diversity does not necessarily improve item fairness [42]. Although providing a set of diverse paths also enlarges the set of nodes covered by the recommended routes, as our experiments in Section 5 prove, this does not guarantee that the nodes in the graph are *equitably covered.*

Forward paths. When addressing diversity along with fairness, it is often necessary to relax the requirement that the route be the shortest, as in weighted networks – like real-world road networks – the point-to-point shortest path is typically unique. This relaxation is usually achieved by introducing a parameter to control the desired path-length deviation from the shortest path [23, 25]. However, in this paper, we take a different approach.

Our first contribution is the concept of *forward paths.* Intuitively, each step along a forward path brings you closer to your destination. This aligns well with user preferences in real-world applications, as travelers generally favor routes that consistently progress toward their destination without backtracking or moving in circles. In unweighted graphs, forward paths coincide with shortest paths, while, in weighted graphs, forward paths visit a wider variety of nodes, making them a suitable foundation for fairness. Figure 1(a) provides a toy example depicting the shortest path, a forward path that is not a shortest path, and another path that is neither.

Although instances can be constructed where the ratio between forward path length and shortest path length may be arbitrarily large, we show empirically that in many real-world weighted graphs, *forward paths are very close in length to the shortest path.*[1] For example, in the network of drivable streets of Piedmont (California, USA) shown in Figure 1(b), for randomly sampled source-destination pair of nodes, there are on average 634 different forward paths while the shortest path is almost always unique. Forward paths visit 81% more nodes on average than the shortest paths, and the longest forward path is only 11% longer than the shortest path.

Based on these observations, we adopt forward paths as our notion of near-shortest paths, removing the need for the parameter to control path-length deviation from the shortest path.

Maxmin distributional fairness. The second ingredient of our solution is the fairness criterion we employ. In particular, given our focus on fairness from the individual perspective of the nodes of the network, we adopt the notion of *maxmin distributional fairness* [17, 18, 26, 32, 36]. Such a paradigm exploits the power of randomization to provide *individual ex-ante fairness* and is inspired by Rawls's theory of justice [34], which advocates arranging social and financial inequalities to the benefit of the worst-off.

In the context of route recommendation, this translates into the goal of producing a probabilistic distribution over valid routes, such that it is impossible to improve the probability of any suitable location being visited, without decreasing it for some other location which already has a lower probability. In our context, the routes

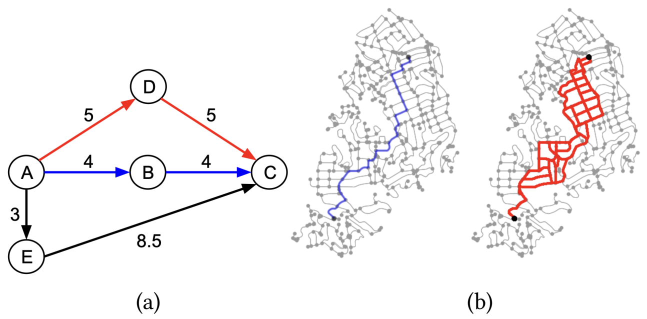


Figure 1: (a) Toy example: in red, a forward path that is not the shortest path (blue). Moving from the source Ⓐ to Ⓓ reduces the distance to the destination Ⓒ from 8 to 5, satisfying the definition of forward path. In black, a path that is not forward: moving from Ⓐ to Ⓔ increases the distance to Ⓒ from 8 to 8.5. (b) Drivable street network of Piedmont (California). For a random pair of source-destination nodes, the shortest path is shown in blue, while the union of the forward paths is highlighted in red. In this dataset, the shortest path between random pairs is almost always unique. By contrast, on average there are 634 different forward paths, visiting 81% more nodes than the shortest path, with the *longest* forward path being only 11% longer than the shortest path.

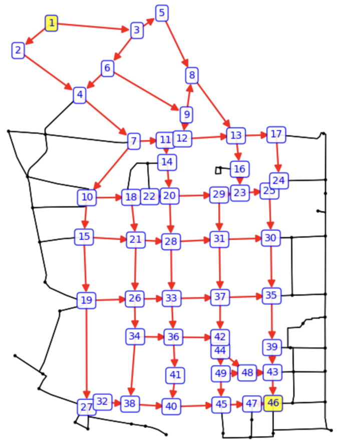
that we consider valid are forward paths, and the nodes that are subject to the fairness requirement are precisely the nodes that belong to at least one forward path.

Wrapping up, the technical problem we address in this paper (formally defined as Problem 1 in Section 4) is the following: *given a network, a source node s, a destination node t, produce a probabilistic distribution over forward paths from s to t that is maxmin-fair for all the eligible nodes (i.e., those that belong to at least one forward path).*

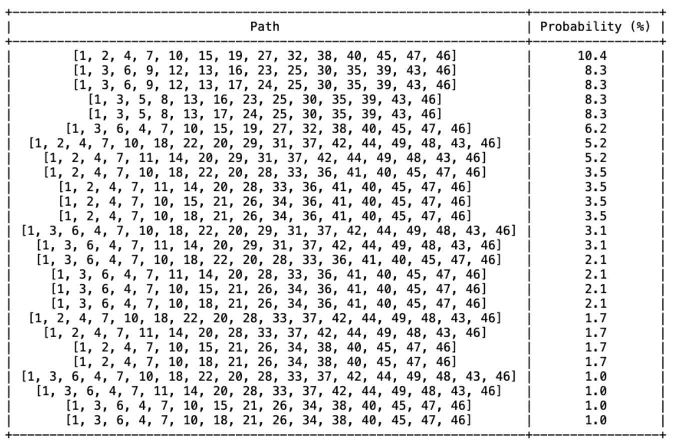
Example 2 (Maxmin-fair distribution over forward paths.). *In Figure 2 we use the setting of Example 1 to showcase the input and output of the problem we address. In Figure 2(a) we show a portion of the real road network of the city center of Florence. Our input is a source-destination pair of nodes, marked in yellow: node 1 (Central Train Station) and node 46 (Uffizi Gallery). The union of all forward paths is depicted in red. A maxmin-fair distribution assigns probabilities to the possible paths. In Figure 2(b) we report the output, a maxmin-fair distribution over forward paths: a list of paths from the source to the destination, each associated to its probability of being recommended, satisfying the desired fairness property.*

Practical implications and deployment. Our proposal embraces randomization to produce a probability distribution over forward paths that guarantees the maximum possible visibility to all eligible nodes. This distributional fairness approach is very well suited to contexts in which the same query can be served many times for different users of a platform. Consider again Example 1, where tens of thousands of path requests may be received every day for top landmarks, such as Florence Central Station and the Uffizi Gallery. Having precomputed a maxmin-fair distribution, such as the one in Figure 2(b), it becomes easy to serve all the requests simply by sampling a route recommendation for any request independently,

[1]See Table 1 in Section 3 for empirical results on real-world road networks.



(a)



Path	Probability (%)
[1, 2, 4, 7, 10, 15, 19, 27, 32, 38, 40, 45, 47, 46]	10.4
[1, 3, 6, 9, 12, 13, 16, 23, 25, 30, 35, 39, 43, 46]	8.3
[1, 3, 6, 9, 12, 13, 17, 24, 25, 30, 35, 39, 43, 46]	8.3
[1, 3, 5, 8, 13, 16, 23, 25, 30, 35, 39, 43, 46]	8.3
[1, 3, 5, 8, 13, 17, 24, 25, 30, 35, 39, 43, 46]	8.3
[1, 3, 6, 4, 7, 10, 15, 19, 27, 32, 38, 40, 45, 47, 46]	6.2
[1, 2, 4, 7, 10, 18, 22, 20, 29, 31, 37, 42, 44, 49, 48, 43, 46]	5.2
[1, 2, 4, 7, 11, 14, 20, 29, 31, 37, 42, 44, 49, 48, 43, 46]	5.2
[1, 2, 4, 7, 10, 18, 22, 20, 28, 33, 36, 41, 40, 45, 47, 46]	3.5
[1, 2, 4, 7, 11, 14, 20, 28, 33, 36, 41, 40, 45, 47, 46]	3.5
[1, 2, 4, 7, 10, 15, 21, 26, 34, 36, 41, 40, 45, 47, 46]	3.5
[1, 2, 4, 7, 10, 18, 21, 26, 34, 36, 41, 40, 45, 47, 46]	3.5
[1, 3, 6, 4, 7, 10, 18, 22, 20, 29, 31, 37, 42, 44, 49, 48, 43, 46]	3.1
[1, 3, 6, 4, 7, 11, 14, 20, 29, 31, 37, 42, 44, 49, 48, 43, 46]	3.1
[1, 3, 6, 4, 7, 10, 18, 22, 20, 28, 33, 36, 41, 40, 45, 47, 46]	2.1
[1, 3, 6, 4, 7, 11, 14, 20, 28, 33, 36, 41, 40, 45, 47, 46]	2.1
[1, 3, 6, 4, 7, 10, 15, 21, 26, 34, 36, 41, 40, 45, 47, 46]	2.1
[1, 3, 6, 4, 7, 10, 18, 21, 26, 34, 36, 41, 40, 45, 47, 46]	2.1
[1, 2, 4, 7, 10, 18, 22, 20, 28, 33, 37, 42, 44, 49, 48, 43, 46]	1.7
[1, 2, 4, 7, 11, 14, 20, 28, 33, 37, 42, 44, 49, 48, 43, 46]	1.7
[1, 2, 4, 7, 10, 15, 21, 26, 34, 38, 40, 45, 47, 46]	1.7
[1, 2, 4, 7, 10, 18, 21, 26, 34, 38, 40, 45, 47, 46]	1.7
[1, 3, 6, 4, 7, 10, 18, 22, 20, 28, 33, 37, 42, 44, 49, 48, 43, 46]	1.0
[1, 3, 6, 4, 7, 11, 14, 20, 28, 33, 37, 42, 44, 49, 48, 43, 46]	1.0
[1, 3, 6, 4, 7, 10, 15, 21, 26, 34, 38, 40, 45, 47, 46]	1.0
[1, 3, 6, 4, 7, 10, 18, 21, 26, 34, 38, 40, 45, 47, 46]	1.0

(b)

Figure 2: (a) A portion of the Florence city-center road network. The input to our problem, along with the map, is a source-destination pair: here the user needs to go from the Central Train Station (node 1) to the Uffizi Gallery (node 46). The first step of our solution is constructing the DAG of forward paths (Algorithm 1 in Section 3), shown in red. The DAG is fed into Algorithm 2 (Section 4.1), which produces a probability distribution over forward paths (b) with the required fairness property (Problem 1, Section 4). Finally, the system samples one path according to the computed probabilities and suggests it to the user.

with the guarantee of fairness in the spirit of *amortized fairness over multiple trials* [5, 18, 38].

In contrast, the baseline methods from the literature on diverse path recommendations face significant challenges when computing high volumes of different alternative paths. We will show in Section 5 that, as the number of paths requested for a point-to-point query increases, our method becomes significantly faster than the baseline approaches. Additionally, it is worth noting that each source-destination pair defines its own problem instance, independent of other instances. Consequently, our proposal is highly parallelizable, since computations for different pairs can proceed concurrently. Moreover, the maxmin-fair distribution for a source-target pair needs only be computed once; after storing it, the only computation needed at query time is sampling from the distribution.

Technical challenges and roadmap. A first technical difficulty towards developing our solution is the need to enumerate the set of all possible forward paths, which, in the worst case, is exponential in the number of nodes. We show that it is possible to compute, with the same asymptotic computational runtime as solving a single shortest-path query, the Directed Acyclic Graph (DAG) representing all the forward paths from a given source to a given target, avoiding the need to explicitly enumerate the set of all possible forward paths.

The main technical challenge is to ensure a maxmin-fair distribution of forward paths. While the work of [18] proposes a general framework for a wide class of combinatorial problems based on solving numerous linear programs (LPs), applying it to our problem presents significant obstacles to overcome.

Firstly, such a method requires the existence of a *separation oracle* [21], which is a method to detect quickly a violated constraint in an exponentially large set of constraints. While this oracle can be shown to exist for exact shortest paths, it is unlikely to exist for near-shortest paths (see Section 4 for details).

Secondly, the general approach in [18] is purely theoretical, relying on solving an exponentially large LP via the ellipsoid algorithm [21], which is the only known algorithm to solve LPs in polynomial time using separation oracles. There is no practical implementation of the ellipsoid algorithm: all real-world LP solvers need all constraints provided upfront or added lazily (with no guarantee of polynomial-time efficiency in the latter case). We propose a compact LP formulation of maxmin-fair forward path distributions with polynomially many constraints and variables. This allows us to use any LP solver, yielding a practically efficient algorithm.

In summary, the contributions of this paper are as follows:

- We introduce the novel problem of individual fairness for the nodes of the network in point-to-point path queries, following the paradigm of maxmin distributional fairness.
- We introduce the concept of forward paths and show that, in weighted graphs, they provide a much wider set of alternative paths than shortest paths, without being much longer. We thus adopt them as our notion of near-shortest paths.
- We provide an algorithm to compute a Directed Acyclic Graph (DAG) representing all forward paths in $\Theta(|E| + |V| \log |V|)$ time, avoiding explicit enumeration of the set of all possible forward paths (of which there can be exponentially many).
- We design a flow problem which allows us to derive the desired maxmin-fair probabilistic distribution over forward paths. It can be solved in polynomial time by solving a sequence of small linear programs.
- We test our method on several real-world road networks. The experiments confirm our theoretical results showing that our method provides a more equitable distribution of visits among nodes than the baselines. We also demonstrate that our method can handle graphs with millions of edges on a commodity laptop.

The rest of the paper is organized as follows. Section 2 presents a brief survey of the related literature. Section 3 introduces the concept of forward paths and the algorithm to compute the DAG containing all forward paths. Section 4 provides the formal problem statement and our method for obtaining a maxmin-fair distribution on forward paths. Section 5 describes the experiments on real-world datasets. Finally, Section 6 discusses limitations and future work.

2 RELATED WORK

As clarified in the previous section, our work is the first to address the problem of guaranteeing an equitable distribution of visits among the nodes of the network when handling a high volume of route recommendations. On the one hand, our contribution can be placed within the wide literature on route recommendation systems (see, e.g., [45] for a recent survey). On the other hand, it naturally belongs in the literature about fairness in recommender systems, especially the part focusing on *item fairness*, i.e., the equitable allocation of exposure to the items being recommended (see, e.g., [12, 42] for recent surveys). However, neither of these two bodies of research offers closely related methods to be used as baselines in our evaluation. We already mentioned the work of Bentert et al. [3, 4] which, however, deals with a completely different notion of group fairness that is not relevant in our context. Hence, in order to identify suitable methods for comparison, we turn our attention to the literature about *diversity* in point-to-point path queries.

Diversity in point-to-point path queries. Chondrogiannis et al. [9] aim to find a set of k paths that are sufficiently dissimilar and as short as possible. Similarly, Luo et al. [30] study the top-k shortest paths under a maximum similarity threshold, using an edge deviation/concatenation method to avoid expensive graph searches.

Häcker et al. [23] limit the maximum path length to identify the k most diverse paths within the length constraint. Abraham et al. [1] explore alternative routes through single-via paths, formed by concatenating shortest paths through an intermediate vertex.

Hanaka et al. [25] focus on maximizing the sum of pairwise Hamming distances among the k shortest paths, but this is unsuitable for real-world weighted graphs due to the uniqueness of the shortest path. To address the challenge of finding diverse solutions, Hanaka et al. [24] present a framework for approximation algorithms in combinatorial problems.

Routing scenarios with precomputed roadmaps that restrict routes to avoid newly introduced obstacles are studied by Voss et al. [41]. Fahmin et al. [15] use precomputed hub labels to calculate shortest paths and prune already-visited hubs for alternative paths. Xie et al. [43] focus on the space complexity of storing alternative paths excluding specific vertices or edges.

While this literature does not address fairness, the diverse set of paths produced ensures that various nodes are visited. In our experiments (Section 5), we use the methods by Chondrogiannis et al. [9] and the DKSP algorithm from Luo et al. [30] as baselines. Additionally, we compare against Yen's algorithm [44], which computes the k shortest paths without diversity constraints.

Maxmin distributional fairness. The notion of fairness considered in this work is the maxmin distributional fairness originally defined by García-Soriano and Bonchi [17] in the context of bipartite matching, and later extended to different contexts: individual fairness in ranking under group-fairness constraints by García-Soriano and Bonchi [18], individual fairness in bipartite matching under group-fairness constraints by Panda et al. [32], max-cut and other combinatorial optimization problems by Salem et al. [36] and other graph-theoretic optimization problems Hojny et al. [26].

3 FORWARD PATHS

We are given a simple, directed, weighted graph $G = (V, E, \ell)$, where V is a finite set of nodes, E is a set of directed edges (ordered pairs of nodes, $(u, v) : u, v \in V$), and $\ell : E \rightarrow \mathbb{R}^{>0}$ is a positive weight function associating each edge $e = (u, v) \in E$ with its length $\ell(e)$. A path in a simple directed graph is a finite sequence of distinct nodes $P = (v_1, \ldots, v_k)$ such that $\forall i \in [k-1], (v_i, v_{i+1}) \in E$. The length of P is given by

$$\ell(P) = \sum_{i=1}^{k-1} \ell(v_i, v_{i+1}).$$

Given a pair of distinct nodes $s, t \in V$, an s-t-path is a path $P = (v_1, \ldots, v_k)$ such that $v_1 = s$ and $v_k = t$. We denote the set of s-t-paths by $\mathbf{P}_{s,t}$. Assuming $\mathbf{P}_{s,t} \neq \emptyset$, a shortest s-t-path is any s-t-path S such that

$$S \in \arg\min_{P \in \mathbf{P}_{s,t}} \ell(P) .$$

The *shortest-path distance* between any two distinct nodes $s, t \in V$, $s \neq t$ is $d(s, t) = \min_{P \in \mathbf{P}_{s,t}} \ell(P)$ if the minimum exists and $+\infty$ otherwise (i.e., if $\mathbf{P}_{s,t} = \emptyset$). When $s = t$, we define $d(s, t) = 0$.

In the remainder of the paper, we might omit the s-t- prefix when the start and end point of the path are clear from the context.

In a directed weighted graph, it is quite common for many s-t pairs to have a unique shortest path, which does not provide a good basis for fairness. Next, to address the uniqueness issue of the shortest path in weighted graphs, we define a novel type of path called *forward path* that provides a much larger set of alternatives, at the price of a slight increase in path length.

Definition 1 (**s-t-Forward path**). *An s-t-path $P = (v_1 = s, \ldots, v_k = t)$ in a graph G is called forward if for $i = 1, ..., k-1$,*

$$d(v_{i+1}, t) < d(v_i, t) ,$$

where d represents the shortest path distance function in G. We will write $\mathbf{FP}_{s,t}$ for set of forward paths from s to t.

Intuitively, the distance to the target decreases along each step in a forward path. Clearly, any shortest path is a forward path:

Proposition 1. *Let $G = (V, E, \ell)$ be a directed weighted graph. Any shortest s-t-path in G is a forward path.*

Proof. Let $S = (v_1 = s, \ldots, v_k = t)$ be an s-t-shortest path and take any edge (v_i, v_{i+1}) in S. Then $d(v_i, t) = d(v_{i+1}, t) + \ell(v_i, v_{i+1})$, because (v_i, v_{i+1}) belongs to a shortest path. Then, since $\ell(v_i, v_{i+1}) > 0$, $d(v_{i+1}, t) < d(v_i, t)$. This holds for any $i = 1, \ldots, k-1$, hence S satisfies the forward path condition. □

In unweighted graphs, i.e., graphs in which the weights are all one (i.e, $\ell \equiv 1$), it is more common to have various alternative shortest paths. In this case, shortest paths and forward paths coincide:

Table 1: The table contains data from the road networks of five cities extracted from OpenStreetMap and two datasets from the 9th DIMACS Implementation Challenge – Shortest Paths, namely the Florida and the Eastern USA datasets. Nodes represent locations and points of interest with a geographic position, stored as latitude-longitude pairs. Edges represent connections between these locations. The table reports, for 100 source-target pairs sampled uniformly at random, the average shortest path (SP), the average length of the longest forward path (LFP), and the ratio of the LFP length to the SP length. The ratio is computed per pair and averaged across all pairs. The remaining three columns show the average number of nodes visited by SP, FP (i.e., the node count of the DAG), and the mean and standard deviation of their ratio. Overall, we observe that with a minimal increase in the worst-case forward path length, the number of locations visited increases significantly.

Dataset	# nodes	# edges	SP length	LFP length	LFP/SP length	SP nodes	FP nodes	FP/SP nodes
Piedmont, California	352	937	1 860.07	2 061.58	1.11 ± 0.11	19.91	39.48	1.81 ± 0.66
Essaouira, Morocco	1 277	3 429	3 650.36	3 967.11	1.13 ± 0.14	37.20	107.03	2.52 ± 1.33
Florence, Italy	6 096	11 737	6 909.52	7 287.29	1.05 ± 0.05	75.10	121.27	1.58 ± 0.65
Buenos Aires, Argentina	17 890	37 474	9 065.83	9 642.57	1.06 ± 0.04	89.10	387.50	4.00 ± 2.12
Kyoto, Japan	44 828	118 087	8 501.42	9 243.75	1.09 ± 0.06	117.90	536.77	4.16 ± 2.21
Florida, USA	1 070 376	2 687 902	$4.12 \cdot 10^6$	$4.22 \cdot 10^6$	1.02 ± 0.02	1 194.27	2 720.63	2.33 ± 0.95
Eastern USA	3 598 623	8 708 058	$4.39 \cdot 10^6$	$4.57 \cdot 10^6$	1.04 ± 0.02	1 924.43	7 046.97	3.28 ± 1.45

Proposition 2. *Let $G = (V, E, \ell)$ be a directed unweighted (i.e., $\ell \equiv 1$) graph. An s-t-path $P = (v_1, \dots, v_k)$ is a shortest path if and only if it is a forward path.*

Proof. The "only if" part is a special case of Proposition 1. Conversely, let $F = (v_1 = s, \dots, v_k = t)$ be a forward path. Its length is $\ell(F) = k - 1$. From the forward path condition, we have $d(v_{i+1}, t) < d(v_i, t)$, and by the triangle inequality and the existence of the edge (v_i, v_{i+1}), we conclude that $d(v_i, t) \leq d(v_{i+1}, t) + 1$. Hence $d(v_i, t) > d(v_{i+1}, t) \geq d(v_i, t) - 1$. Since d can assume only integer values, we must in fact have $d(v_{i+1}, t) = d(v_i, t) - 1$. It follows that $d(s, t) = d(v_2, t) + 1 = \dots = d(v_{k-1}, t) + k - 2 = k - 1 = \ell(F)$, so F is a shortest path. □

Forward paths not only have the desirable property of getting closer to the target after traversing any edge, but also offer the flexibility of allowing for various alternatives and visiting a wider set of nodes, while remaining competitive in terms of length compared to the shortest path. Table 1 reports statistics on seven real-world road networks, confirming that forward paths, with a minimal increase in length, allow us to visit a significantly larger number of locations. We thus adopt them as our notion of near-shortest paths.

Constructing the DAG of all forward paths. In Section 4, we will discuss how to create a maxmin-fair distribution over forward paths. A preliminary step in our solution is to compute the Directed Acyclic Graph (DAG) representing all the forward paths from a given source to a given target, avoiding the need to explicitly enumerate the set of all possible forward paths, which, in the worst case, is exponential in the number of nodes of the DAG.[2] This is done by Algorithm 1, presented next.

DAG-FP takes as input a directed graph, a source, and a target node. It outputs a DAG in which any path from s to t is an s,t-forward path. The algorithm proceeds by computing a list containing the distance from the source to all the nodes, and the distance from any node to the target. These distances can be computed by Dijkstra's algorithm [14], once on the input graph and once more on the reversed graph. Then, it removes any node v for which there does not exist an s-t-path that also visits v. The induced subgraph containing all the remaining nodes and the edges among them is then considered, and all the edges for which the forward path property is not respected are removed. The resulting graph is returned.

[2]In fact, the longest forward path lengths reported in Table 1 have been computed by dynamic programming on this DAG.

Algorithm 1 DAG-FP: Creates a DAG containing the Forward Paths from s to t

```
Input: Graph $G = (V, E, \ell)$, source node $s$, target node $t$
Output: DAG with Forward Paths
1: $dist\_from\_s \leftarrow$ distance$(G, s, V)$
2: $dist\_to\_t \leftarrow$ distance$(G, V, t)$
3: $reachable\_from\_s \leftarrow \{i \mid dist\_from\_s[i] < \infty\}$
4: $reachable\_to\_t \leftarrow \{i \mid dist\_to\_t[i] < \infty\}$
5: $reachable \leftarrow reachable\_from\_s \cap reachable\_to\_t$
6: $G' \leftarrow G$.induced_subgraph$(reachable)$
7: for all $(u, v)$ in $G'.edges$ do
8:     if $dist\_to\_t[u] \leq dist\_to\_t[v]$ then
9:         remove $(u, v)$ from $G'$
10:    end if
11: end for
12: return $G'$
```

Proposition 3. *Algorithm 1 produces a DAG whose edge set is exactly the edges in G in some forward path from s to t.*

Proof. The DAG returned contains exactly those edges (u, v) where (a) $d(s, u) < \infty$, (b) $d(v, t) < \infty$, and (c) $d(u, t) > d(v, t)$. Indeed, edges where (a) or (b) fail are removed in line 6, and edges where (c) fails are removed in line 9; all other edges are kept. We need to show that an edge (u, v) belongs to some forward path if and only if these three conditions hold. The "only if" part follows directly from the definition of forward path. For the "if" part, assume that the three conditions hold. Then there is a path in G from s to u and another one from t to v; in particular, G has a shortest (hence forward) path from s to u and another shortest (hence forward) path from t to v. By pasting these two paths together with the edge (u, v) we obtain a forward path from s to t traversing (u, v), because $d(u, t) > d(v, t)$.

Finally, observe that the resulting graph is acyclic since the distance to t is finite and strictly decreasing along any path. □

Furthermore, the algorithm has the same asymptotic complexity as computing the distance from a source to all the nodes, which can be done with Dijkstra's algorithm. The following holds:

Proposition 4. *Algorithm 1 has computational complexity* $\Theta(|E| + |V| \log |V|)$.

Proof. Line 1 is computed with Dijkstra's algorithm, and also line 2, after reversing the direction of all the edges in the graph, can be computed with Dijkstra's algorithm starting from the target. Lines 3 to 11 have time complexity $\Theta(|E|)$. Dijkstra's algorithm with a Fibonacci heap priority queue has a worst-case running time of $\Theta(|E| + |V| \log |V|)$ [16], and hence also does Algorithm 1. □

Given the ability to efficiently compute the DAG that contains all the forward paths, one might be tempted to randomly sample paths from it. However, such random selection would lack fairness guarantees: one could end up visiting certain nodes too often, while ignoring others for no good reason, causing inequalities in terms of nodes' exposure. To avoid such inequalities, our goal is to produce a distribution on the forward paths that is individually fair to the nodes. We formalize this notion in the next section.

4 MAXMIN-FAIR FORWARD PATHS

Consider a general instance $\mathcal{T}$ of our path search problem, defined by a directed weighted graph $G = (V, E, \ell)$ and a pair of terminal nodes $s, t \in V$, which in turn implicitly defines the set of feasible solutions $\mathbf{FP}_{s,t}$ (i.e., the set of forward paths from s to t), which is finite and assumed to be non-empty. Let us associate with each path $P \in \mathbf{FP}_{s,t}$ and each node $v \in V$ a binary satisfaction $A(P, v) \in \{0, 1\}$ such that $A(P, v) = 1$ if $v \in P$ and 0 otherwise. Consider a randomized algorithm $\mathcal{A}$ that, for any given search problem $\mathcal{T}$, always halts and selects a solution path $\mathcal{A}(\mathcal{T}) \in \mathbf{FP}_{s,t}$. Then $\mathcal{A}$ induces a probability distribution $\mathbb{D}$ over $\mathbf{FP}_{s,t}$ by $\mathbb{P}_{\mathbb{D}}[P] = \mathbb{P}[\mathcal{A}(\mathcal{T}) = P]$, $\forall P \in \mathbf{FP}_{s,t}$. Let us denote the *expected satisfaction* of each node $u \in V$ under $\mathbb{D}$ by $\mathbb{D}[u] := \mathbb{E}_{P \sim \mathbb{D}}[A(P, u)]$.

Example 3 (Expected satisfaction.). *Consider again the maxmin-fair distribution of Figure 2(b). The expected satisfaction of all nodes (except the source and destination, which are trivially always satisfied) are reported in Table 2. This is simply the probability of being part of a forward path sampled from the distribution.*

It is worth highlighting that a distribution being fair does not require providing equal satisfaction probability to all eligible nodes. In fact, there are nodes that are better positioned to be part of a forward path from the given source to the given destination. Their "strategic" position should be naturally rewarded by a fair distribution, rather than penalized. Intuitively, a maxmin-fair distribution provides, on any given input instance, the strongest guarantee possible for all nodes, in terms of expected satisfaction [17]. In other words, a distribution $\mathbb{F}$ over $\mathbf{FP}_{s,t}$ is maxmin-fair for a set of nodes $U \subseteq V$ if it is impossible to improve the expected satisfaction of any node in U without decreasing it for some other node which is no better off. This is formalized in the following definition.

Table 2: Expected satisfaction of the nodes in Figure 2(a) under the probability distribution in Figure 2(b). Source and destination nodes are omitted, as they are always satisfied.

u	$\mathbb{D}[u]$ (%)	u	$\mathbb{D}[u]$ (%)	u	$\mathbb{D}[u]$ (%)	u	$\mathbb{D}[u]$ (%)
4	66.7	25	33.3	48	22.2	22	16.7
7	66.7	30	33.3	49	22.2	23	16.7
3	58.3	35	33.3	5	16.7	24	16.7
43	55.6	39	33.3	8	16.7	26	16.7
10	50.0	15	25.0	9	16.7	27	16.7
40	44.4	18	25.0	11	16.7	28	16.7
45	44.4	36	22.2	12	16.7	29	16.7
47	44.4	37	22.2	14	16.7	31	16.7
2	41.7	38	22.2	16	16.7	32	16.7
6	41.7	41	22.2	17	16.7	33	16.7
13	33.3	42	22.2	19	16.7	34	16.7
20	33.3	44	22.2	21	16.7	–	–

Definition 2. *A distribution* $\mathbb{F}$ *over* $\mathbf{FP}_{s,t}$ *is* ***maxmin-fair*** *for* $U \subseteq V$ *if for all distributions* $\mathbb{D}$ *over* $\mathbf{FP}_{s,t}$ *and all* $u \in U$,

$$\mathbb{D}[u] > \mathbb{F}[u] \Rightarrow \exists v \in U \mid \mathbb{D}[v] < \mathbb{F}[v] \leq \mathbb{F}[u].$$

With these concepts in place, we can now formally state our problem as follows:

Problem 1. *Given a graph* $G = (V, E, \ell)$ *and* $s, t \in V$, *our problem consists of finding a maxmin-fair distribution* $\mathbb{F}$ *over* $\mathbf{FP}_{s,t}$ *for the set* $U = \{u \in V \mid \exists P \in \mathbf{FP}_{s,t} \text{ such that } u \in P\}$.

There is some flexibility in choosing the graph G. Practical applications may require including only a subset of "valuable" nodes (such as points of interest, major intersections, or important landmarks) for fairness consideration. In these cases, a weighted graph based on the shortest distances between the valuable nodes can be constructed, which can serve as the input to our problem.

In the remainder of this section, we present a solution to Problem 1 by a carefully crafted linear programming algorithm, which avoids the need to compute the set of all possible forward paths (which can be exponential in the number of nodes in the DAG).

4.1 Algorithm

In [18], the authors introduce a general framework to solve maxmin-fair distributional problems. The problem defined above specializes their framework to forward paths, and is thus amenable to their techniques provided that one shows the existence of a certain separation oracle [21]. Specifically, given an assignment of non-negative weights to users, the separation oracle needs to find a feasible solution of maximum weight. The method of [18] applies whenever such a separation oracle admits an efficient implementation. When the set of feasible solutions is large (as in our setting), avoiding explicit enumeration is paramount.

We remark that the choice of forward paths in Problem 1 has the additional benefit of making the separation oracle tractable. Indeed, if we were to define the feasible set S as the set of simple paths from s to t, the separation oracle would ask for a simple path from s to t of maximum node weight. As the Hamiltonian path problem can be reduced to this question, it follows that the separation oracle problem for simple s, t-paths is NP-hard. However, when the graph is a DAG, there is a linear-time algorithm to find a simple path of maximum node weight: it suffices to process the

Table 3: The LP (1) encapsulating our problem and its dual (2).

$$
\begin{aligned}
\text{maximize} \quad & \lambda \in \mathbb{R} \quad \text{subject to} \\
& \sum_{u \mid (u,t) \in E} f_{u,t} = 1 \\
& \sum_{u \mid (s,u) \in E} -f_{s,u} = -1 \\
& \sum_{u \mid (u,v) \in E} f_{u,v} - \sum_{w \mid (v,w) \in E} f_{v,w} = 0 \quad \forall v \neq s,t \\
& \lambda - \sum_{u \mid (u,v) \in E} f_{u,v} \leq 0 \quad \forall v \notin K \\
& - \sum_{u \mid (u,v) \in E} f_{u,v} \leq -\alpha_v \quad \forall v \in K \\
& f_{u,v} \geq 0 \quad \forall (u,v) \in E
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
\text{minimize} \quad & d_t - d_s - \sum_{v \in K} \alpha_v w_v \quad \text{subject to} \\
& d_v - d_u - w_v \geq 0 \quad \forall (u,v) \in E \\
& \sum_{v \notin K} w_v = 1 \\
& w_v \geq 0 \quad \forall v \in V \\
& d_v \in \mathbb{R} \quad \forall v \in V
\end{aligned} \tag{2}
$$

vertices in topological order and apply dynamic programming (DP) (Section 24.2 of [11]). As we have shown in Section 3 that the set of all forward paths forms a DAG, it follows that the separation oracle for forward paths is polynomial-time solvable.

Using the DP-based separation oracle for the forward path DAG, one could theoretically apply the techniques from [18] to solve Problem 1. Unfortunately, the general algorithm proposed therein, while efficient in theory, requires the use of the impractical ellipsoid algorithm to solve exponential-sized linear programs in polynomial time. The ellipsoid method is the only known theoretically efficient method to solve large LPs via separation oracles, but it suffers from numerical stability issues and its performance is far from being competitive with the simplex or interior point methods used in established solvers. In fact, [18] poses as an open problem the development of faster algorithms for concrete problems.

We address this challenge by taking a different approach. By interpreting the probabilities of traversing the edges in a maxmin-fair solution as flows in a network, in the following we provide a compact Linear Programming (LP) formulation for maxmin-fair forward paths with only polynomially many constraints, thus bypassing the need of separation oracles and the ellipsoid algorithm.

Throughout our discussion, we use E to represent the set of edges in the forward path DAG for a certain fixed pair s, t of source-target vertices. To exclude trivial cases, we assume that the set $U - \{s, t\}$ is non-empty. Let us introduce a flow variable $f_{u,v}$ for each directed edge $(u, v) \in E$ denoting the (unconditional) probability that the edge $u \to v$ will be traversed in a path drawn from the solution. We interpret the search for a maxmin-fair distribution as a flow problem in G, where the probabilities "flow" from one vertex to its successors in the DAG. As every path starts at the source and ends at the target (and does not stall at any intermediate vertex), we need to ensure that (a) the incoming flow for the target and the outgoing flow from the source is 1; (b) flow conservation holds, i.e., the incoming flow equals the outgoing flow for every vertex other than s and t; and (c) the minimum incoming flow of a vertex is maximized (as required by the maxmin-fairness condition). All of these are expressible as linear constraints in the flow variables.

However, this does not quite suffice, since maxmin-fairness yields the lexicographically largest vector of sorted satisfaction probabilities [17]. To achieve this ordering, we need to solve several optimization problems. In order to do so, let α_v denote the satisfaction probability of v in the maxmin-fair distribution. We will obtain these values step by step, assuming that the values of α_v are known for an (initially empty) subset $K \subseteq V$ and augmenting K in each iteration. Thus, we wish to ensure that all members of K are visited with probabilities indicated by α, while we maximize the minimum visiting probability among the rest. Our problem is thus encapsulated by LP (1) in Table 3 or, equivalently, its dual (2).

Lemma 4.1. *Let $K \subseteq V$ and $\alpha \in \mathbb{R}^K$. Any distribution of forward paths from s to t satisfying the two conditions below gives rise to a solution $(\lambda, \{f_{u,v}\}_{(u,v) \in E})$ to LP (1):*

(a) The probability of visiting each vertex $v \in K$ is at least α_v;
(b) The probability of visiting each vertex $v \notin K$ is at least λ.

Conversely, from any solution $(\lambda, \{f_{u,v}\}_{(u,v) \in E})$ to LP (1), one may construct a distribution of forward paths from s to t such that (a) and (b) above hold.

Proof. Consider a distribution $\mathbb{D}$ of forward paths from s to t satisfying (a) and (b). Define $f_{u,v} = \Pr_{p \in \mathbb{D}}[\text{edge } (u,v) \text{ is traversed in } p] \geq 0$. Notice that when drawing a path from $\mathbb{D}$, the probability of entering vertex $v \neq s$ is $\sum_{u \mid (u,v) \in E} f_{u,v}$, and the probability of exiting $v \neq t$ is $\sum_{w \mid (v,w) \in E} f_{v,w}$. Hence these two must be equal for every $v \neq s, t$. Also, since the source and the target are visited with probability 1, the constraints $\sum_{u \mid (u,t) \in E} f_{u,t} = 1$ and $\sum_{u \mid (s,u) \in E} f_{s,u} = 1$ hold. Finally, we also have $\sum_{u \mid (u,v) \in E} f_{u,v} \geq \lambda$ when $v \notin K$ by condition (b) and $\sum_{u \mid (u,v) \in E} f_{u,v} \geq \alpha_v$ by (a). Hence all constraints are satisfied.

Conversely, take any solution $(\lambda, \{f_{u,v}\}_{(u,v) \in E})$ to LP (1). For each $(u, v) \in E$, let

$$
p_{u,v} = \begin{cases} f_{u,v} & \text{if } u = s \\ \frac{f_{u,v}}{\sum_{w \mid (w,u) \in E} f_{w,u}} \geq 0 & \text{if } u \neq s \end{cases} \tag{3}
$$

(We leave $p_{u,v}$ undefined when the denominator is zero.) Note that $\sum_{v \mid (u,v) \in E} p_{u,v} = 1$ for all $u \neq t$ because of the constraint $\sum_{u \mid (u,v) \in E} f_{u,v} = \sum_{w \mid (v,w) \in E} f_{v,w}$. So consider a random walk starting at s and stopping at t with transition probabilities $\{p_{u,v}\}_{(u,v) \in E}$, and define $e(v) = \sum_{u \mid (u,v) \in E} f_{u,v}$ if $v \neq s$ and 1 otherwise.

We show that $\Pr[v \text{ is visited}] = e(v)$. The proof proceeds by induction on the maximum length $q(v)$ of a path from s to v in the DAG G. The base case is when $q(v) = 0$ (i.e., $v = s$), and the property holds in this case because the path starts at s and $e(v) = 1$. For the inductive step, consider a vertex v and suppose that the

Algorithm 2 MMFP - MaxMin Fair Forward Paths

```
Input: DAG $G$, source node $s$, target node $t$
Output: Encoding of a maxmin-fair distribution for $s,t$-forward paths
 1: $K \leftarrow \emptyset$
 2: while $K \neq V$ do
 3:     Solve LP (1) and its dual LP (2);
 4:     let $\lambda^*$, $\{f^*_{u,v}\}$, $\{w^*_v\}$ be the optimum values
 5:     $K' \leftarrow \{v \notin K \mid w^*_v > 0\}$
 6:     $K \leftarrow K \cup K'$
 7:     for all $v \in K'$ do
 8:         $\alpha_v \leftarrow \lambda^*$
 9:     end for
10: end while
11: Return the flow values $f^*_{u,v}$
```

claim holds for all u with $q(u) < q(v)$. Observing that $(u,v) \in E$ implies $q(u) < q(v)$, we have

$$\Pr[v \text{ is visited}] = \sum_{u\mid(u,v)\in E} p_{u,v} \cdot \Pr[u \text{ is visited}] =$$
$$\sum_{u\mid(u,v)\in E} \frac{f_{u,v}}{e(u)} \cdot e(u) = \sum_{u\mid(u,v)\in E} f_{u,v} = e(v).$$

This establishes our claim.

Now observe that the walk must end (because the graph is finite and acyclic), but cannot end anywhere else than t (as for any vertex $u \neq t$ visited with non-zero probability $e(u) > 0$, the probability that the walk exits u is $\sum_{v\mid(u,v)\in E} p_{u,v} = 1$). In particular t is visited with probability one[3]. Moreover, properties (a) and (b) follow from the constraints $\sum_{u\mid(u,v)\in E} f_{u,v} \geq \lambda$ when $v \notin K$ by condition (b) and $\sum_{u\mid(u,v)\in E} f_{u,v} \geq \alpha_v$ when $v \in K$ by condition (a). □

Algorithm 2 outlines the complete procedure. It takes as input the DAG, constructed with Algorithm 1, as well as the source and target nodes, and outputs the flow values. The algorithm goes through the following steps. We maintain a set K containing the bottom nodes for which we already ensured a maxmin-fair probabilistic satisfaction; it is initialized with the empty set. Then, the algorithm iteratively performs the following steps until K contains all the nodes. First, LP (1) and its dual LP (2) are solved. Then K', a set containing all the nodes with dual weights greater than zero and not yet in K, is computed, and K gets updated with the nodes of K'. The nodes in K' then have their maxmin probabilistic satisfaction value α_v assigned to the optimum LP value λ^*; these values are then used as input for the updated LP (1). Finally, when the loop terminates, the flow values in the solution of the last LP 1 are returned.

Theorem 4.2. *Algorithm 2 finds a maxmin-fair distribution of forward paths in polynomial time.*

Proof. Given $K \subseteq V$ and $\{\alpha_v\}_{v\in K}$, call a distribution $\mathbb{D}$ of s-t-forward paths *compatible with* (K,α) if $\mathbb{D}[v] = \alpha_v$ for all $v \in K$ and $\mathbb{D}[v] \geq \alpha_w$ for all $v \notin K, w \in K$. We show that at the start of each iteration of the while loop, the maxmin-fair distribution $\mathbb{F}$ is compatible with (K,α). Furthermore, the size of K is strictly increasing on each iteration. This implies that the algorithm terminates with $\alpha_v = \mathbb{F}[v]$ for all $v \in V$.

The case $K = \emptyset$ is vacuous. We establish the general claim by induction. Assuming $\mathbb{F}$ was compatible with (K,α) at line 3, we need to argue that $\mathbb{F}$ is compatible with the next pair $(K \cup K', \alpha')$ after line 9 in the same iteration, where α' denotes the modified values of α. To this end, we employ the lexicographic characterization of maxmin-fairness [17]: a distribution is maxmin-fair iff its vector of satisfaction probabilities, sorted from low to high, is lexicographically largest. Since we know that $\mathbb{F}$ is compatible with (K,α), it follows from the aforementioned characterization and Lemma 4.1 that LP (1) is feasible and its optimum value λ^* equals $\min_{v\notin K} F[v]$. Observe that $K' \neq \emptyset$ because of the constraint $\sum_{v\notin K} w_v = 1$ in the dual. Moreover, by complementary slackness, the constraints in any optimal solution to the primal that are complementary to the non-zero dual variables are tight. Hence $\sum_{u\mid(u,v)\in E} f^*_{u,v} = \lambda^*$ for all $v \in K'$. In other words, any distribution $\mathbb{D}$ compatible with (K,α) and maximizing $\min_{v\notin K} D[v]$ needs to satisfy $D[v] = \lambda^*$ for all $v \in K'$. In particular this holds for $\mathbb{F}$, which implies that F is compatible with the next pair $(K \cup K', \alpha')$.

After the last iteration, we can recover a distribution of forward paths $\mathbb{D}$ determined by the last set of flow variables $f^*_{u,v}$ by performing the random walk described in the proof of the "conversely" part of Lemma 4.1. This distribution is maxmin-fair, since the satisfaction probabilities of $\mathbb{D}$ and $\mathbb{F}$ agree everywhere on V.

Regarding the running time, the number of iterations of the while loop is bounded by n, since $|K|$ is strictly increasing. Therefore at most n calls to the LP solver are made. The rest of the algorithm, including computing the forward path DAG, takes linear time. Since LP is solvable in polynomial time, so is the Maxmin Fair Forward Paths problem. Sampling each path from this distribution after the $f^*_{u,v}$ values have been found takes linear time as well. □

[3]In fact, this shows that the constraint $\sum_{(u,t)\in E} f_{u,t} = 1$ in LP (1) is actually redundant, and the variable d_t in the dual can be taken to be zero.

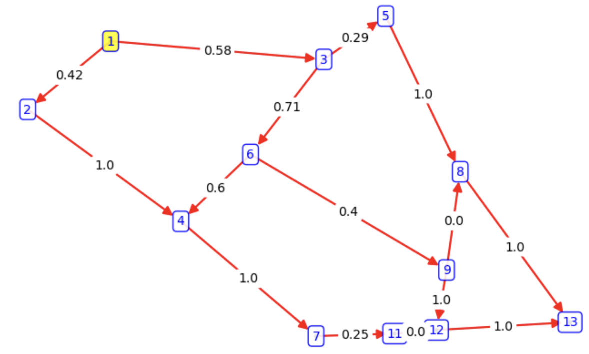


Figure 3: A portion of the DAG from Figure 2(a). The weights on the edges computed by Algorithm 2 encode the maxmin-fair distribution in Figure 2(b). Performing a random walk from the source to the target with the computed transition probabilities yields the maxmin-fair distribution.

It is worth noting that our method does not explicitly materialize the maxmin-fair distribution shown in Figure 2(b) due to its redundancy and space demands; indeed, many of the different routes share a common prefix. Instead, it encodes the distribution compactly by labeling the DAG edges with the correct transition probabilities (line 11 of Algorithm 2). We showed in Lemma 4.1 that a DAG always exists such that performing a random walk from the source to the target with its transition probabilities is equivalent to sampling directly a path from the maxmin-fair distribution.

Example 4 (Encoding the maxmin-fair distribution.). *Figure 3 illustrates how we encode the maxmin-fair distribution in Figure 2(b) by labeling the edges of the DAG in Figure 2(b) with the correct transition probabilities as computed by Algorithm 2. For space reasons, the figure only shows a portion of the DAG.*

Complexity and scalability. Recall that the DAG can be constructed in the same asymptotic time it takes to perform a single-source shortest path computation (Algorithm 1). Moreover, the time taken to sample each path given the flow values is linear in the path length. Hence, the bulk of the overall computational effort is spent on solving the sequence of LPs in Algorithm 2. The fastest algorithm to date for linear programming by [10] takes roughly $p^{2.37}D$, where p is the number of variables and D is the largest absolute value of the determinant of any submatrix of the matrix of coefficients. Since LP is not known to be solved in strongly polynomial time, such dependence on a parameter like D is unavoidable. In the worst case, our algorithm may need to solve up to n linear programs. However, as demonstrated in Section 5, these bounds are overly pessimistic. One reason is that, because we update K based on the value of the dual variables in line 6 of Algorithm 2 rather than one element at a time, the number of intermediate LP problems needed is much smaller than n in practice. Furthermore, our LPs are sparse if the original graph is, since the constraints follow the edge structure of the DAG of the graph. Finally, the actual practical performance LP solvers is much faster than the bounds suggest, and state-of-the-art commercial solvers like Gurobi [22] can handle LPs with hundreds of thousands of non-zeroes.

5 EXPERIMENTAL EVALUATION

In this section, we compare our proposal against several baselines. Our empirical analysis focuses on assessing (*i*) fairness of node exposure, (*ii*) path lengths, and (*iii*) runtime and scalability.

Datasets. We consider five real-world road networks extracted from OpenStreetMap (publicly available) with the OSMnx Python library [6]. Furthermore, we consider the graph of the state of Florida from the 9th DIMACS Implementation Challenge – Shortest Paths [13] and the Eastern region of the USA from the same collection. Statistics of the seven datasets used are reported in Table 1.

Queries. For the first six networks in Table 1, we sample uniformly at random 100 source to target pairs. For the Eastern USA network, we sample 100 pairs uniformly at random from the set of pairs within the distance range $\left[\frac{d_{\max}}{16}, \frac{d_{\max}}{8}\right]$, where $d_{\max}$ represents the estimated diameter of the graph. Following Luo et al. [30], we indicate these constrained pairs with Q1.

Measures. The main goal of our method is to guarantee individual fairness in terms of nodes' satisfaction. For our purposes, we compute the generalized Lorenz curve [29] between the bottom fraction of the nodes and the cumulative sum of the nodes' satisfaction, when the satisfactions are sorted in increasing order. Furthermore, we compute the Gini inequality coefficient [19], as $1 - 2 \cdot \mathrm{B}$, where B is the area under the Lorenz curve. The Gini coefficient ranges in $[0, 1]$; higher values are undesirable and indicate higher inequality levels between methods with similar total cumulative satisfactions.

As an (unattainably optimistic) reference line, we compute, for every k between 1 and the number n of nodes, the maximum cumulative satisfaction probability by a distribution of forward paths. We refer to the curve thus obtained as the *pointwise best*. It has been calculated for the four smallest networks using the sequence of majorized LPs devised in Section 3.1 of [20]. By definition, the generalized Lorenz curve of any method cannot be above it at any point. Although every point (k, y) in this curve can be attained by *some* distribution of paths, the entire curve cannot, in general, be attained by any fixed distribution. Indeed, the only case in which the pointwise best is attainable is when it coincides with the maxmin-fair distribution. However, it provides a useful reference as to how suboptimal each method is for different values of k simultaneously.

We also consider the length of the paths, as well as the running time of the methods, measured as wall-time. Due to the runtime of some baselines on the larger datasets, along with the need to run 100 different source-target pairs, we allow a maximum computational time of 1000 seconds per point-to-point query.

Experiments were run on a MacBook Pro with Apple $M1$ chip, 8 cores, and 16 GB RAM. To solve the LP problems, we have used the Gurobi optimizer [22].

Baselines. To the best of our knowledge, we are the first to investigate node fairness in path recommendations. As no direct comparison with previous work is possible, we use baselines from the literature on diversity in short paths. Specifically:

- **Yen**'s k-shortest path algorithm [44] computes a shortest path and then proceeds to find $k-1$ deviations of the shortest path.
- Two heuristic algorithms from Chondrogiannis et al. [9]: **OnePass+** (OP+) and **ESX-C**. The goal of these methods is to find a set of k paths as short as possible and for which their pairwise similarity (defined as a weighted fraction of edges shared by two paths) is below a certain pre-specified threshold $\theta \in [0, 1]$. Given a source node s and a target node t, OnePass+ traverses the road network expanding every path from the source to any node $v \in V$ satisfying certain shortness and diversity properties, until k paths are found. ESX-C, instead, executes shortest-path searches while progressively excluding edges from the road network. We use the default parameter value of $\theta = 0.8$ and consider $k = 10$ paths, unless stated otherwise in the tables and figures. In a preliminary analysis, smaller θ and higher k did not improve the results on node fairness, instead significantly increasing the runtime and path length.
- Moreover, we compare against **DKSP**, Luo et al. [30], an edge deviation and concatenation method, based on shortest path trees, which aims to diversify the top-k paths between a source-destination pair such that their pairwise similarities are under a threshold while their total length is minimal. To run DKSP, we set the similarity upper bound between each pair of paths to 0.9.
- Lastly, we consider random paths drawn uniformly at random from the set of all forward paths (**Random FP**). This is equivalent

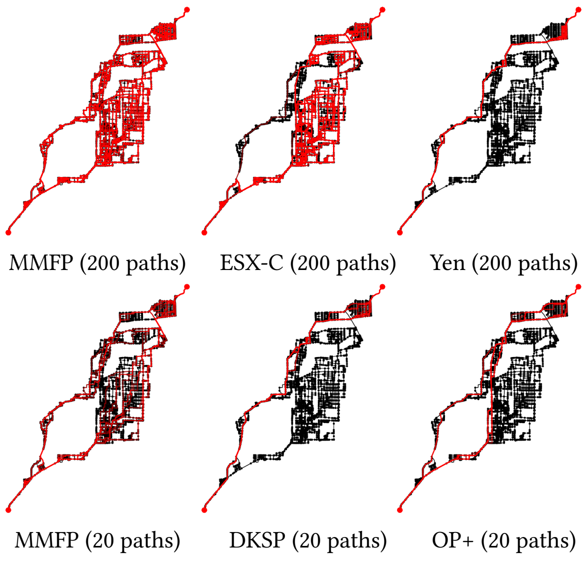


Figure 4: Visualization of the paths (in red) produced by different methods, applied to the DAG obtained for a source-target pair of the Kyoto dataset (nodes and edges in black). Our method, MMFP (top-left), visits nearly all the nodes. ESX-C (top-center) also performs well, leaving only a small portion of nodes unvisited. Yen (top-right) finds only minimal deviations from the shortest path. Due to the runtime limits of DKSP and OP+, the second row reports a comparison generating only 20 different paths. Even in this setting, MMFP with the same number of paths covers significantly more nodes.

to drawing random walks from s to t in the DAG, where the transition probability is uniform on the outgoing edges of a vertex. We sample 100 paths for each source-destination pair. Essentially, this results in the use of random forward paths only, without the maxmin-fairness optimization.

5.1 Comparison against the baselines

We start with the anecdotal evidence reported in Figure 4, where the union of the paths recommended by various methods is colored red. In the first row, we compare methods for which we could compute 200 paths, while in the bottom row we compare our method against baselines for which a maximum of 20 different paths could be computed. For our method MMFP, we sample from a maxmin-fair distribution the exact number of paths, while for the other methods the number of paths is a parameter to the algorithm. To facilitate visual comparison, all methods are applied directly to the DAG constructed for a random source-destination pair from the Kyoto dataset. The figure highlights how MMFP is able to recommend paths covering a broader set of nodes than the baselines. In particular, Yen and DKSP appear to produce only limited deviations from the shortest path. OP+ satisfies a few additional nodes, while ESX-C visits several more nodes but still leaves some groups of nodes uncovered.

Fairness. The first column of Tables 4-10 reports the mean and standard deviation of the Gini coefficient. MMFP obtains the lowest level of inequality with respect to the visibility of the nodes, consistently across all different datasets. This is further illustrated in Figure 5 (Left), where the generalized Lorenz curve for different methods is depicted (results for different source-target pairs and other datasets are analogous and not reported for space reasons). The horizontal axis represents the fraction of nodes receiving the least visibility, while the vertical axis contains the cumulative visibility received by that bottom fraction. Observe that the curve for our method lies consistently above the others, especially in its initial segment, corresponding to the least visible nodes. This highlights how our approach aligns with Rawls's theory of justice, which advocates maximizing benefits for the least advantaged. Note also how closely our method is to the ideal (yet unattainable) *pointwise best* curve.

So far, we have focused on the fact that MMFP guarantees ex-ante individual fairness. In Figure 5 (Center), we investigate the convergence of ex-post fairness in terms of the allocation of visibility as provided by forward paths sampled from the maxmin-fair distribution. As expected, increasing the number of sampled paths leads to an allocation that closely approximates the theoretical optimum. Specifically, with around 100 paths, the empirical generalized Lorenz curve is already very close to the ideal, while for 1000 sampled paths it becomes essentially identical. This shows the convergence of our technique to maxmin-fairness when serving a large volume of path queries.

Length of the paths. The second column of Tables 4-10 reports the average path length. Our method is competitive in this respect, aligning with our earlier observation in Table 1 that the longest forward paths closely approximate shortest paths. MMFP paths are slightly longer than randomly sampled forward paths, reflecting a trade-off between fairness and path length. Yen's algorithm, focused on finding the k s-t-paths as short as possible, naturally yields the shortest average lengths. OP+ and DKSP still produce quite short paths. By contrast, those returned by ESX-C are longer; the length of its paths increases as the similarity threshold θ decreases and the number k of requested paths increases.

Runtime analysis. The third column of Tables 4-10 reports the runtime in seconds. Our method is quite competitive and scales well with larger networks. It is worth highlighting that, on the commodity laptop we use for our experiments, computing the maxmin-fair distribution takes on average 150 seconds in a network with 3.6M nodes and 8.7M edges.

Unsurprisingly, random forward paths are faster to compute than MMFP. As for the other methods, their runtime is highly dependent on the number of paths k requested which, unlike our method, needs to be specified in advance. Requesting around k = 10 paths is reasonably fast for all the baselines, except for very large road networks (Table 7 and 10). However, asking for a larger k was computationally expensive for a single source-destination pair, even for small networks. This was particularly noticeable for OnePass+ and, to a lesser degree, for Yen and DKSP. Although ESX-C is able to produce a higher number of different paths k, it becomes prohibitively slow on large networks.

In Figure 5 (Right), we investigate further the relationship between the number of requested paths and the runtime for the Florence dataset (for larger datasets, we couldn't run several baselines

Table 4: Essaouira, Morocco. We report the Gini coefficient, path lengths, and runtime averaged over 100 random source-destination pairs. The maximum computational time allowed per pair is 1000 seconds. For DAG and MMFP (Algorithms 1 and 2), Gini and the expected length are computed exactly from the distribution; for the rest, they refer to the empirical average over the k paths produced.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.27 ± 0.13	3936 ± 3664	0.48 ± 0.81
Random FP	0.36 ± 0.19	3822 ± 3582	0.05 ± 0.01
Yen	0.48 ± 0.22	3692 ± 3485	0.27 ± 0.29
OP+	0.43 ± 0.16	4040 ± 3831	0.07 ± 0.38
DKSP	0.46 ± 0.20	3704 ± 3562	3.70 ± 25.47
ESX-C θ=0.5 k=10	0.47 ± 0.13	5537 ± 4500	0.03 ± 0.02
ESX-C θ=0.8 k=10	0.50 ± 0.12	6055 ± 5863	0.04 ± 0.03
ESX-C θ=0.5 k=100	0.46 ± 0.17	6089 ± 3618	0.18 ± 0.17
ESX-C θ=0.8 k=100	0.50 ± 0.16	9871 ± 4882	1.03 ± 1.08

Table 5: Florence, Italy. Same as in Table 4.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.18 ± 0.10	7233 ± 3859	0.59 ± 0.42
Random FP	0.20 ± 0.12	7155 ± 3791	0.28 ± 0.04
Yen	0.29 ± 0.17	6953 ± 3598	0.97 ± 0.95
OP+	0.32 ± 0.09	7301 ± 3572	0.82 ± 4.02
DKSP	0.29 ± 0.12	6996 ± 3171	0.43 ± 1.17
ESX-C θ=0.5 k=10	0.46 ± 0.07	13985 ± 4830	0.17 ± 0.18
ESX-C θ=0.8 k=10	0.46 ± 0.06	9517 ± 3671	0.03 ± 0.01
ESX-C θ=0.5 k=100	0.55 ± 0.15	29075 ± 10320	2.75 ± 2.03
ESX-C θ=0.8 k=100	0.62 ± 0.10	37191 ± 8903	30.6 ± 15.3

Table 6: Kyoto, Japan. Same as in Table 4.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.40 ± 0.12	9214 ± 4973	12.66 ± 15.19
Random FP	0.56 ± 0.19	8931 ± 4806	2.18 ± 0.15
Yen	0.66 ± 0.20	8510 ± 4589	35.88 ± 45.45
OP+	0.61 ± 0.18	8600 ± 4599	1.12 ± 3.10
DKSP	0.64 ± 0.20	8533 ± 4672	1.53 ± 5.96
ESX-C θ=0.5 k=10	0.57 ± 0.10	9883 ± 4892	0.31 ± 0.39
ESX-C θ=0.8 k=10	0.61 ± 0.13	9066 ± 4681	0.10 ± 0.02
ESX-C θ=0.5 k=100	0.63 ± 0.12	25868 ± 12884	198.4 ± 179.6
ESX-C θ=0.8 k=100	0.66 ± 0.09	17015 ± 6365	15.08 ± 44.80

Table 7: Eastern USA (Q1). Same as in Table 4.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.37 ± 0.11	(1.60 ± 0.24) $\cdot 10^6$	149.8 ± 83.0
Random FP	0.45 ± 0.14	(1.58 ± 0.24) $\cdot 10^6$	67.5 ± 6.0
Yen k=5	0.51 ± 0.17	(1.51 ± 0.25) $\cdot 10^6$	641.9 ± 476.8
OP+ k=5	0.53 ± 0.14	(1.54 ± 0.24) $\cdot 10^6$	229.9 ± 622.3
DKSP k=5	-	-	$> 10^3$
ESX-C θ=0.5 k=10	0.57 ± 0.08	(1.68 ± 0.26) $\cdot 10^6$	9.5 ± 9.1
ESX-C θ=0.8 k=10	0.59 ± 0.10	(1.60 ± 0.26) $\cdot 10^6$	4.5 ± 1.3
ESX-C θ=0.5 k=100	-	-	$> 10^3$
ESX-C θ=0.8 k=100	-	-	$> 10^3$

Table 8: Piedmont, California, USA. Same as in Table 4.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.20 ± 0.11	2038 ± 1004	0.29 ± 0.24
Random FP	0.26 ± 0.14	2034 ± 1001	0.02 ± 0.03
Yen	0.34 ± 0.17	1958 ± 855	0.04 ± 0.03
OP+	0.34 ± 0.13	2125 ± 835	0.02 ± 0.00
DKSP	0.34 ± 0.16	2004 ± 856	0.02 ± 0.00
ESX-C θ=0.5 k=10	0.38 ± 0.09	4395 ± 1739	0.02 ± 0.00
ESX-C θ=0.8 k=10	0.41 ± 0.09	3447 ± 1195	0.02 ± 0.00
ESX-C θ=0.5 k=100	0.30 ± 0.11	3315 ± 1354	0.05 ± 0.02
ESX-C θ=0.8 k=100	0.30 ± 0.10	4264 ± 1843	0.08 ± 0.04

Table 9: Buenos Aires, Argentina. Same as in Table 4.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.39 ± 0.13	8824 ± 3996	4.14 ± 5.81
Random FP	0.53 ± 0.18	8668 ± 3915	0.57 ± 0.15
Yen	0.62 ± 0.21	8364 ± 3747	5.44 ± 6.20
OP+	0.57 ± 0.15	8476 ± 3734	0.72 ± 3.76
DKSP	0.61 ± 0.19	8392 ± 3748	0.05 ± 0.04
ESX-C θ=0.5 k=10	0.54 ± 0.10	9643 ± 3776	0.07 ± 0.05
ESX-C θ=0.8 k=10	0.58 ± 0.11	9098 ± 3814	0.06 ± 0.01
ESX-C θ=0.5 k=100	0.64 ± 0.07	21793 ± 6057	26.64 ± 28.35
ESX-C θ=0.8 k=100	0.67 ± 0.05	22593 ± 5751	3.45 ± 2.97

Table 10: Florida, USA. Same as in Table 4.

Method	Gini	Length	Runtime (s)
DAG & MMFP	0.43 ± 0.12	(4.22 ± 2.01) $\cdot 10^6$	232.1 ± 213.6
Random FP	0.50 ± 0.17	(4.19 ± 2.00) $\cdot 10^6$	14.2 ± 2.0
Yen k=5	-	-	$> 10^3$
OP+ k=5	-	-	$> 10^3$
DKSP k=5	-	-	$> 10^3$
ESX-C θ=0.5 k=10	0.64 ± 0.08	(4.49 ± 2.01) $\cdot 10^6$	61.3 ± 64.9
ESX-C θ=0.8 k=10	0.66 ± 0.08	(4.28 ± 2.00) $\cdot 10^6$	3.2 ± 1.6
ESX-C θ=0.5 k=100	-	-	$> 10^3$
ESX-C θ=0.8 k=100	-	-	$> 10^3$

for k larger than 5). Increasing the number of paths per point-to-point query, the baseline methods become slower when several paths are requested, while MMFP is hardly affected. These observations highlight another advantage of our method: by constructing a distribution over multiple distinct paths, we eliminate the need to pre-specify the number of paths required. This approach enables us to efficiently generate a larger set of paths without the computational overhead associated with increasing k in other methods.

5.2 Ablation study and scalability

We analyze the scalability of our method by breaking it into three components: DAG construction (Algorithm 1), maxmin-fair distribution computation (Algorithm 2), and path sampling.

Figure 6 (Left) shows the average runtime of each of the three components across the seven datasets. As expected, MMFP is the most computationally intensive. However, the linear trend in the log-log plot suggests polynomial growth, consistent with Sections 3 and 4. Figure 6 (Center) plots runtime against DAG size for random source-target pairs from the Kyoto dataset. For each pair, we build the DAG and report its number of nodes on the horizontal axis. MMFP shows sublinear growth in the semi-log plot, aligning with theoretical expectations. DAG construction time depends mainly on the input graph size, not the (smaller) output DAG size, as Algorithm 1 relies on distance computations from sources to all nodes and from all nodes to the target. Path sampling time increases only slightly with DAG size.

Finally, Figure 6 (Right) reports peak memory usage. We can observe that MMFP's LP optimization uses minimal memory. DAG construction memory usage is comparable to that of storing the original graph, as Dijkstra's algorithm operates on it directly.

6 CONCLUSIONS

To the best of our knowledge, our work is the first to address the problem of individual fairness towards the nodes of a network in route recommendation. We introduce the concept of forward paths

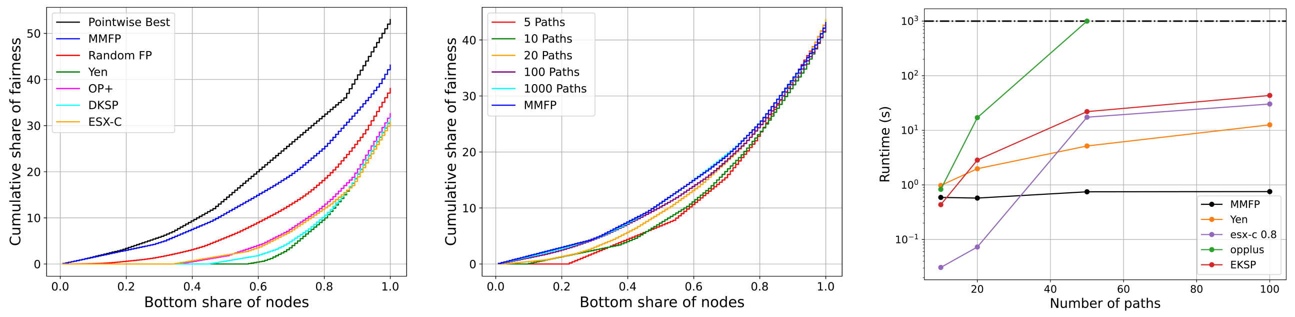


Figure 5: (Left) and (Center) Generalized Lorenz curve for a source-destination random pair of the Essaouira dataset. The horizontal axis reports the cumulative fraction of nodes that are receiving the least amount of satisfaction, while the vertical axis contains the cumulative amount of satisfaction received by the corresponding bottom share of nodes. On the left, the curve is depicted for different methods and Pointwise Best (black) represents an idealistic, in general unattainable, curve that maximize the cumulative share of fairness, for each fixed the bottom share of nodes. In the center, as more paths are sampled from the maxmin-fair distribution, the observed fairness allocation increasingly converges to the one of the maxmin-fair distribution. (Right) Average runtime w.r.t. number of requested paths per point-to-point query, (Florence dataset). Baseline methods becomes slower when several paths are requested. The time complexity of MMFP is due to the optimization of the maxmin-fair distribution; sampling several paths does not increases its runtime significantly.

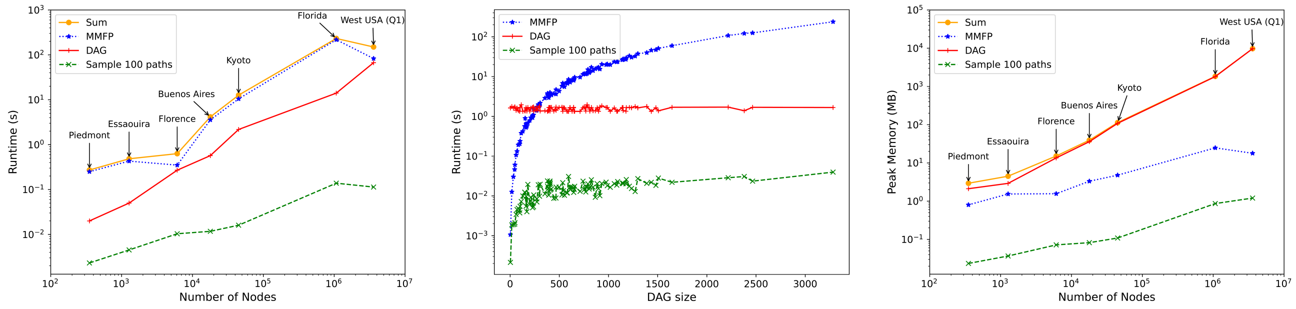


Figure 6: (Left) Average runtime over 100 random source-target pairs for each component of our method, as well as their total sum, across datasets of varying sizes. (Center) Runtime for each component of our method for the Kyoto dataset. Each point corresponds to a source-target pair, with the number of nodes in the corresponding DAG reported on the horizontal axis. The sub-linear growth in the semi-log plot of MMFP corresponds to the theoretical polynomial runtime. The constant runtime of the DAG creation is in accordance with the fact that the cost to construct the DAG is asymptotically the same as Dijkstra's algorithm, which mainly depends on the size of the original graph. (Right) The Peak Memory allocated, averaged across multiple source-target pairs. The memory allocated by MMFP is very low, as the LPs iteratively solved are small.

and show their viability as an alternative to shortest paths. We show how to compute a DAG that contains all the forward paths and how a probability distribution on the forward paths, such that the individual nodes receive a maxmin-fair individual satisfaction, can be obtained by solving a flow problem by linear programming. Our experimental results confirm the benefits and practical relevance of our proposal.

One limitation of our work is that even though forward paths tend to be short in practice, this property is not guaranteed. The degree to which forward paths approximate shortest paths is data-dependent, whereas for some applications a strict pre-specified upper bound on the deviation from the optimal path length may be desirable. In this regard, we note that it is straightforward to modify LP (1) to include a constraint on the *expected* length of the forward paths found, which is given by $\sum f_{u,v} \cdot \ell(u,v)$; and by using rejection sampling, we can constrain the *maximum* path length to be at most $1+\alpha$ times its expectation by sampling $O(1/\alpha)$ paths (in expectation) and rejecting those that are too long. However, such a modification would not conform to the strict definition of maxmin-fairness. Reconciling maxmin-fairness with hard bounds on path lengths remains an open problem for future research.

We have focused on fairness from a locational perspective, assuming users aim to reach their destination in a reasonable time. However, users may have diverse preferences: for example, a tourist might prefer routes with museums, while another might favor architecturally significant sites. Incorporating such personalized preferences into our fairness framework is a compelling challenge for future work. Additionally, our method assumes static graphs; extending it to dynamic networks or incorporating evolving edge weights would be valuable directions for future research.

REFERENCES

[1] Ittai Abraham, Daniel Delling, Andrew V Goldberg, and Renato F Werneck. 2013. Alternative routes in road networks. *Journal of Experimental Algorithmics (JEA)* 18 (2013), 1–1.

[2] Richard Bellman. 1958. On a routing problem. *Quarterly of applied mathematics* 16, 1 (1958), 87–90.

[3] Matthias Bentert, Leon Kellerhals, and Rolf Niedermeier. 2022. The structural complexity landscape of finding balance-fair shortest paths. *Theoretical Computer Science* 933 (2022), 149–162.

[4] Matthias Bentert, Leon Kellerhals, and Rolf Niedermeier. 2023. Fair short paths in vertex-colored graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12346–12354.

[5] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*. 405–414.

[6] Geoff Boeing. 2025. Modeling and analyzing urban networks and amenities with OSMnx. *Geographical Analysis* (2025).

[7] Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2013. Where shall we go today? Planning touristic tours with TripBuilder. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 757–762.

[8] Xin Cao, Lisi Chen, Gao Cong, and Xiaokui Xiao. 2012. Keyword-aware Optimal Route Search. *Proc. VLDB Endow.* 5, 11 (2012), 1136–1147.

[9] Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, Ulf Leser, and David B Blumenthal. 2020. Finding k-shortest paths with limited overlap. *The VLDB Journal* 29, 5 (2020), 1023–1047.

[10] Michael B Cohen, Yin Tat Lee, and Zhao Song. 2021. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)* 68, 1 (2021), 1–39.

[11] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[12] Yashar Deldjoo, Dietmar Jannach, Alejandro Bellogin, Alessandro Difonzo, and Dario Zanzonelli. 2024. Fairness in recommender systems: research landscape and future directions. *User Modeling and User-Adapted Interaction* 34, 1 (2024), 59–108.

[13] Camil Demetrescu, Andrew V Goldberg, and David S Johnson. 2009. *The shortest path problem: Ninth DIMACS implementation challenge*. Vol. 74. American Mathematical Soc.

[14] Edsger W. Dijkstra. 1959. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* 50 (1959), 269–271.

[15] Ahmed Fahmin, Bojie Shen, Muhammad Aamir Cheema, Adel N. Toosi, and Mohammed Eunus Ali. 2024. Efficient alternative route planning in road networks. *IEEE Transactions on Intelligent Transportation Systems* 25, 10 (Oct. 2024), 14220–14232.

[16] Michael L Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)* 34, 3 (1987), 596–615.

[17] David García-Soriano and Francesco Bonchi. 2020. Fair-by-design matching. *Data Mining and Knowledge Discovery* 34 (2020), 1291–1335.

[18] David García-Soriano and Francesco Bonchi. 2021. Maxmin-fair ranking: individual fairness under group-fairness constraints. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 436–446.

[19] Corrado Gini. 1936. On the measure of concentration with special reference to income and statistics, Colorado College Publication. *General series* 208, 1 (1936).

[20] Ashish Goel and Adam Meyerson. 2006. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica* 44 (2006), 301–323.

[21] Martin Grötschel, László Lovász, and Alexander Schrijver. 1988. *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics, Vol. 2. Springer.

[22] Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[23] Christian Häcker, Panagiotis Bouros, Theodoros Chondrogiannis, and Ernst Althaus. 2021. Most diverse near-shortest paths. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 229–239.

[24] Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. 2023. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 3968–3976.

[25] Tesshu Hanaka, Yasuaki Kobayashi, Kazuhiro Kurita, See Woo Lee, and Yota Otachi. 2022. Computing diverse shortest paths efficiently: A theoretical and experimental study. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 3758–3766.

[26] Christopher Hojny, Frits Spieksma, and Sten Wessel. 2025. Fairness in graph-theoretical optimization problems. *Discrete Applied Mathematics* 375 (2025), 178–192.

[27] Chen Karako and Putra Manggala. 2018. Using image fairness representations in diversity-based re-ranking for recommendations. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*. 23–28.

[28] Shan Liu, Hai Jiang, Shuiping Chen, Jing Ye, Renqing He, and Zhizhao Sun. 2020. Integrating Dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning. *Transportation Research Part E: Logistics and Transportation Review* 142, Article 102070 (Oct. 2020), 102070 pages.

[29] Max O Lorenz. 1905. Methods of measuring the concentration of wealth. *Publications of the American statistical association* 9, 70 (1905), 209–219.

[30] Zihan Luo, Lei Li, Mengxuan Zhang, Wen Hua, Yehong Xu, and Xiaofang Zhou. 2022. Diversified Top-k Route Planning in Road Network. *Proc. VLDB Endow.* 15, 11 (2022), 3199–3212.

[31] Amgad Madkour, Walid G Aref, Faizan Ur Rehman, Mohamed Abdur Rahman, and Saleh Basalamah. 2017. A survey of shortest-path algorithms. *arXiv preprint arXiv:1705.02044* (2017).

[32] Atasi Panda, Anand Louis, and Prajakta Nimbhorkar. 2024. Individual Fairness under Group Fairness Constraints in Bipartite Matching - One Framework to Approximate Them All. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024*. 175–183.

[33] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. 2014. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *Proceedings of the 25th ACM conference on Hypertext and social media*. 116–125.

[34] John Rawls. 1971. *A theory of justice*. MA: Harvard University Press.

[35] Senjuti Basu Roy, Gautam Das, Sihem Amer-Yahia, and Cong Yu. 2011. Interactive itinerary planning. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 15–26.

[36] Jad Salem, Reuben Tate, and Stephan Eidenbenz. 2024. Expected Maximin Fairness in Max-Cut and other Combinatorial Optimization Problems. *arXiv preprint arXiv:2410.02589* (2024).

[37] Joy Lal Sarkar, Abhishek Majumder, Chhabi Rani Panigrahi, Sudipta Roy, and Bibudhendu Pati. 2023. Tourism recommendation system: A survey and future research directions. *Multimedia tools and applications* 82, 6 (2023), 8983–9027.

[38] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of Exposure in Rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). 2219–2228.

[39] Panote Siriaraya, Yuanyuan Wang, Yihong Zhang, Shoko Wakamiya, Péter Jeszenszky, Yukiko Kawai, and Adam Jatowt. 2020. Beyond the Shortest Route: A Survey on Quality-Aware Route Navigation for Pedestrians. *IEEE Access* 8 (2020), 135569–135590.

[40] Christian Sommer. 2014. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 1–31.

[41] Caleb Voss, Mark Moll, and Lydia E Kavraki. 2015. A heuristic approach to finding diverse short paths. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4173–4179.

[42] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. 2023. A Survey on the Fairness of Recommender Systems. *ACM Trans. Inf. Syst.* 41, 3 (2023).

[43] Kexin Xie, Ke Deng, Shuo Shang, Xiaofang Zhou, and Kai Zheng. 2012. Finding alternative shortest paths in spatial networks. *ACM Transactions on Database Systems (TODS)* 37, 4 (2012), 1–31.

[44] Jin Y Yen. 1970. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of applied mathematics* 27, 4 (1970), 526–530.

[45] Shiming Zhang, Zhipeng Luo, Li Yang, Fei Teng, and Tianrui Li. 2024. A survey of route recommendations: Methods, applications, and opportunities. *Inf. Fusion* 108 (2024), 102413.