One-Shot Traffic Assignment with Forward-Looking Penalization

Giuliano Cornacchia
University of Pisa and ISTI-CNR
Pisa, Italy
giuliano.cornacchia@phd.unipi.it

Mirco Nanni
ISTI-CNR
Pisa, Italy
mirco.nanni@isti.cnr.it

Luca Pappalardo
ISTI-CNR
Pisa, Italy
luca.pappalardo@isti.cnr.it

ABSTRACT

Traffic assignment (TA) is crucial in optimizing transportation systems and consists in efficiently assigning routes to a collection of trips. Existing TA algorithms often do not adequately consider real-time traffic conditions, resulting in inefficient route assignments. This paper introduces Metis, a coordinated, one-shot TA algorithm that combines alternative routing with edge penalization and informed route scoring. We conduct experiments in several cities to evaluate the performance of Metis against state-of-the-art one-shot methods. Compared to the best baseline, Metis significantly reduces CO2 emissions by 18% in Milan, 28% in Florence, and 46% in Rome, improving trip distribution considerably while still having low computational time. Our study proposes Metis as a promising solution for optimizing TA and urban transportation systems.

CCS CONCEPTS

• **Information systems → Geographic information systems**; • **Computing methodologies** → *Agent / discrete models*; • **Applied computing → Transportation**.

KEYWORDS

traffic assignment, alternative routing, route planning, path diversification, CO2 emissions, urban sustainability

ACM Reference Format:
Giuliano Cornacchia, Mirco Nanni, and Luca Pappalardo. 2023. One-Shot Traffic Assignment with Forward-Looking Penalization. In *The 31st ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '23), November 13–16, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3589132.3625637

1 INTRODUCTION

Traffic Assignment (TA) has emerged as a crucial problem today due to the rapid growth of urbanization and increasing traffic congestion [2, 5, 6, 21, 24, 30, 32]. As cities expand and populations rise, transportation networks face pressure to efficiently accommodate the growing demand for mobility. Efficient TA plays a pivotal role in achieving several Sustainable Development Goals (SDGs) set by the United Nations [28], promoting effective traffic management and reducing greenhouse gas emissions.

Existing approaches to TA can be broadly classified into one-shot and iterative approaches. One-shot approaches assign routes to a collection of trips without any additional optimization [5, 6], while iterative approaches involve multiple iterations to improve efficacy [2, 21, 32]. However, these approaches predominantly rely on basic road network information and travel times, failing to harness the potential of more sophisticated measures based on human mobility patterns. As a result, there is ample opportunity for further advancements of TA solutions to enhance their effectiveness.

In contrast to one-shot and iterative approaches, alternative routing (AR) ones adopt an individualistic approach. They focus on providing alternative routes to individual users, aiming to strike a balance between proximity to the fastest route and route diversity [1, 7–9, 11, 14, 17, 18, 20, 26, 34]. However, their individualistic nature overlooks vehicle interactions, leading to suboptimal outcomes at the collective level. As a result, they often lead to increased congestion and a higher environmental impact.

To overcome these limitations, we propose Metis, a novel coordinated approach that improves TA by incorporating alternative routing, edge penalization, and informed route scoring. Metis introduces some key innovations. Firstly, Metis estimates vehicles' current position to penalize road edges expected to be traversed, discouraging future vehicles from using those congested edges. Secondly, Metis generates alternative routes using the penalized road network and assigns them to individual trips favouring unpopular routes with high-capacity roads. These innovative components enable Metis to promote a more balanced distribution of traffic, improving the efficiency of TA and providing drivers with fast paths while addressing the limitations of existing approaches.

Through comprehensive experiments conducted in three cities, where we compare Metis with various state-of-the-art one-shot approaches, we highlight its superior performance in optimizing routing and while maintaining computational efficiency. Notably, Metis significantly reduces total CO2 emissions compared to the best baseline, ranging from 18% to 46%, depending on the city.

Metis is a significant step forward in TA, offering a dynamic approach to provide drivers with efficient routes, alleviating urban congestion. In summary, the key contributions of the paper are:

- We introduce Forward-Looking Edge Penalization (FLEP) to estimate vehicles' current positions and penalize road edges that are expected to be traversed (Section 3.2);
- We integrate AR into TA, showing how generating alternative routes may improve traffic assignment (Section 3.3);
- We introduce a pattern-based route scoring to discourage the selection of popular, congested routes (Section 3.4).
- We conduct extensive experiments to demonstrate Metis' superior performance over existing AR and one-shot approaches in reducing CO2 emissions while maintaining competitive computational performance (Section 5).

SIGSPATIAL '23, November 13–16, 2023, Hamburg, Germany

ACM ISBN 979-8-4007-0168-9/23/11.
https://doi.org/10.1145/3589132.3625637

Open Source. The code that implements METIS, the baselines, and the experiments can be accessed at https://bit.ly/metis_ta.

2 RELATED WORK

Traffic assignment (TA) consists in allocating vehicle trips on a road network to minimize congestion and travel time [2, 5, 6, 21, 30, 32]. We group TA solutions into individual approaches, providing a route to a single trip, and collective approaches, providing a set of routes for an entire collection of trips.

Individual approaches

The fastest path is the most straightforward approach to connect two locations in a road network [33]. However, from a collective point of view, aggregating all individual fastest paths may increase congestion and CO_2 emissions [10, 23].

Several works focus on alternative routing (AR) to distribute the vehicles more evenly on the road network [17]. In particular, the k-shortest path problem [1, 34] aims to find the k shortest paths between an origin and a destination. In practical scenarios, k-shortest path solutions fail to provide significant path diversification, as the generated paths exhibit a 99% overlap in terms of road edges [7]. The k-shortest disjointed paths problem [26] focuses on identifying k paths that do not overlap. Solutions to this problem often result in routes that significantly deviate from the optimal path, leading to a notable increase in travel time. Several approaches lie between the k-shortest path and k-shortest disjoint paths, which can be divided into edge weight, plateau, and dissimilarity approaches.

Edge weight approaches. They compute the shortest paths iteratively. At each iteration, they update the edge weights of the road network to compute k alternative paths. Edge weight updating may consist of a randomization of the weights or a cumulative penalization of the edges contributing to the shortest paths. Although easy-to-implement, edge-weight approaches do not guarantee the generation of paths considerably different from each other [17].

Plateau approaches. They build two shortest-path trees, one from the source and one from the destination, and identify their common branches, known as plateaus [20]. The top-k plateaus are selected based on their lengths, and alternative paths are generated by appending the shortest paths from the source to the first edge of the plateau and from the last edge to the target. As the plateaus are inherently disjointed, they may create significantly longer routes than the fastest path [20].

Dissimilarity approaches. They generate k paths that satisfy a dissimilarity constraint and a desired property. Liu et al. [18] propose the k-Shortest Paths with Diversity (kSPD) problem, defined as top-k shortest paths that are the most dissimilar with each other and minimize the paths' total length. Chondrogiannis et al. [8] propose an implementation of the k-Shortest Paths with Limited Overlap (kSPLO), seeking to recommend k-alternative paths that are as short as possible and sufficiently dissimilar. Chondrogiannis et al. [9] formalize the k-Dissimilar Paths with Minimum Collective Length (kDPML) problem where, given two road edges, they compute a set of k paths containing sufficiently dissimilar routes and the lowest collective path length. Hacker et al. [14] propose k-Most Diverse Near Shortest Paths (KMD) to recommend the set of k near-shortest paths (based on a user-defined cost threshold) with the highest diversity (lowest pairwise dissimilarity). Dissimilarity approaches do not guarantee that a set of k paths exists that satisfies the desired property.

Collective approaches

Collective approaches consider the impact of traffic in a collective environment where vehicles interact. There are two main categories of collective approaches: one-shot and iterative methods.

One-shot methods. They assign a route to each trip without further optimizing the routes. They are computationally efficient and provide a quick, yet not optimal, traffic allocation. The simplest one-shot method is the All-Or-Nothing assignment (AON) [5], in which each trip is assigned to the fastest path between the trip's origin and destination, considering the free-flow travel time.

Incremental Traffic Assignment (ITA) [6] extends AON incorporating the dynamic travel time changes within a road edge. ITA splits the mobility demand into n splits of a specified percentage (n=4 with 40%, 30%, 20%, and 10% are common values [29]). The trips in the first split are assigned using AON and each edge's travel time is updated using the function proposed by the Bureau of Public Roads (BPR) [4]. The trips in the second split are assigned using AON, considering the updated travel time. Iteratively, ITA assigns the trips in each split, updating the travel time at each iteration.

Iterative methods. Iterative approaches employ multiple iterations to compute TA until a convergence criterion is satisfied. While these approaches can be computationally demanding, they offer the advantage of yielding the optimal solution once convergence is achieved. Two main iterative approaches are the user equilibrium (UE) and the system optimum (SO).

UE is based on the Wardrop principle [32], which states that no individual driver can unilaterally improve their travel time by changing their route. In UE, each individual selfishly selects the most convenient path, and all the unused paths will have a travel time greater than the selected route. UE assumes that drivers are rational and have perfect network knowledge [21]. However, a system in user equilibrium does not imply that the total travel time is minimized [22]. Dynamic User Equilibrium (DUE) [13] approximates the user equilibrium by performing simulations to estimate travel times more accurately.

In contrast with UE, SO is based on Wardrop's second principle, which suggests drivers cooperate to minimize the total system travel time [31]. In SO, drivers are considered selfless and willingly adhere to assigned routes to reduce congestion and travel time. Both UE and SO may be solved using an iterative algorithm for optimization. Beckmann et al. [2] provide the mathematical models for the traffic assignment as a convex non-linear optimization problem with linear constraints that may be solved through an iterative algorithm to solve the quadratic optimization problems [12].

One-shot methods are faster than iterative ones but offer only an approximation of the solution. Therefore, the choice between these approaches depends on the specific requirements of the problem, balancing accuracy with computational efficiency.

Position of our Work. METIS is a one-shot, coordinated approach that effectively and quickly solves TA by balancing environmental concerns and drivers' needs.

3 METIS

The idea behind METIS is to shift from an individualistic paradigm to a collective, coordinated one.[1] In contrast with existing AR algorithms, METIS acts as a central unit that provides drivers with suggested routes considering dynamic estimation of traffic conditions. METIS estimates vehicles' current positions to penalize edges expected to be traversed, thus avoiding congested edges. Moreover, METIS incorporates a pattern-based choice criterion that discourages the selection of popular routes likely to be chosen by other drivers. By doing so, METIS optimizes the routing process and provides drivers with efficient routes that minimize travel time and alleviate traffic congestion.

Algorithm 1 presents METIS' high-level pseudocode. It takes four inputs: *(i)* a mobility demand D, i.e., a time-ordered collection of trips, each represented by its origin o, destination d, and departure time t; *(ii)* a directed weighted graph $G = (V, E)$, representing the road network, where V is the set of intersections and E the set of road edges, each associated with the expected travel time estimated as its length divided by the maximum speed allowed; *(iii)* a parameter $p > 0$, which controls to what extent to penalize crowded edges; *(iv)* a slowdown parameter $s \geq 1$ accounting for reduced speeds on edges due to the presence of other vehicles and various events like traffic lights.

Algorithm 1: METIS

```
Input  : road network G, mobility demand D, penalization
         parameter p, slowdown parameter s
Output : assigned routes R
   // Initialization Phase
1  K_road^(source), K_road^(end) ← KRoadEstimation(G, D);
2  R ← ∅;
   // Perform the Traffic Assignment (TA)
3  foreach j = (o, d, t) ∈ D do
       // Apply the Forward-Looking Edge Penalization (FLEP)
4      H ← FLEP(G, R, p, s, t);
       // Generate k candidates on the penalized road network
5      P ← KMD(H, o, d);
       // Select the route that minimizes the route scoring function
6      r ← RouteSelection(P, K_road^(source), K_road^(end));
       // Update the assigned routes collection
7      R ← R ∪ {r};
8  return R;
```

The algorithm starts with the initialization phase (lines 1-2 of Algorithm 1), where it computes two K_{road}-based measures. Subsequently, the algorithm performs the traffic assignment (lines 3-7): for each trip $j \in D$, METIS employs FLEP (Forward-Looking Edge Penalization) to penalize edges based on other vehicles' estimated current position, thus producing a penalized road network H (line 4). Then, METIS employs KMD [14] to generate a set P of k alternative routes between each trip's origin o and destination d, based on H (line 5). Then, the algorithm assigns to the trip j the route r with the minimum value of a route scoring function (line 6), adding it to the routes collection R (line 7). Once each trip in D has been associated with a route, METIS returns R (line 8).

The following sections provide details on METIS' components. Section 3.1 outlines the initialization phase and introduces the K_{road}-based measures, Section 3.2 introduces FLEP, Section 3.3 describes KMD, and Section 3.4 describes route scoring.

3.1 Initialization Phase

During the initialization phase (line 1 of Algorithm 1), METIS computes $K_{\text{road}}^{(\text{source})}(e)$ and $K_{\text{road}}^{(\text{end})}(e)$ for every edge e in the road network. This computation requires a collection of routes to estimate sources and destinations of traffic on the road network. In contrast to the approach by Wang et al. [29], which utilizes real GPS data to compute K_{road} for each edge, we adopt a more adaptable strategy. We establish connections between origin and destination points in D with the fastest routes in the road network assuming free-flow travel time, enabling us to estimate the $K_{\text{road}}^{(\text{source})}$ and $K_{\text{road}}^{(\text{end})}$ values for each edge, even in situations where GPS data are unavailable.

K_{road} measures. The K_{road} of an edge quantifies how many areas of the city (e.g., neighbourhoods) contribute to most of the traffic flow over that edge [29]. The computation of K_{road} involves constructing a road usage network, which is a bipartite network where each road edge is connected to its major driver areas, i.e., those responsible for 80% of the traffic flow on that edge [29]. The $K_{\text{road}}(e)$ of an edge e is the degree of e within the road usage network. K_{road} indicates an edge's popularity: an edge with a low K_{road} is chosen by only a limited number of traffic sources, indicating relatively low popularity; an edge with a high K_{road} attracts traffic from more diverse areas, indicating higher popularity among them.

We expand upon the K_{road} concept by introducing $K_{\text{road}}^{(\text{source})}$ and $K_{\text{road}}^{(\text{end})}$ as follows. First, an area A is a driver source for an edge e if at least one vehicle originating from A travels through e. Similarly, A is a driver destination for e if at least one vehicle traverses e and completes its trip in A. An area can be both driver source and driver destination for a particular edge. In this work, an area is a square tile of 1 km within a square tessellation of the city.

We define the major driver sources (MDS) and the major driver destinations (MDD) as the areas to which 80% of the traffic flowing through an edge starts or ends, respectively. To calculate these two measures, we construct a bipartite network where a connection is established from an area A to an edge e if A is an MDS for e. Similarly, a connection is formed from an edge e to an area A if A is an MDD for e. Specifically, for an edge e, $K_{\text{road}}^{(\text{source})}(e)$ is the in-degree of e within the bipartite network, while $K_{\text{road}}^{(\text{end})}(e)$ is e's out-degree.

[1] The name METIS has been inspired by the Greek goddess who personifies wisdom, cunning, strategy, and prudence.

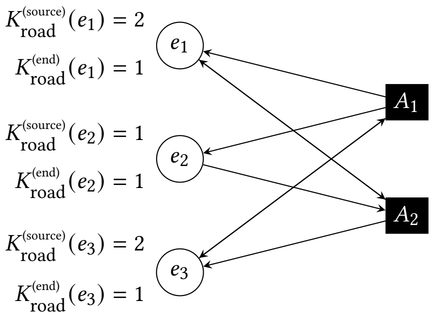


Figure 1: Graphical representation of the bipartite network of road edges and areas. $K_{\text{road}}^{(\text{source})}$ is the in-degree of edge nodes, $K_{\text{road}}^{(\text{end})}$ is the out-degree of edge nodes.

We also define $K_{\text{route}}^{(\text{source})}(r)$ of a route $r = (e_1, \cdots, e_n)$ as the average $K_{\text{road}}^{(\text{source})}$ computed over its edges weighted with edge length:

$$K_{\text{route}}^{(\text{source})}(r) = \frac{\sum_{i=1}^{n} K_{\text{road}}^{(\text{source})}(e_i) \cdot l(e_i)}{\sum_{i=1}^{n} l(e_i)} \quad (1)$$

where $l(e_i)$ is the length of edge e_i. Similarly:

$$K_{\text{route}}^{(\text{end})}(r) = \frac{\sum_{i=1}^{n} K_{\text{road}}^{(\text{end})}(e_i) \cdot l(e_i)}{\sum_{i=1}^{n} l(e_i)} \quad (2)$$

Example. Figure 1 illustrates the concepts of $K_{\text{road}}^{(\text{source})}$ and $K_{\text{road}}^{(\text{end})}$ with three edges (e_1, e_2, e_3, circles) connected with two areas (A_1, A_2, squares). Let us consider edge e_1: it has one outgoing connection towards A_2, leading to an out-degree of 1, and thus $K_{\text{road}}^{(\text{end})}(e_1) = 1$. Moreover, edge e_1 has incoming connections from areas A_1 and A_2, resulting in an in-degree of 2 and, consequently, $K_{\text{road}}^{(\text{source})}(e_1) = 2$.

3.2 Forward-Looking Edge Penalization

Forward-Looking Edge Penalization (FLEP) is based on penalizing road edges to reflect the dynamic changes in travel time caused by increasing traffic volume.

Generally, existing methods penalize the entire routes assigned to currently travelling vehicles [6, 14, 18]. However, this indiscriminate penalization of all edges, including those currently unoccupied, may discourage the utilization of potentially efficient routes, leading to congestion in alternative paths that are not penalized.

FLEP overcomes this problem by estimating the current positions of vehicles in transit and penalizing the edges that these vehicles are projected to visit. Assuming that a vehicle departed t seconds ago, FLEP computes the distance it has travelled during t seconds, assuming that the vehicle travelled at a speed of max_speed/s on each edge, where s is a slowdown parameter accounting for reduced speeds on edges due to the presence of other vehicles and various events like traffic lights. Then, FLEP modifies the weights $w(e)$ assigned to the edges that the vehicle is expected to traverse by applying a penalty factor p: $w(e) \leftarrow w(e) \cdot (1+p)$. The penalization is cumulative, i.e., the edge is penalized for each vehicle that will traverse that edge. This penalization discourages the selection of edges that vehicles are likely to traverse, promoting alternative routes and a balanced distribution of traffic.

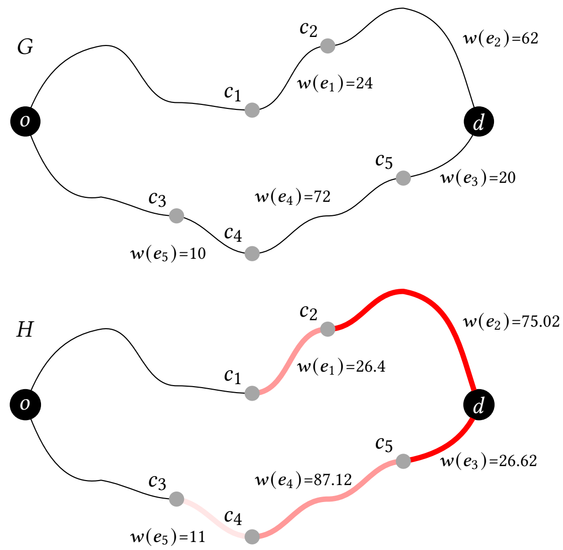


Figure 2: FLEP with $p = 0.1$ applied to road network G, resulting in penalized network H. Grey circles represent estimated vehicle positions. FLEP applies cumulative penalization to edges based on the vehicles' expected traversal, with a multiplicative factor of $(1 + p)$. Darker red color indicates higher penalties imposed on road edges. For example, edge e_4 is traversed by vehicles c_3 and c_4, leading to a penalty of $(1 + p)^2$, resulting in $w(e_4) = 72 \cdot (1.1)^2 = 87.12$.

Algorithm 2 provides the pseudocode of FLEP. First, FLEP considers each previously assigned route r and calculates the time Δt the vehicle spent travelling based on its departure time $t(r)$ and the current time t (line 2). Then, it computes the required travel time to reach each edge $e \in r$ using s (line 3). If the vehicle has yet to reach its destination (line 4), FLEP determines the index of the first unvisited edge in the route (line 5). Subsequently, it penalizes every unvisited edge in route r (lines 7-8). Finally, FLEP outputs the penalized network (line 9).

Example. Figure 2 illustrates how FLEP works, assuming a penalization $p = 0.1$. FLEP estimates the position of each vehicle in transit (grey circles) within the road network considering s. Subsequently, FLEP applies cumulative penalization to the edges that each vehicle will traverse to reach the destination d. This penalization is accomplished by multiplying the weights of these edges by $(1 + p)$ for each vehicle that will traverse it. For example, vehicles c_3 and c_4 are projected to pass through edge e_4. Consequently, the initial weight $w(e_4) = 72$ is penalized by $(1+p)^2$, resulting in a new weight of $w(e_4) = 72 \cdot (1.1)^2 = 87.12$. FLEP generates a modified road network H through this iterative process, penalizing edges according to the anticipated vehicle movements.

3.3 KMD

k-Most Diverse Near Shortest Paths (KMD) is an AR algorithm that generates a collection of k routes with the highest dissimilarity among each other while still adhering to a user-defined cost threshold ϵ [14]. As KMD becomes computationally challenging

Algorithm 2: Forward-Looking Edge Penalization (FLEP)

```
Input  : road network G, collection of assigned paths R,
         penalty parameter p, slowdown parameter s,
         current time t
Output : updated road network G
foreach r ∈ R do
    // Compute the vehicle's time spent travelling. t(r) is the
       departure time of r
    Δt = t − t(r);
    // list of travel times
    tt_list ← TravelTimesComputation(r, s);
    // Penalize unvisited edges
    if Δt < tt_list[len(r)] then
        i ← min{x ∈ [1, len(r)] | tt_list[x] > Δt};
        for j ∈ [i, ..., len(r)] do
            e ← r[j];
            w(e) ← w(e) · (1 + p);
// Return the road network with the penalized weights
return G;
Function TravelTimesComputation(r, s):
    tt_agg ← 0;
    l ← [];
    foreach e ∈ r do
        tt_agg ← tt_agg + w(e) · s ;
        l.append(tt_agg)
    return l;
```

for $k > 2$ due to its NP-hard nature, a penalization-based heuristic is commonly employed to accelerate the computation process [14].

Given an origin o and a destination d, KMD first calculates the fastest path between o and d. The cost c of the fastest path, along with the parameter ϵ, determines the maximum allowed cost threshold $c \cdot (1+\epsilon)$ for a path to be considered near-shortest. Next, KMD iteratively applies the penalization-based heuristic to compute a new near-shortest path p, which is then added to the set of near-shortest paths S. Subsequently, it generates all subsets of S composed of k elements. Among these subsets, KMD identifies the most diverse using the Jaccard coefficient, which compares the dissimilarity between pairs of paths. When no more near-shortest paths can be found using the penalization approach, KMD returns the subset of k paths with the highest diversity. The detailed pseudo-code of KMD is in [14].

Metis applies KMD on the road network penalized by FLEP to discourage routes passing pass through potentially congested edges. We use parameter values k=3 (three alternative routes) and ϵ=0.3 (maximum cost increase of 30% compared to the fastest path) for KMD, which are common values used in the literature [14, 17].

3.4 Route Selection

In the final step, Metis scores and ranks the set of alternative routes generated by KMD. To determine the best route among the alternatives, Metis assigns a score (the lower, the better) to each route r based on the following formula:

$$score(r) = \frac{K_{\text{road}}^{(\text{source})}(r) \cdot K_{\text{road}}^{(\text{end})}(r)}{C_r} \tag{3}$$

where C_r is the average of the capacities $C(e)$ of the edges in route r, taking into account the edge length. The capacity $C(e)$ of an edge e is computed as follows:

$$C(e) = \begin{cases} 1900 \cdot l \cdot q & \text{if } s_{\max} \leq 45 \\ (1000 + 20 \cdot s_{\max}) \cdot l & \text{if } 45 < s_{\max} < 60 \\ (1700 + 10 \cdot s_{\max}) \cdot l & \text{if } s_{\max} \geq 60 \end{cases} \tag{4}$$

where $s_{\max}$ is the speed limit associated with edge e (in miles/hour), l is the number of lanes in edge e, and $q = 0.5$ is the green time-to-cycle length ratio. The equation and the values above are taken from the 2000 Highway Capacity Manual [27, 29].

Route scoring combines two essential elements. In the denominator, the average capacity C_r favours routes composed mainly of high-capacity edges, which are expected to handle larger traffic volumes. In the numerator, the product $K_{\text{road}}^{(\text{source})}(r) \cdot K_{\text{road}}^{(\text{end})}(r)$ penalizes routes that predominantly consist of popular edges, promoting a balanced traffic distribution.

4 EXPERIMENTAL SETUP

This section describes the experimental settings (Section 4.1), an overview of the baselines we compare with Metis (Section 4.2), and the measures used for the comparison (Section 4.3).

4.1 Experimental Settings

We conduct experiments in three Italian cities: Milan, Rome, and Florence. These cities represented diverse urban environments with varying traffic dynamics, sizes, and road networks (Table 1).

Road Networks. We obtain a road network for each city using OSMWebWizard. The three cities' road network characteristics are heterogeneous (see Table 1). While the smallest city, Florence's road network exhibits the highest density (9.11). Milan and Rome are sparse compared to Florence, although they have extensive road networks. This difference in road network characteristics provides a valuable basis for evaluating the performance of TA algorithms in different urban contexts.

Mobility Demand. We divided each city into a grid of 1 km square tiles to discretize the geographical space. Then, we created realistic synthetic OD data that mirror real-world mobility patterns, maintaining the typical distribution of trip distances and the power-law behavior in the number of trips between two locations. These reference distributions were inferred from real GPS data samples in the three cities [3, 25]. We derive an origin-destination matrix M, where $m_{o,d}$ represents the number of trips starting in tile o and ending in tile d. To generate a mobility demand D of N trips, we randomly select a trip $T_v = (e_o, e_d)$ for a vehicle v, choosing matrix elements $m_{o,d}$ with probabilities $p_{o,d} \propto m_{o,d}$. We then select with random uniform probability two edges e_o and e_d within tiles o and d from the road network G. For our experiments, we set N = 10k trips in Florence, N = 20k trips in Rome, and N = 30k in Milan.

city	$\|V\|$	$\|E\|$	$L(E)$	area	den	# trips	N
Florence	6,140	11,804	1,050	115.28	9.11	4,076	10k
Milan	24,063	46,488	4,340	495.55	8.76	5,617	30k
Rome	31,798	63,384	6,569	788.11	8.34	7,622	20k

Table 1: Road network characteristics in the three cities. The columns show the number of vertices $|V|$ and edges $|E|$, the total road length $L(E)$, the area of the city, the ratio of road length to surface area (den, in km road/km^2), the total number of trips described by synthetic OD data, and the number N of routes generated in each city.

4.2 Baselines

We evaluate METIS against several one-shot TA solutions, both individual and collective. We exclude iterative solutions like User Equilibrium (UE) [13, 32] and System Optimum (SO) [31] from our analysis. While these iterative approaches may offer optimal results after convergence, their computationally intensive nature and multiple iterations make them unsuitable for real-time applications. Furthermore, we exclude navigation system services (e.g., Google Maps and TomTom) from our analysis because a direct comparison with METIS and the baselines under identical traffic conditions is unfeasible. Indeed, navigation services APIs recommend routes based on real-time, real-world traffic conditions, which inherently differ from those generated and considered within our simulation.

AR baselines. AR algorithms are designed to generate k alternative routes for an individual trip. We extend these algorithms to TA by aggregating the recommended routes for each trip within a mobility demand. In particular, we use an AR algorithm to compute $k = 3$ alternative routes for each trip in mobility demand D, and we randomly select one of them uniformly. In this study, we consider the following state-of-the-art methods:

- **PP (Path Penalization)** generates k alternative routes by penalizing the weights of edges contributing to the fastest path [7]. In each iteration, PP computes the fastest path and increases the weights of the edges that contributed to it by a factor p as $w(e) = w(e) \cdot (1 + p)$. The penalization is cumulative: if an edge has already been penalized in a previous iteration, its weight will be further increased [7].
- **GR (Graph Randomization)** generates k alternative paths by randomizing the weights of all edges in the road network before each fastest path computation. The randomization is done by adding a value from a normal distribution, given by the equation $N(0, w(e)^2 \cdot \delta^2)$ [7].
- **PR (Path Randomization)** generates k alternative paths randomizing only the weights of the edges that were part of the previously computed path. Similar to GR, it adds a value from a normal distribution to the edge weights, following the equation $N(0, w(e)^2 \cdot \delta^2)$ [7].
- **KD (k-shortest disjointed paths)** returns k alternative non-overlapping paths (i.e., with no common edges) [26].
- **PLA (Plateau)** builds two shortest-path trees, one from the origin and one from the destination, and identifies their common branches (plateaus) [20]. The top-k plateaus are selected based on their lengths, and alternative paths are generated by appending the fastest paths from the source to the plateau's first edge and from the last edge to the target.
- **KMD (k-Most Diverse Near Shortest Paths)** generates k alternative paths with the highest dissimilarity among each other while adhering to a user-defined cost threshold ϵ [14].

One-shot baselines. In contrast with AR approaches, one-shot (OS) ones assign a route to each trip of a mobility demand without further optimization on the assigned routes. In this study, we consider the two most common OS approaches:

- **AON (All-Or-Nothing)** assigns each trip to the fastest path between the trip's origin and destination, assuming free-flow travel times [5].
- **ITA (Incremental Traffic Assignment)** [6] uses four splits (40%, 30%, 20%, 10%, as recommended in the literature [29]) to assign routes to trips. In the first split, ITA uses AON considering free-flow travel time t_{free}. It then updates the travel times using the Bureau of Public Roads (BPR) function $t_a = t_{\text{free}} \cdot (1 + \alpha \cdot \text{VOC}^{\beta})$ [4], where VOC indicates an edge's traffic volume over its capacity and $\alpha = 0.15$ and $\beta = 4$ are values recommended in the literature [22, 29]. This process is repeated for each split, progressively updating the travel times and assigning trips accordingly.

Table 2 shows the parameter ranges tested for each baseline and the best parameter combinations obtained in our experiments.

algo	params range	best params Florence	best params Milan	best params Rome
PP	$p \in \{.1, .2, \ldots, .5\}$	$p = .2$	$p = .1$	$p = .2$
GR	$\Delta \in \{.2, .3, .4, .5\}$	$\Delta = .2$	$\Delta = .2$	$\Delta = .2$
PR	$\delta \in \{.2, .3, .4, .5\}$	$\Delta = .2$	$\Delta = .2$	$\Delta = .3$
KMD	$\epsilon \in \{.01, .05, .1, .2, .3\}$	$\epsilon = .2$	$\epsilon = .1$	$\epsilon = .01$
METIS	$p \in \{.01, .015, .02, \ldots, .1\}$	$p = .025$	$p = .01$	$p = .01$
	$s \in \{1.5, 1.75, 2, 2.25\}$	$s = 2.25$	$s = 2.25$	$s = 1.75$

Table 2: Parameter values explored for each algorithm and the best values obtained for each approach.

4.3 Measures

To assess the effectiveness of METIS and the baselines, we use three measures: total CO2 emissions, road coverage, and redundancy.

Total CO2. To accurately account for vehicle interactions and calculate CO2 emissions, we utilize the traffic simulator SUMO (Simulation of Urban MObility) [10, 19], which simulates each vehicle's dynamics, considering interactions with other vehicles, traffic jams, queues at traffic lights, and slowdowns caused by heavy traffic.

For each city and algorithm, we generate N routes (with N depending on the city, see Table 1) and simulate their interaction within SUMO during one peak hour, uniformly selecting a route's starting time during the hour.

To estimate CO2 emissions related to the trajectories produced by the simulation, we use the HBEFA3 emission model [3, 15], which estimates the vehicle's instantaneous CO2 emissions at a trajectory point j as:

$$\mathcal{E}(j) = c_0 + c_1 sa + c_2 sa^2 + c_3 s + c_4 s^2 + c_5 s^3 \quad (5)$$

where s and a are the vehicle's speed and acceleration in point j, respectively, and $c_0, \dots, c_5$ are parameters changing per emission type and vehicle taken from the HBEFA database [16]. To obtain the total CO2 emissions, we sum the emissions corresponding to each trajectory point of all vehicles in the simulation.

Road Coverage (RC). It quantifies the extent to which the road network is utilized by vehicles. It is calculated by dividing the cumulative length of the edges visited by any vehicle by the road network's overall length. Mathematically, given a set of routes R and the set of edges in these routes $S_R = \bigcup_{r \in R}\{e \in r\}$, we define RC as:

$$RC(R) = \frac{\sum_{e \in S_R} l(e)}{L(E)} \cdot 100 \quad (6)$$

where $l(e)$ is the length of edge e and $L(E) = \sum_{e \in E} l(e)$ is the total road length of the road network.

Road coverage characterizes a TA algorithm's road infrastructure usage. A higher road coverage indicates a larger proportion of the road network being utilized, which typically results in improved traffic distribution and reduced congestion. However, excessively high road coverage may increase vehicle travel distances, potentially producing higher emissions. Therefore, road coverage is a critical metrics for evaluating the effectiveness of TA algorithms in effectively utilizing the road infrastructure.

Time redundancy (RED). In the literature, redundancy is defined as the popularity of edges in a set of routes, also interpreted as the average utilization of edges that appear in at least one route [7]. Specifically, it is the fraction of the total number of edges of all routes divided by the total number of unique edges of all routes. Formally, given a set of routes R and a set of edges in these routes $S_R = \bigcup_{r \in R}\{e \in r\}$, we define it as:

$$RED(R) = \frac{\sum_{r \in R} |r|}{|S_R|} \quad (7)$$

If $RED(R) = 1$, there is no overlap among the routes in R, while $RED(R) = |R|$ when all routes are identical.

Note that RED does not consider traffic's dynamic evolution. To account for it, we define time redundancy as:

$$RED(R, t) = \frac{1}{|I|} \sum_{i \in I} \text{RED}(R_{i,t}) \quad (8)$$

where t is the length of the time window, $I = \{t_0, t_0+\sigma, t_0+2\sigma, \dots, t_{max}\}$ is the set of the starting times of each time window in the observation period $[t_0, t_{max})$ shifted by σ, and $\text{RED}(R_{i,t})$ is the RED of trips in R departed within time interval $[i, i+t)$. Low $RED(R, t)$ indicates that routes close in time are better distributed across edges.

5 RESULTS

Table 3 and Figure 4 compare Metis with all the baselines for all cities and measures. For each model, we show the results regarding the combination of parameter values leading to the lowest CO2 emissions (see Table 2). We use total CO2 emissions to assess the algorithms' effectiveness, while RC and RED are used to characterize the properties of the assigned routes.

Metis emerges as a significant breakthrough, with impressive reductions of CO2 emissions of 28% in Florence, 18% in Milan, and 46% in Rome compared to the best baseline (see Figure 4a-c and Table 3). This remarkable result is due to the synergistic combination of its unique core components: FLEP, KMD, and route scoring. FLEP is crucial in identifying less congested routes by estimating vehicles' current positions and dynamically adjusting edge weights. Complementing FLEP, KMD offers alternative routes that substantially cover the road network. Lastly, route scoring prioritizes less popular routes with higher capacity, helping accommodate traffic volume over uncongested routes.

Indeed, Metis achieves the highest road coverage in Florence (79.66%) and Milan (86.68%) and the second-highest in Rome (48.51%) (see Figure 4d-f and Table 3). Moreover, Metis achieves the lowest time redundancy in Florence (7.81) and Milan (7.41) and the second lowest in Rome (5.57): on average, the number of routes on each edge within a 5-minute temporal window is relatively low.

Figure 3 visually illustrates the spatial distribution of sample routes generated by Metis and KMD (the second-best model) in Milan. It is evident from the figure that Metis produces routes that are more evenly distributed across the city, leading to higher road coverage and lower time redundancy compared to KMD.

Among the baselines, GR shows the lowest CO2 emissions in Florence and KMD the lowest in Milan and Rome. GR has a high road coverage of 78.35% in Florence, 86.57% in Milan, and 51.57% in Rome (Figure 4d-f and Table 3).

KD and PLA exhibit high road coverage and time redundancy, resulting in the highest levels of CO2 emissions across all three cities. This is primarily because these methods have a tendency to assign trips to considerably long routes. Despite their simplicity, AON and ITA achieve CO2 emissions comparable to edge-weight methods (PP, PR, and GR).

Role of time redundancy. Figure 5 shows a strong correlation between time redundancy and CO2 emissions in Florence (r=0.92) and Milan (r=0.98) and a moderate correlation in Rome (r=0.52). As time redundancy decreases, CO2 emissions also decrease: low redundancy implies that trips close in time are likely to take different routes, alleviating congestion on edges. This means that, by utilizing the equations of Figure 5, we can estimate the CO2 emissions of TA algorithms based solely on the characteristics of the generated routes without the need for time-consuming simulations.

Ablation study. To understand the role of Metis' components, we selectively remove them creating three models: *(i)* M_1 uses KMD and route scoring but penalizes the entire routes of vehicles in transit instead of using FLEP; *(ii)* M_2 uses KMD and route scoring but no edge penalization; and *(iii)* M_3 uses FLEP and KMD but selects among alternative routes uniformly at random.

We find that removing components from Metis increases CO2 emissions compared to the complete Metis algorithm (Figure 6). In Milan and Rome, M_1, M_2, M_3 all outperform the best baseline (KMD). Only M_1 surpasses the best baseline in Florence, while M_2 and M_3 show slightly inferior performance. These findings highlight the importance of the synergistic combination of Metis' components.

Parameter Sensitivity. Figure 7 shows the relationship between Metis' parameter p, which controls penalization in FLEP,

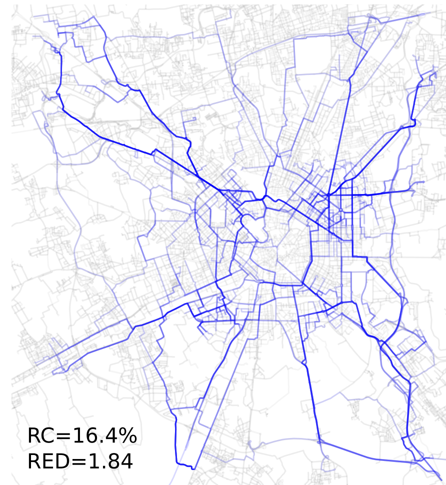
		algo	CO2 [t]	RC (%)	RED$(R, 5m)$
Florence	AR	PP	38.94 (2.99)	70.83 (.30)	9.70 (.03)
		GR	34.78 (1.21)	78.35 (.26)	8.87 (.03)
		PR	35.20 (1.88)	71.65 (.37)	10.45 (.03)
		KD	69.13 (6.96)	77.48 (.30)	11.94 (.06)
		PLA	67.00 (1.67)	72.56 (.37)	11.83 (.04)
		KMD	36.93 (.93)	73.40 (.35)	8.72 (.03)
	OS	AON	49.02	62.41	10.58
		ITA	48.16	62.42	10.58
		Metis	**25.19**	**81.06**	**7.81**
Milan	AR	PP	114.66 (1.62)	80.44 (.12)	9.27 (.01)
		GR	108.47 (1.80)	86.57 (.09)	7.87 (.02)
		PR	119.54 (1.44)	80.47 (.10)	9.27 (.01)
		KD	148.84 (2.14)	86.30 (.14)	10.41 (.02)
		PLA	265.51 (2.53)	79.71 (.12)	14.30 (.04)
		KMD	106.11 (1.31)	79.83 (.09)	8.93 (.02)
	OS	AON	126.61	76.40	9.70
		ITA	125.44	76.40	9.71
		Metis	**87.42**	**86.68**	**7.41**
Rome	AR	PP	143.85 (4.14)	34.44 (.07)	6.82 (.02)
		GR	138.36 (3.77)	**51.57 (.26)**	**5.41 (.02)**
		PR	141.74 (2.70)	42.02 (.14)	6.40 (.02)
		KD	133.95 (4.05)	43.61 (.11)	7.39 (.03)
		PLA	197.95 (2.11)	43.74 (.20)	10.19 (.04)
		KMD	118.89 (3.10)	26.90 (.03)	8.58 (.01)
	OS	AON	124.14	26.31	8.78
		ITA	123.23	26.36	8.77
		Metis	**64.07**	48.51	5.57

Table 3: Results of Metis and the baselines on CO2 emissions, road coverage (RC), and time redundancy (five-minute window). Std deviation in parentheses for non-deterministic methods. In bold, the lowest value of each measure and city.

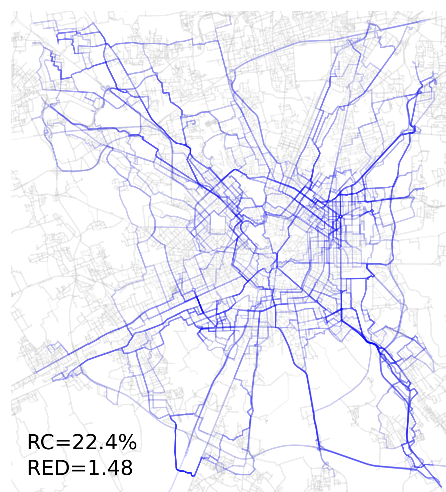
and CO2 emissions: apart from small values, higher values of p are associated with higher emissions. As p increases, FLEP penalizes more the edges that will be traversed by in-transit vehicles, forcing KMD to find alternative routes that may diverge considerably from the fastest route, resulting in increased congestion and CO2 emissions. In Milan and Rome, there is a clear increasing trend, showing that as p increases, CO2 emissions also increase (Figure 7b-c). Although there is a generally increasing trend in Florence, there are multiple peaks, indicating a complex relationship between p and CO2 emissions (Figure 7a). We also conduct a sensitivity analysis of the slowdown parameter s and observe no significant differences compared to the optimal parameter value shown in Table 2.

Execution times. Figure 8 compares Metis' response time with the baselines for 1,000 trips on 16 Intel(R) Core(TM) i9-9900 CPU 3.10GHz processors with 31GB RAM on Linux 5.15.0-56-generic.

AON and ITA are the fastest approaches: the former only requires computing the fastest path; the latter involves a single weight update for each of the four splits. PP, PR, and KMD are the second-fastest baselines, while GR and PLA are the slowest. GR is time-consuming because it modifies the weights of every edge in the network at each iteration; PLA because it computes the shortest path trees for each trip, which is time-intensive for large graphs.



(a) KMD



(b) Metis

Figure 3: Routes generated by KMD (a) and Metis (b) in Milan for 150 trips. RC is the road coverage, and RED is the time redundancy (5-minute window). Note how Metis exhibits a more spatially uniform traffic distribution than KMD.

Metis' average response times are within the same order of magnitude of baselines: 0.034s per request in Florence, 0.116s in Milan, and 0.125s in Rome, making it suitable for real-time TA where both efficiency and promptness matter (Figure 8).

Figure 8 also shows the response time of DUE, an iterative approach approximating the Dynamic User Equilibrium [13]. DUE has much longer execution times than Metis: 9 minutes for Florence (14.5 times slower), 25 for Milan (11.72 times slower) and 31 for Rome (14.44 times slower). However, this longer time does not always lead to lower CO2 emissions: in Milan, DUE achieves an 18% reduction in emissions compared to Metis, but in Florence and Rome, DUE increases them by 13% and 11%.

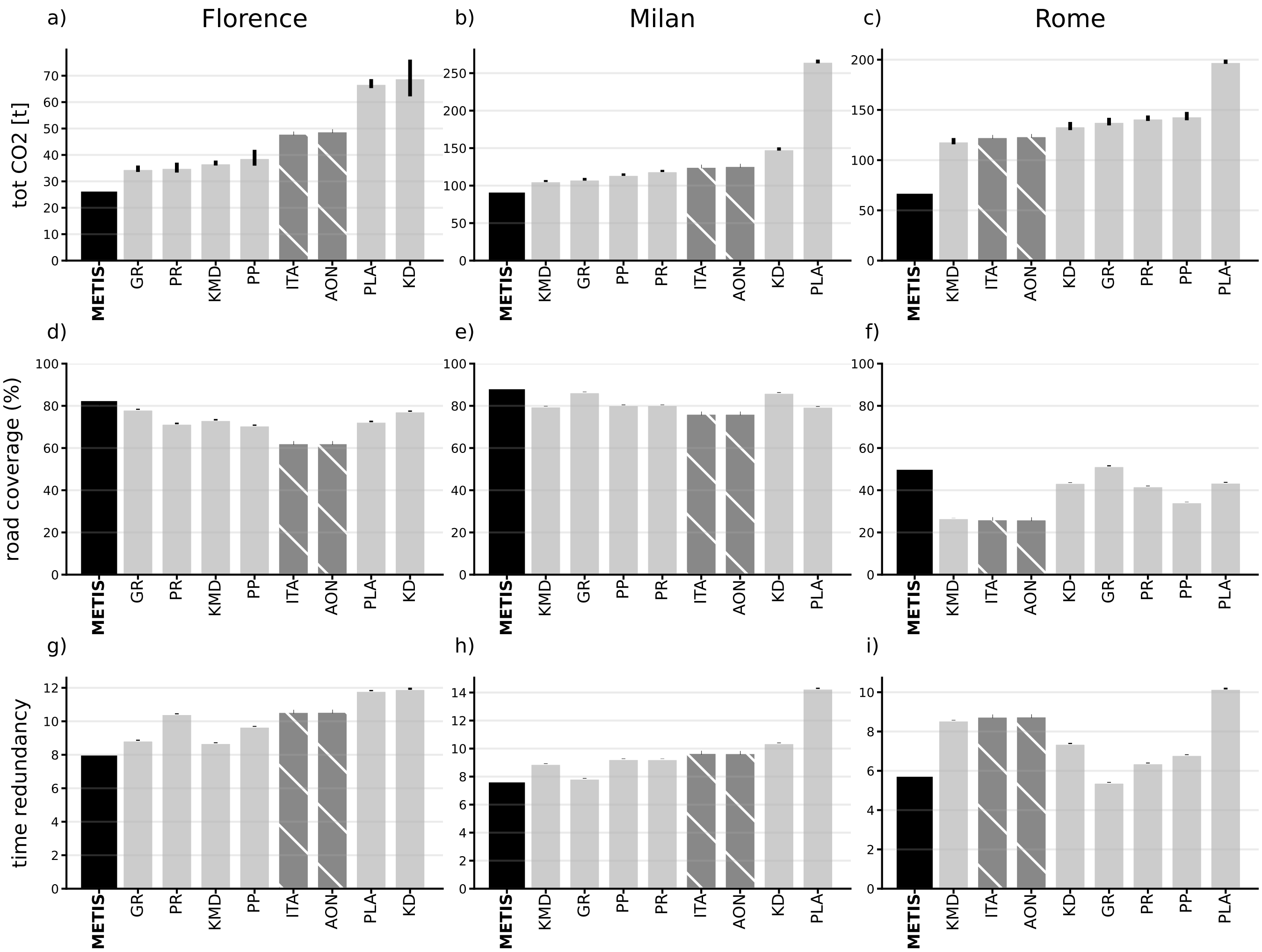


Figure 4: Comparison of METIS (black bar) with the baselines in Florence, Milan, and Rome on CO2 emissions (in tons), road coverage (in %), and time redundancy. For statistical reliability, we run non-deterministic algorithms (GR, PR, KMD, PP, PLA, KD) ten times and present the average values and the corresponding standard deviation.

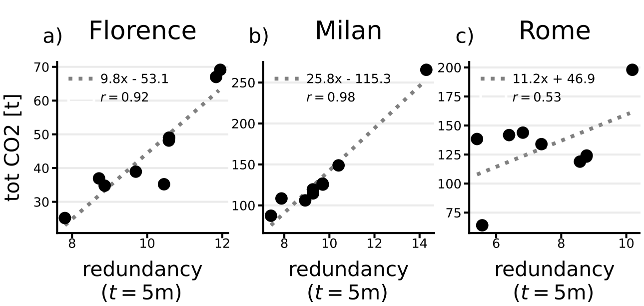


Figure 5: Pearson correlation (r) between time redundancy (5-minute window) and CO2 emissions. Black dots represent TA algorithms. The grey dashed line represents the curve fit.

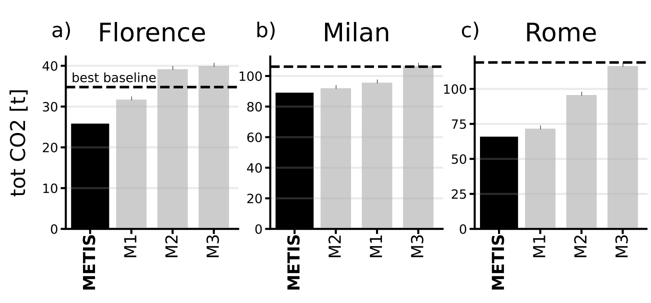


Figure 6: Comparison of METIS with models based on its components (M_1, M_2, M_3) and the best baseline (horizontal dashed line) in terms of total CO2 emissions.

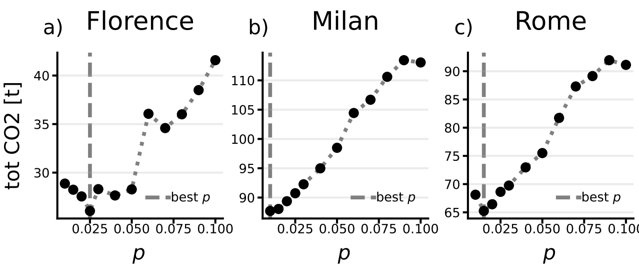


Figure 7: Relationship between METIS' parameter p and CO2 emissions. The vertical dashed line indicates the p value leading to the lowest emissions.

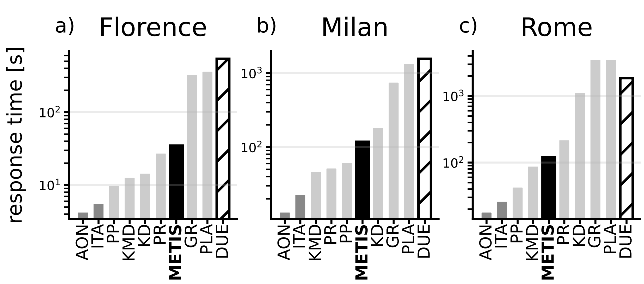


Figure 8: Comparison of response time (seconds) of METIS with the baselines and DUE for 1,000 trips.

6 CONCLUSION

In this paper, we introduced METIS, a one-shot algorithm for traffic assignment that effectively reduces CO2 emissions while maintaining computational efficiency. Future enhancements include incorporating additional measures to prioritize or discourage specific routes, refining FLEP using machine learning techniques for position estimation, estimating the slowdown factor for each road, and developing a distributed version for faster traffic assignments.

7 ACKNOWLEDGMENTS

Work supported by 1) EU H2020 HumaneAI-net 952026; 2) EU H2020 SoBigData++ 871042; 3) EU CHIST-ERA-19-XAI-010; 4) NextGenerationEU-National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) Project "SoBigData.it-Strengthening the Italian RI for Social Mining and Big Data Analytics", prot. IR0000013, avviso n. 3264 on 28/12/2021; 5) PNRR-M4C2-Investimento 1.3, Partenariato Esteso PE00000013 - FAIR - Future Artificial Intelligence Research - Spoke 1 Human-centered AI, funded by the European Commission under the NextGeneration EU programme. We thank Daniele Fadda for his support in data visualization.

REFERENCES

[1] Husain Aljazzar and Stefan Leue. 2011. K *: A heuristic search algorithm for finding the k shortest paths. *Artificial Intelligence - AI* 175 (12 2011), 2129–2154.

[2] Martin Beckman, C. McGuire, Christopher Winsten, and Tjalling Koopmans. 1956. Studies in the Economics of Transportation. *OR* 7 (12 1956), 146.

[3] Matteo Böhm, Mirco Nanni, and Luca Pappalardo. 2022. Gross polluters and vehicle emissions reduction. *Nat. Sustain.* 5, 8 (2022), 699–707.

[4] Bureau of Public Roads. 1964. *Traffic Assignment Manual for Application with a Large, High Speed Computer.* Vol. 37. U.S. Dept. of Commerce, Bureau of Public Roads, Office of Planning, Urban Planning Division.

[5] M. E. Campbell. 1950. Route selection and traffic assignment. In *Highway Research Board.* Washington, DC.

[6] Ming-Ying Chen and Attahiru Sule Alfa. 1991. A Network Design Algorithm Using a Stochastic Incremental Traffic Assignment Approach. *Transp. Sci.* 25 (1991), 215–224.

[7] Dan Cheng, Olga Gkountouna, Andreas Züfle, Dieter Pfoser, and Carola Wenk. 2019. Shortest-Path Diversification through Network Penalization: A Washington DC Area Case Study. In *IWCTS@SIGSPATIAL.* Article 10, 1-10 pages.

[8] Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, and Ulf Leser. 2015. Alternative Routing: K-Shortest Paths with Limited Overlap. In *ACM SIGSPATIAL GIS.* Article 68, 4 pages.

[9] Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, Ulf Leser, and David B. Blumenthal. 2018. Finding K-Dissimilar Paths with Minimum Collective Length. In *ACM SIGSPATIAL GIS.* 404–407.

[10] Giuliano Cornacchia, Matteo Böhm, Giovanni Mauro, Mirco Nanni, Dino Pedreschi, and Luca Pappalardo. 2022. How Routing Strategies Impact Urban Emissions. In *ACM SIGSPATIAL GIS.* Article 42, 4 pages.

[11] Dino Pedreschi et al. 2023. Social AI and the Challenges of the Human-AI Ecosystem. *arXiv preprint arXiv:2306.13723* (2023).

[12] Marguerite Frank and Philip Wolfe. 1956. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly* 3 (03 1956), 95 – 110.

[13] Terry L Friesz and Terry L Friesz. 2010. Dynamic user equilibrium. *Dynamic optimization and differential games* (2010), 411–456.

[14] Christian Häcker, Panagiotis Bouros, Theodoros Chondrogiannis, and Ernst Althaus. 2021. Most Diverse Near-Shortest Paths. In *ACM SIGSPATIAL GIS.* 229–239.

[15] INFRAS. 2013. Handbuch für Emissionsfaktoren. http://www.hbefa.net/.

[16] Daniel Krajzewicz, Michael Behrisch, Peter Wagner, Raphael Luz, and Mario Krumnow. 2015. Second Generation of Pollutant Emission Models for SUMO. In *Modeling Mobility with Open Data*, Michael Behrisch and Melanie Weber (Eds.). Springer International Publishing, Cham, 203–221.

[17] L. Li, M. Cheema, H. Lu, M. Ali, and A. N. Toosi. 2022. Comparing Alternative Route Planning Techniques: A Comparative User Study on Melbourne, Dhaka and Copenhagen Road Networks. *IEEE Trans. Knowl. Data Eng.* 34, 11 (2022), 5552–5557.

[18] Huiping Liu, Cheqing Jin, Bin Yang, and Aoying Zhou. 2018. Finding Top-k Shortest Paths with Diversity. *IEEE Trans. Knowl. Data Eng.* 30, 3 (2018), 488–502.

[19] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. 2018. Microscopic Traffic Simulation using SUMO. In *21st Int. Conf. Intell. Transp. Syst.* 2575–2582.

[20] Cambridge Vehicle Information Technology Ltd. 2005. Choice Routing. http://www.camvit.com.

[21] Marin Lujak, Stefano Giordani, and Sascha Ossowski. 2015. Route guidance: Bridging system and user optimization in traffic assignment. *Neurocomputing* 151 (2015), 449–460.

[22] Valentina Morandi. 2023. Bridging the user equilibrium and the system optimum in static traffic assignment: a review. *4OR* (2023), 1–31.

[23] Luca Pappalardo, Matteo Böhm, Giuliano Cornacchia, Giovanni Mauro, Dino Pedreschi, and Mirco Nanni. 2022. Human Mobility, AI assistants, and urban emissions: an insidious triangle. (2022).

[24] Luca Pappalardo, Ed Manley, Vedran Sekara, and Laura Alessandretti. 2023. Future directions in human mobility science. *Nature Computational Science* (2023), 1–13.

[25] Luca Pappalardo, Filippo Simini, Gianni Barlacchi, and Roberto Pellungrini. 2022. scikit-mobility: A Python Library for the Analysis, Generation, and Risk Assessment of Mobility Data. *J. Stat. Softw.* 103, 4 (2022), 1–38.

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

[27] Transportation Research Board 2000. *Highway Capacity Manual.* Transportation Research Board, Washington, D.C.

[28] United Nations General Assembly. 2015. *Transforming our world: the 2030 Agenda for Sustainable Development.* Technical Report. Accessed: 2021-02-23.

[29] Pu Wang, Timothy Hunter, Alexandre Bayen, Katja Schechtner, and Marta C. Gonzalez. 2012. Understanding Road Usage Patterns in Urban Areas. *Sci. rep.* 2 (12 2012), 1001.

[30] Yi Wang, Wai Y Szeto, Ke Han, and Terry L Friesz. 2018. Dynamic traffic assignment: A review of the methodological advances for environmentally sustainable road transportation applications. *Transport. Res. B-Meth* 111 (2018), 370–394.

[31] JG Wardrop. 1952. Proceedings of the Institute of Civil Engineers. (1952).

[32] J Wardrop. 1952. Some Theoretical Aspects of Road Traffic Research. *Ice Proceedings: Engineering Divisions* 1 (01 1952), 325–362.

[33] Lingkun Wu, Xiaokui Xiao, Dingxiong Deng, Gao Cong, Andy Diwen Zhu, and Shuigeng Zhou. 2012. Shortest Path and Distance Queries on Road Networks: An Experimental Evaluation. (2012).

[34] Jin Y. Yen. 1971. Finding the K Shortest Loopless Paths in a Network. *Manag. Sci.* 17, 11 (1971), 712–716.