ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA251389

Efficient Inference of Sources and Targets in a Graph with Limited Observations

Wanrong Yang[a] **and Dominik Wojtczak**[b,*]

[a,b]Department of Computer Science, University of Liverpool
ORCID (Wanrong Yang): https://orcid.org/0000-0003-0216-3762, ORCID (Dominik Wojtczak): https://orcid.org/0000-0001-5560-0546

Abstract. We study the problem of inferring all possible sources and targets of shortest walks in a weighted directed graph, constrained by a set of observed edges. We present two efficient polynomial-time algorithms to solve this problem. The first algorithm applies to graphs with strictly positive edge weights, while the second, a more complex algorithm, handles graphs with non-negative weight cycles, the broadest class of graphs where shortest walks exist. We demonstrate the effectiveness of the first algorithm by evaluating it on real-world road networks, achieving consistent performance on graphs with up to 7.7 million nodes and over 16 million edges. Our results show that the proposed approach scales efficiently and robustly across large networks.

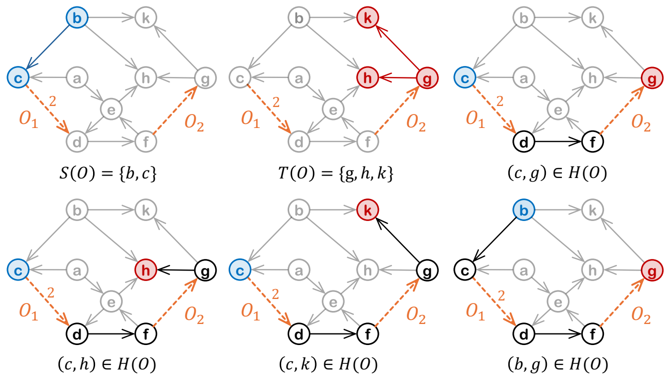

Figure 1: An example input graph and observation set $O = \{o_1, o_2\}$

1 Introduction

The trajectory prediction problem involves predicting the future positions of an object based on its past and current GPS coordinates. This problem has numerous practical applications across various domains (see, e.g. surveys [11, 24]). In urban transportation, trajectory prediction can help forecast vehicle or pedestrian [25] routes in cities, enabling improvements in traffic management and navigation systems. For autonomous vehicles, accurate trajectory prediction allows these systems to anticipate the movements of nearby vehicles or pedestrians, see e.g. [11], ensuring safer and more efficient route planning. Finally, in surveillance and security, predicting an individual's path can aid in strategic planning, such as intercepting a suspect by establishing a blockade along the anticipated route.

Variants of this problem include *next Point of Interest (POI) recommendation* [18], which focuses on predicting the next location only, and *taxi destination* problem [20, 17], that focuses on predicting the final location instead. Traditionally, all these problems are addressed using data-driven approaches that assume the existence of a large dataset of past trajectories (see the related work section below or [14] for a survey). Moreover, these methods typically operate in a spatio-temporal continuous state space, although some discretize the space[34] or use graph-based representations instead [13, 7].

Despite the extensive research using data-centric methods, there is a notable gap in the algorithmic exploration of trajectory prediction and its related problems, particularly concerning its computational complexity. To the best of our knowledge, no prior work has formally analyzed such problems from an algorithmic perspective. We aim to bridge this gap by investigating several new variants of the destination prediction problem within a discrete state space modeled as a directed weighted graph. Graphs are a natural representation in this context, as they effectively model complex relationships through nodes and edges[26, 21, 22], making them suitable for urban road networks where nodes denote locations and edges represent routes with associated costs, such as distance or travel time[4].

We depart from the traditional data-centric formulation of these problems and focus on the source and target prediction based on a partial trajectory under the assumption that the trajectory follows a shortest route. This assumption is both practical and relevant, as prior work [13, 7] has shown that adding such a constraint while predicting trajectory can enhance accuracy. Moreover, this reflects how many real-world journeys are navigated today: most drivers rely on GPS-based tools that recommend routes optimized for distance or travel time. These are either built-in car navigation systems or mobile applications such as Waze, Google Maps etc. The optimal routes are either computed statically (based on typical travel times) or dynamically (based on current traffic congestion). Either way, the underlying weighted graphs used to calculated these can be obtained and one can immediately reason about the possible sources or targets of a route without relying on a large number of previously observed trajectories. Furthermore our solution is capable to work with a very limited number of observations (trajectory points) and even when their temporal order may be unknown. For instance, imagine a situation where we are looking for a missing person traveling in a car. We may only know certain locations where the car was spotted, but the exact order or timestamps of these sightings may be unavailable.

To illustrate these computational problems, consider the example

[*] Corresponding Author. Email: D.Wojtczak@liverpool.ac.uk

in Figure 1. It represents a graph of connections between different locations and all edges have weight of 1 except the edge $c \to d$ that has weight of 2. This weight can be the physical distance, or current travel time as given by a car navigation system. In this example, we only know that the person traveled along edges o_1 and o_2, but we do not know the order in which these edges were taken. The goal is to determine the possible origin and destination of the person's journey. Assuming that only walks with the lowest total weight are taken, we find that the possible origins are $S(O) = \{b, c\}$, the possible destinations are $T(O) = \{g, h, k\}$, and the set of valid origin-destination pairs of shortest walks is $H(O) = \{(c, g), (c, h), (c, k), (b, g)\}$, as shown in the figure. For instance, b can be an origin, because there exists a shortest walk $b \to c \to d \to f \to g$ that passes through o_1 and o_2. Similarly, k can be a destination, because there exists a shortest walk $c \to d \to f \to g \to k$ that passes through o_1 and o_2. Also, a cannot be an origin because any shortest walk starting from a and visiting f passes through e instead of c, because that results in a lower total weight.

It is important to note that $H \neq S \times T$, meaning the problem of computing the origin-destination pairs cannot be reduced to independently computing the set of origins and the set of destinations. For instance, the pair (b, k) is invalid because the shortest walk from b to k is $b \to k$ and it does not traverse either o_1 or o_2. Similar argument applies to pair (b, h) as well.

To address these source and target prediction problems, we first define them formally, and then propose two polynomial-time algorithms to solve them. The first one only works for graphs with positive weights, which is a natural assumption to make, because distances or travel times are always positive. Although this algorithm is relatively simple, it requires careful proof analysis and only computing the shortest distances that are strictly necessary for it to work efficiently. While restricting all weights to be positive implies any shortest walk to be a path (no node is revisited), our second algorithm works for all graphs with no negative cycles. Note that shortest walks may not even exist in graphs with negative cycles, because any walk can be made shorter by repeating such a cycle again. Therefore, the class of graph with no negative cycles is the most general class for which these problems are well-defined and can be studied. Finally, we evaluate our approach on real-world road network datasets containing up to 7.7 million nodes, demonstrating its scalability and applicability. Our work represents a significant step toward bridging the gap between data-driven techniques and algorithmic research in trajectory prediction, establishing a foundation for further advancements in this area.

2 Related work

The trajectory prediction problem and its variants (e.g., POI recommendation and taxi destination prediction) have been extensively studied in the literature. Works like [33, 12] utilize Recurrent Neural Networks(RNNs), Long Short-Term Memory(LSTM) to capture temporal patterns in trajectory sequences, while [29] extends this by integrating long and short-term preferences via non-local networks and geo-dilated RNNs. Recent advances adopt transformer architectures [2, 31, 16] or meta-learning frameworks [28] to address cold-start scenarios. However, these methods primarily focus on continuous spatial representations and rely on dense training data, diverging from our graph-structured problem formulation. Notably, [27, 15, 6] explore uncertainty modeling in trajectory prediction but remain constrained to probabilistic frameworks rather than algorithmic point of view. Several studies model trajectories on graphs to incorporate topological constraints. [33] first highlights the importance of graph-structured trajectory modeling using RNNs, while [15] proposes a method for path failure prediction in time-evolving graphs. Recent works like [32] employ spatial-temporal graph convolutional networks to capture neighborhood interactions, [19] introduces graph-LSTM model to boosts vehicle trajectory prediction by better capturing feature interactions. The closest to our work is [30], which predicts interactive trajectories via marginal-to-interactive reasoning, but it still operates within data-driven paradigms without formal complexity analysis. Our proposed polynomial-time algorithms operate without training data and overhead, directly solving the inverse problem of reconstructing possible sources/targets from limited observations. This aligns with the scalability demonstrated in [10, 3] for POI recommendation but extends it to worst-case complexity guarantees.

3 Preliminaries

A *weighted directed graph* G is a triple (V, E, w), where V is a finite set of *nodes*, $E \subseteq V \times V$ is a set of directed *edges*, and $w : E \to \mathbb{R}$ is a *weight function* that assigns a weight to each edge in G. Specifically, an edge is a pair (u, v), where u is the *source* of this edge and v its *target*. For an edge e, let $s(e)$ denote its source, $t(e)$ its target, and $w(e)$ be its weight. Any sequence of edges $\rho = e_1, \ldots, e_l$, such that $\forall_{i \leq l}\ e_i \in E$ and for all $i < l$ we have $t(e_i) = s(e_{i+1})$ is a *walk* in G of length l that connects its source $s(\rho) \overset{\text{def}}{=} s(e_1)$ with its target $t(\rho) \overset{\text{def}}{=} t(e_l)$. We write that $e \in \rho$, if the walk ρ uses edge e at some point, i.e., $e = e_i$ for some $i \leq l$. For two walks ρ, ρ', such that $t(\rho) = s(\rho')$, let the walk $\rho \cdot \rho'$ be the concatenation of ρ with ρ'. We say that a walk ρ is *consistent with a set of observations* $O = \{o_1, \ldots, o_k\}$, if for all i we have $o_i \in \rho$. On the other hand, it is *consistent with a sequence of observations* $O = (o_1, \ldots, o_k)$ if all o_i are used by ρ and in this particular order, i.e., there exists an increasing sequence $i_1, \ldots, i_l$ such that $e_{i_j} = o_j$ for all $j \leq k$.

If $e = (u, v)$ is an edge, then its reverse, denoted e^R, is an edge (v, u). Similarly, if $G = (V, E)$ is a graph, then its reverse is $G^R = (V, E^R)$ where $E^R = \{e^R : e \in E\}$. Finally, if $O = \{o_1, \ldots, o_k\}$ is a set of observations then $O^R = \{o_1^R, \ldots, o_k^R\}$, and if $O = (o_1, \ldots, o_k)$ is a sequence of observations then $O^R = (o_k^R, \ldots, o_1^R)$.

The set of all walks connecting u with v in G will be denoted by $\mathcal{W}_G(u, v)$. The weight of a walk $\rho = e_1, \ldots, e_l$ is $w(\rho) = \Sigma_{i=1}^{l} w(e_i)$. A *cycle* is a walk such that $s(\rho) = t(\rho)$. A graph G is *strongly connected* if there exists a walk between any two nodes in G. A graph (V', E', w') is a *sub-graph* of (V, E, w) if $V' \subseteq V$, $E' = E \cap (V' \times V')$, and $w' = w|_{E'}$. Any graph G can be decomposed into strongly connected components, where each component is a strongly connected sub-graph of G.

Assuming that there are no negative cycles in G, the *shortest distance* in G from u to v is $\mathcal{D}_G(u, v) = \min_{\rho \in \mathcal{W}_G(u,v)} w(\rho)$. If there is no walk from u to v in G then we stipulate that $\mathcal{D}_G(u, v) = \infty$. The set of all *shortest walks from u to v* is denoted by $\mathcal{S}_G(u, v) = \{\rho \in \mathcal{W}_G(u, v) : w(\rho) = \mathcal{D}_G(u, v)\}$. The set of all *shortest walks starting at u* is $\mathcal{S}_G(u, \cdot) = \cup_{v \in V}\ \mathcal{S}_G(u, v)$, and *shortest walks terminating at v* is $\mathcal{S}_G(\cdot, v) = \cup_{u \in V}\ \mathcal{S}_G(u, v)$. The set of all *shortest walks* is $\mathcal{S}_G(\cdot, \cdot) = \cup_{v \in V} \mathcal{S}_G(\cdot, v)$. If $\rho \in \mathcal{S}_G(\cdot, \cdot)$ then ρ is called *a shortest walk* (from $s(\rho)$ to $t(\rho)$).

4 Sources and Targets Inference Problem

From now on we fix a graph G, so we can avoid using it in the subscripts. We now formally define the problems that we study. We assume that we are given a set of observations $O = \{o_1, \ldots, o_k\}$

where $o_i \in E$ for all $i \leq k$ and look at the possible sources and targets of shortest walks consistent with this observation set O. Typically, we will have $k \ll |E|$, meaning the number of observations is significantly smaller than the number of edges in the graph.

Definition 1 (POSSIBLE SOURCES PROBLEM). *Given a set of observations $O = \{o_1, \ldots, o_k\}$, find the set, $S(O)$, of all sources $u \in V$ for which there exists a shortest walk $\rho \in \mathcal{S}(u, \cdot)$ that is consistent with O.*

Definition 2 (POSSIBLE TARGETS PROBLEM). *Given a set of observations $O = \{o_1, \ldots, o_k\}$, find the set, $T(O)$, of all targets $v \in V$ for which there exists a shortest walk $\rho \in \mathcal{S}(\cdot, v)$ that is consistent with O.*

Definition 3 (SOURCES-TARGETS PROBLEM). *Given a set of observations $O = \{o_1, \ldots, o_k\}$, find the set, $H(O)$, of all possible pairs (u, v) for which there exists a shortest walk $\rho \in \mathcal{S}(u, v)$ that is consistent with O.*

Note that we assume that we do not know the exact order in which observations take place. So we merely know that the object was moving between different points, but not at what time. If the order of the observations is fully known the problem actually becomes easier.

Now, we start off by establishing some basic well-known facts about shortest walks that we will use throughout the paper.

Proposition 1. *If $\rho = \rho_1 \cdot \rho_2 \cdot \rho_3$, where ρ_1 or ρ_3 are possibly empty, is a shortest walk from $s(\rho)$ to $t(\rho)$ then ρ_2 is a shortest walk from $s(\rho_2)$ to $t(\rho_2)$.*

Proof. Suppose there is a shorter walk ρ_2' from $s(\rho_2)$ to $t(\rho_2)$, then $\rho_1 \cdot \rho_2' \cdot \rho_3$ would be a shorter walk from $s(\rho)$ to $t(\rho)$ contradicting the optimality of ρ. □

Proposition 2. *If ρ_1 and ρ_2 are two walks such that $t(\rho_1) = s(\rho_2)$ and $\mathcal{D}(s(\rho_1), t(\rho_2)) = w(\rho_1) + w(\rho_2)$ then $\rho_1 \cdot \rho_2$ is a shortest walk from $s(\rho_1)$ to $t(\rho_2)$.*

We can already proceed to establish a characterization of the set of possible targets for a sequence of observations.

Theorem 3. *For a sequence of observations $O = (o_1, \ldots, o_k)$, we have $v \in T(O)$ iff $\infty \neq \mathcal{D}(s(o_1), v) = \sum_{i=1}^{k-1} (w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1}))) + w(o_k) + \mathcal{D}(t(o_k), v)$.*

Proof. (⇒) If v is a possible target consistent with a sequence of observations $O = (o_1, \ldots, o_k)$ that means that there exists a shortest walk $\rho = \rho_0 \cdot o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ such that $t(\rho_k) = v$ and each ρ_i is a shortest walk due to Proposition 1. The walk $o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ also has to be a shortest one from $s(o_1)$ to v, and so its total weight $\sum_{i=1}^{k-1}(w(o_i) + w(\rho_i)) + w(o_k) + w(\rho_k) = \sum_{i=1}^{k-1}(w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1})) + w(o_k) + \mathcal{D}(t(o_k), v)$ has to be the same as $\mathcal{D}(s(o_1), v)$. This of course implies that $\mathcal{D}(s(o_1), v) \neq \infty$.

(⇐) Note that $\sum_{i=1}^{k-1}(w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1})) + w(o_k) + \mathcal{D}(t(o_k), v) \neq \infty$ implies that $\mathcal{D}(t(o_i), s(o_{i+1})) \neq \infty$ for all $i < k$ and $\mathcal{D}(t(o_k), v) \neq \infty$. So there exists a shortest walk, ρ_i, from $t(o_i)$ to $s(o_{i+1})$ for all $i < k$ and a shortest walk ρ_k from $t(o_k)$ to v. Then the walk $o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ is consistent with O and is a shortest walk from $s(o_1)$ to v, because its weight, $\sum_{i=1}^{k-1}(w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1})) + w(o_k) + \mathcal{D}(t(o_k), v)$ is the same as the smallest possible weight $\mathcal{D}(s(o_1), v)$. □

Note that the above theorem shows how we can easily deal with this computational problem when the complete ordering of observations is given. So the biggest challenge is to find the right ordering of observations. We show in the next section that in the case of positive weights we can simply order them by their distance to the destination.

5 Positive Weighted Graphs

Every statement in this section assume that $w(e) > 0$ for every $e \in E$. This assumption allows use to establish the following.

Lemma 4. *Let ρ be a shortest walk consistent with a sequence of observations $O = (o_1, \ldots, o_k)$. Then the sequence $(\mathcal{D}(t(o_1), t(\rho)), \ldots, \mathcal{D}(t(o_k), t(\rho)))$ is strictly decreasing.*

Proof. Suppose there exists i such that $\mathcal{D}(t(o_i), v) \leq \mathcal{D}(t(o_{i+1}), v)$. Since ρ is consistent with O, it has to have a suffix $o_i \cdot \rho_i \cdot o_{i+1} \cdot \rho'$ for some walks ρ_i and ρ'. This suffix has weight $w(o_i) + w(\rho_i) + w(o_{i+1}) + w(\rho')$. At the same time, we know that this suffix has to be a shortest walk from $s(o_i)$ to $t(\rho)$ thanks to Proposition 1, so this weight is the same as $\mathcal{D}(t(o_i), t(\rho))$. We also know that ρ' is a shortest walk from $t(o_{i+1})$ to $t(\rho)$ thanks to Proposition 1, so $w(\rho') = \mathcal{D}(t(o_{i+1}), t(\rho))$. Therefore, we would have that $\mathcal{D}(t(o_i), t(\rho)) = w(o_i) + w(\rho_i) + w(o_{i+1}) + \mathcal{D}(t(o_{i+1}, t(\rho))$; a contradiction as all weights are positive. □

We now establish the correctness of Algorithm 1 below for computing the set of possible targets.

Theorem 5. *Algorithm 1 solves* POSSIBLE-TARGET-PROBLEM *for a positive-weighted G and observation set O.*

Proof. (⇒) Suppose that this algorithm returns v as a possible target. Let $(o_1, \ldots, o_k)$ be the sequence that we get after sorting O by $\mathcal{D}(t(o_i), v)$ in descending order. As v was added to T at line 8, it means that $\mathcal{D}(s(o_1), v) \neq \infty$ and $\mathcal{D}(s(o_1), v) = \sum_{i=1}^{k-1}(w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1})) + w(o_k) + \mathcal{D}(t(o_k), v)$ had to hold. It follows from Theorem 3 that $v \in T(o_1, \ldots, o_k)$, so also $v \in T(O)$ because, straight from the definition, any walk that is consistent with a given sequence of observations is also consistent with the set of these observations.

(⇐) Suppose $v \in T(O)$, which means that there exists a walk, ρ, that terminates at v and is consistent with the set of observations O. Let $(o_1, \ldots, o_k)$ be the order of the observations in O in which ρ actually visits them, and so ρ is consistent with this particular sequence of observations. Thanks to Lemma 4 we then know that $(\mathcal{D}(t(o_1), v), \ldots, \mathcal{D}(t(o_k), v))$ is strictly decreasing. This means that this particular order of observations $(o_1, \ldots, o_k)$ will be found by Algorithm 1 at line 3. Now, ρ has a prefix, ρ', that starts with o_1 and is a shortest walk from $s(o_1)$ to $t(\rho') = t(\rho) = v$ due to Proposition 1, which implies that $\mathcal{D}(s(o_1), v) \neq \infty$. This prefix ρ' has to look as follows: $\rho' = o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ for some walks ρ_i that all have to be shortest due to Proposition 1. As $\mathcal{D}(s(o_1), v) \neq \infty$, the algorithm will proceed to line 7, where the equality will hold because ρ' is a shortest walk from $s(o_1)$ to v, and $w(\rho_i) = \mathcal{D}(t(o_i), s(o_{i+1}))$ for all $i < k$ and $w(\rho_k) = \mathcal{D}(t(o_k), v)$. □

To deal with the problem of computing possible sources, we will simply show how to reduce it to computing possible targets. First note the following.

Proposition 6. *Walk ρ is shortest and consistent with observation set/sequence O in G iff walk ρ^R is shortest and consistent with observation set/sequence O^R in G^R.*

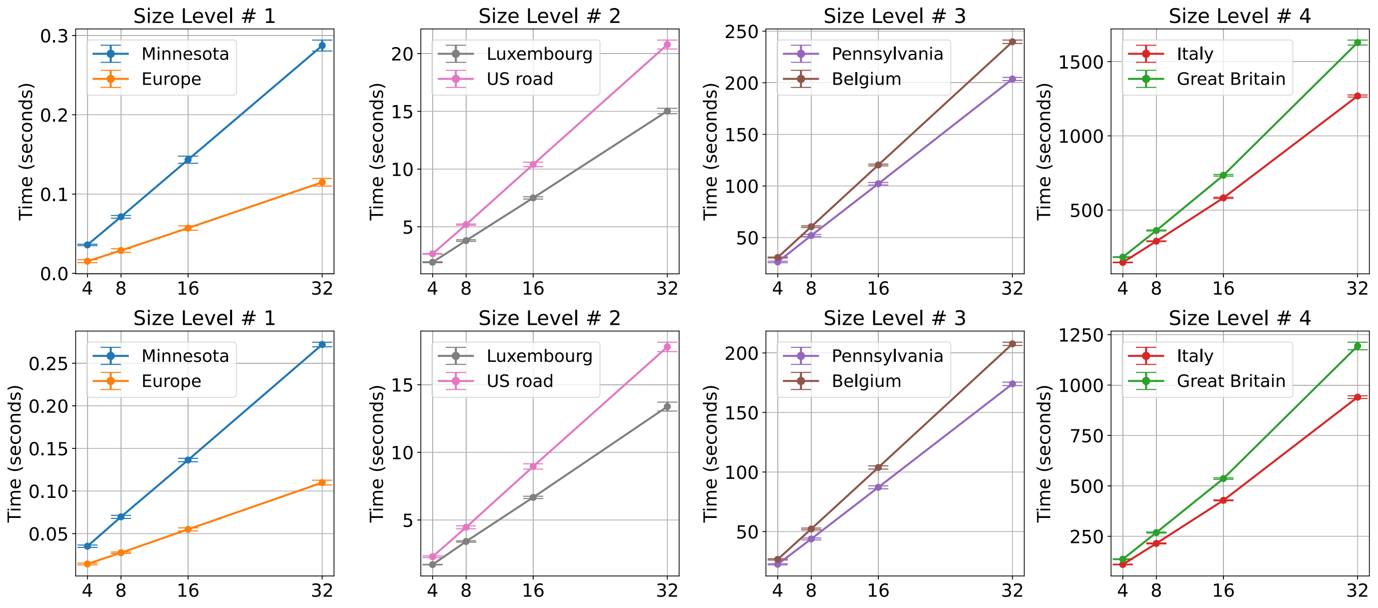

Figure 2: Average running time for sources searching (first row) and targets searching (second row). The x-axis is the number of observations.

Algorithm 1: PTP(G, O)

```
Input: A graph $G = (V, E, w)$ such that $w : E \to \mathbb{R}_+$ and a
       set of observations $O = \{o_1, \ldots, o_k\}$.
Output: The set of all possible targets $T(O)$.
1  $T \leftarrow \emptyset$;
2  for each possible target $v \in V$ do
3      sort $O$ by $\mathcal{D}(t(o_i), v)$ in descending order;
4      if $\mathcal{D}(s(o_1), v) = \infty$ then
5          continue;
6      end
7      if $\mathcal{D}(s(o_1), v) = \sum_{i=1}^{k-1}(w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1})) + w(o_k) + \mathcal{D}(t(o_k), v)$ then
8          add $v$ to $T$
9      end
10 end
11 return $T$
```

Proof. Straight from the definition if ρ is consistent with observation set/sequence O then ρ^R is consistent with observation set/sequence O^R, because all edges are reversed and so is the order of observations in case O is a sequence. Also, ρ^R is a shortest walk in G^R, because for every $u, v \in V$ we have $\mathcal{D}_G(u, v) = \mathcal{D}_{G^R}(v, u)$. □

We now establish the correctness of Algorithm 2 below for computing the set of possible sources that simply reverses the graph and observations, and then calls Algorithm 1.

Theorem 7. *Algorithm 2 solves* POSSIBLE-SOURCE-PROBLEM *for a given G and observation set O.*

Proof. Note that $u \in S(O)$ if and only if there exists a shortest walk, ρ, consistent with O that starts at u. Thanks to Proposition 6, the latter is the same as saying that ρ^R is a shortest walk in G^R consistent with O^R that terminates at u. So $u \in S_G(O)$ if and only if $u \in T_{G^R}(O^R)$. Therefore, Algorithm 2 correctly simply returns the set of possible targets for G^R and observation set O^R. □

Algorithm 2: PSP(G, O)

```
Input: A graph $G = (V, E, w)$ such that $w : E \to \mathbb{R}_+$ and a
       set of observations $O = \{o_1, \ldots, o_k\}$.
Output: The set of all possible sources $S(O)$.
1  return PTP$(G^R, O^R)$
```

Finally, we deal with the problem of computing the source-target pairs using Algorithm 3.

Theorem 8. *Algorithm 3 solves* SOURCE-TARGET-PROBLEM *for a given G and observation set O.*

Proof. ($\Rightarrow$) Suppose that this algorithm returns (u, v) as a possible source-target pair. That means that v had to be returned at line 2 as a possible target, and so there exists a walk ρ consistent with O that ends at v. Let $(o_1, \ldots, o_k)$ be the sequence that we get after sorting O by $\mathcal{D}(t(o_i), v)$ in descending order. Note that ρ has to be consistent with the sequence of observations $(o_1, \ldots, o_k)$. This is because only this ordering of O makes $(\mathcal{D}(t(o_1), v)), \ldots, \mathcal{D}(t(o_k), v))$ strictly decreasing, and so any other ordering would be wrong due to Lemma 4. So $\rho = \rho_0 \cdot o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ for some shortest walks ρ_i.

We also know that $\mathcal{D}(u, v) \neq \infty$ and $\mathcal{D}(u, v) = \mathcal{D}(u, s(o_1)) + \mathcal{D}(s(o_1), v)$ had to hold for (u, v) to be added to H at line 10. Both of them together imply that $\mathcal{D}(u, s(o_1)) \neq \infty$ and $\mathcal{D}(s(o_1), v) \neq \infty$. Let ρ'_0 be any shortest walk from u to $s(o_1)$ that has to exists as $\mathcal{D}(u, s(o_1)) \neq \infty$. We claim that $\rho'_0 \cdot o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ is a shortest walk that starts at u and ends at v and is consistent with the observation set O, justifying that $(u, v) \in H(O)$. This is simply because $o_1 \cdot \rho_1 \cdot o_2 \cdot \ldots \cdot o_k \cdot \rho_k$ is a shortest walk, and so its weight is $\mathcal{D}(s(o_1), t(\rho_k)) = \mathcal{D}(s(o_1), v)$, and then we can directly apply Proposition 2 as $w(\rho'_0) = \mathcal{D}(u, s(o_1))$ and $\mathcal{D}(u, v) = \mathcal{D}(u, s(o_1)) + \mathcal{D}(s(o_1), v)$ holds.

($\Leftarrow$) Suppose $(u, v) \in H(O)$, which means that there exists a shortest walk, ρ, that starts at u and terminates at v and is consistent with the set of observations O. The existence of ρ clearly shows that $v \in T(O)$, so Algorithm 3 will find v at line 2 and consider it at line 3. Let $(o_1, \ldots, o_k)$ be the order of the observations in O in which ρ actually visits them, and so ρ is consistent with this particular sequence of observations. Thanks to Lemma 4 we know that $(\mathcal{D}(t(o_1), v), \dots, \mathcal{D}(t(o_k), v))$ is strictly decreasing, so o_1 will be identified correctly at line 4. Now ρ has to look as follows: $\rho = \rho_0 \cdot o_1 \cdot \rho'$ for some walks ρ_0 and ρ'. Due to Proposition 1, ρ_0 and $o_1 \cdot \rho'$ are shortest walks, so $w(\rho_0) = \mathcal{D}(s(\rho_0), t(\rho_0)) = \mathcal{D}(u, s(o_1))$ and $w(o_1 \cdot \rho') = \mathcal{D}(s(o_1), t(\rho')) = \mathcal{D}(s(o_1), v)$. Finally, we have $w(\rho) = \mathcal{D}(s(\rho), t(\rho)) = \mathcal{D}(u, v)$, because ρ is shortest, and $w(\rho) = w(\rho_0) + w(o_1 \cdot \rho')$. This means that equality at line 9 holds and the pair (u, v) will added to H and later returned by the algorithm. □

Algorithm 3: PHP(G, O)

```
   Input: A graph $G = (V, E, w)$ such that $w : E \to \mathbb{R}_+$ and a
          set of observations $O = \{o_1, \dots, o_k\}$.
   Output: The set $H(O)$ of all (source, target) pairs.
1  $H \leftarrow \emptyset$;
2  $T \leftarrow$ PTP$(G, O)$;
3  for each target $v \in T$ do
4      sort $O$ by $\mathcal{D}(t(o_i), v)$ in descending order;
5      for each possible source $u \in V$ do
6          if $\mathcal{D}(u, v) = \infty$ then
7              continue;
8          end
9          if $\mathcal{D}(u, v) = \mathcal{D}(u, s(o_1)) + \mathcal{D}(s(o_1), v)$ then
10             add $(u, v)$ to $H$
11         end
12     end
13 end
14 return $H$
```

5.1 Computational Complexity

We assume the graph has n nodes, m edges ($m \geq n$), and k observations. Since computing $\mathcal{D}(u, v)$ values is the most expensive operation, it is crucial that we calculate only those that are necessary. For a given start node u, a single call to Dijkstra's algorithm computes $\mathcal{D}(u, v)$ for all $v \in V$. Using a Fibonacci heap as the priority queue, a single run of Dijkstra's algorithm takes $\mathcal{O}(m + n \log n)$ time. It should be noted that Fibonacci heaps are rarely used in practice due to large constant factors, and binary heaps are typically preferred despite their slightly worse theoretical performance. In particular, the NetworkX library, used in our experiments (Section 7), implements Dijkstra's algorithm with a binary heap. With a binary heap, the time complexity increases to $\mathcal{O}(m \log n + n \log n) = \mathcal{O}(m \log n)$.

POSSIBLE-TARGET-PROBLEM: Notice that Algorithm 1 only needs to calculate $\mathcal{D}(u, \cdot)$ for u that are either the start or the end of any observation. This leads to $\mathcal{O}(k)$ calls to Dijkstra's algorithm. We assume that all these values are computed at the beginning and simply retrieved later in $\mathcal{O}(1)$ time. So all together this pre-computation takes $\mathcal{O}(km \log n)$ time.

Next, the algorithm need to iterate over $\mathcal{O}(n)$ possible targets. This is because the distance values depend on the specific target, and the sorted order of observations can vary significantly between different targets. For each target the algorithm sorts a list of k values, in time $\mathcal{O}(k \log k)$, and does equality checks that require $\mathcal{O}(k)$ additions. So this part costs $\mathcal{O}(n(k \log k + k)) = \mathcal{O}(kn \log k)$ time.

Put together that gives $\mathcal{O}(km \log n + kn \log k)$ running time. For sparse graphs (i.e., $m = \mathcal{O}(n)$) we get that this complexity becomes $\mathcal{O}(kn(\log n + \log k))$.

When a graph is a directed acyclic graph (DAG), then we can use topological sort followed by a single forward pass instead of Dijkstra's algorithm. This has complexity $\mathcal{O}(m)$, and so the pre-computation of all relevant $\mathcal{D}(u, \cdot)$ takes $\mathcal{O}(km)$ time. The rest stays the same, so we get $\mathcal{O}(km + kn \log k)$. For sparse DAGs this reduces to $\mathcal{O}(kn \log k)$.

POSSIBLE-SOURCE-PROBLEM: The running time is the same as above apart from one additional step of reversing the graph and the set of observations at the beginning. That can be done in $\mathcal{O}(m + k)$ time. As this is dominated by the running time of the other steps, the computational complexity for each graph class is exactly the same as above.

SOURCE-TARGET-PROBLEM: Here the situation changes, because we may need to calculate $\mathcal{D}(u, \cdot)$ for all $u \in V$. Computation of all of these costs $\mathcal{O}(nm \log n)$. The rest of the computation is $\mathcal{O}(n)$ times sorting $\mathcal{O}(k)$ values, and $\mathcal{O}(n^2)$ times equality checking and additions. All together that gives $\mathcal{O}(nm \log n + nk + n^2) = \mathcal{O}(nm \log n)$. For sparse graphs this reduces to $\mathcal{O}(n^2 \log n)$.

For DAGs we would have $\mathcal{O}(nm)$ for the computation of $\mathcal{D}(u, \cdot)$ for all $u \in V$. This leads to $\mathcal{O}(nm + nk + n^2) = \mathcal{O}(nm)$. For sparse DAGs, this reduces to $\mathcal{O}(n^2)$.

6 Graphs with No Negative Cycles

In this section, we show how we can deal with these computational problems for the most general class of graphs in which they are well-defined, i.e., graphs with no negative cycles. Note that negative costs can occur naturally, e.g., in the context of financial transition when an agent earns money by performing certain actions. The central idea behind this algorithm is the construction of the graph $G_v^O = (O, E')$ such that $(o, o') \in E'$ iff $\mathcal{D}(s(o), v) = w(o) + \mathcal{D}(t(o), s(o')) + w(o') + \mathcal{D}(t(o'), v)$. Notice that this condition is the same as in Theorem 3 for the existence of a shortest walk that terminates at v and is consistent with the sequence of two observations (o, o'). Intuitively, we will be trying to create a shortest walk that visits each observations at least once and this graph tells us which observation can follow which in a shortest walk.

This brings us to establishing an interesting problem of independent interest: determining whether a directed graph contains a walk that visits each node at least once (*Hamiltonian walk*). While related to the well-known *Hamiltonian path* problem, which looks for a walk that visits each node *exactly* once, the two problems are fundamentally different. The Hamiltonian path problem is a well-known NP-complete problem, whereas we show Hamiltonian walk to be solvable in linear time using Algorithm 4. Interestingly, similar graph structural properties that we prove here were used to prune the search space when looking for a Hamiltonian path in [8]. This is because non-existence of a Hamiltonian walk implies non-existence of a Hamiltonian path.

Algorithm 4: ONE-WALK-COVER(G)

```
  Input: A directed graph $G = (V, E)$.
  Output: Decide if there exists a walk that visits every node of
          $G$ at least once.
1 Decompose $G$ into strongly connected components and
    topologically sort them to get $(C_1, \dots, C_l)$;
2 if every component $C_i$ has an edge to $C_{i+1}$ then
3     return YES ;
4 else
5     return NO ;
6 end
```

Theorem 9. *Given a directed graph $G = (V, E)$, Algorithm 4 checks in linear time if there exists a single walk that visits each node in G at least once.*

Proof. Suppose that the algorithm returns YES. In this case, we can create a walk as follows. We start such a walk with an arbitrary node of the first strongly connected component (SCC) as listed in the topological order. Then as long as there is a node not visited in the current SCC, we append a walk that reaches this node from the current node. As this is an SCC, such a walk has to exist. Once all the nodes are visited in this SCC, we follow this by a walk to a node that has an edge that leads to the second SCC in the topical order. We repeat the process and visit all nodes in the second SCC, moving onto the third SCC, etc. It is clear to see that, at the end, this walk will visit all nodes in this graph at least once.

Now, suppose that the algorithm returns NO. In this case, there exist two SCCs C_1 and C_2 such that C_1 cannot be reached from C_2, nor C_2 can be reached from C_1. This means that there exist two nodes in this graph, u and v, such that there is no walk from u to v nor from v to u. But then no single walk that visits both u and v at least once can exist, because that would contract the previous statement.

Graph decomposition into SCCs, topological ordering of a graph, and checking if each SCC has an edge to the next, according to the topological ordering, SCC can all be done in $\mathcal{O}(n + m)$ time. □

Algorithm 5: NHP(G, O)

```
Input: A graph $G = (V, E, w)$ with no negative cycles and a
    set of observations $O = \{o_1, \ldots, o_k\}$.
Output: The set $H(O)$ of all (source, target) pairs.
1  $H \leftarrow \emptyset; T \leftarrow \emptyset$;
2  for each possible target $v \in V$ do
3      for every observation $o \in O$ do
4          if $\mathcal{D}(s(o), v) = \infty$ or
             $\mathcal{D}(s(o), v) \neq w(o) + \mathcal{D}(t(o), v)$ then
5              continue;
6          end
7      end
8      Build the graph $G_v^O = (O, E')$ such that $(o, o') \in E'$ iff
         $\mathcal{D}(s(o), v) =$
         $w(o) + \mathcal{D}(t(o), s(o')) + w(o') + \mathcal{D}(t(o'), v)$.
9      if ONE-WALK-COVER$(G_v^O)$ then
10         add $v$ to $T$;
11         Decompose $G_v^O$ into SCCs and topologically sort them
             to get $(C_1, \ldots, C_l)$ ;
12         for each possible source $u \in V$ do
13             if $\mathcal{D}(u, v) = \infty$ then
14                 continue;
15             end
16             for every observation $o_1 \in C_1$ do
17                 if $\mathcal{D}(u, v) = \mathcal{D}(u, s(o_1)) + \mathcal{D}(s(o_1), v)$ then
18                     add $(u, v)$ to $H$
19                 end
20             end
21         end
22     end
23 end
24 return $H$
```

We now present Algorithm 5 that computes $H(O)$. If we are only interested in computing $T(O)$ then we can skip lines 11–21. If we are only interested in $S(O)$, we first reverse the graph G and the set of observations O (like we did in Section 5) and then skip exactly the same lines. We now show the correctness of this algorithm, followed by its computational complexity analysis.

Theorem 10. *Algorithm 5 solves* SOURCE-TARGET-PROBLEM *for a given G with no negative cycles and observation set O.*

Proof. ($\Rightarrow$) Suppose that this algorithm returns (u, v) as a possible source-target pair. Let $(o_1, \ldots, o_l)$ be the sequence of nodes in the single walk in G_v^O that visits all its nodes. Note that this sequence can repeat the same observations. Let us consider a walk $\rho = o_1 \cdot \rho_1 \cdot o_2 \cdot \rho_2 \cdot \ldots o_l \cdot \rho_l$ such that ρ_i is a shortest walk from $t(o_i)$ to $s(o_{i+1})$ for all $i < l$, and so of weight $\mathcal{D}(t(o_i), s(o_{i+1}))$, and ρ_l is a shortest walk from $t(o_l)$ to v.

Note that due to line 4, $\mathcal{D}(s(o_i), v) = w(o_i) + \mathcal{D}(t(o_i), v))$ for all i holds. Also, $\mathcal{D}(s(o_i), v) = w(o_i) + \mathcal{D}(t(o_i), s(o_{i+1})) + w(o_{i+1}) + \mathcal{D}(t(o_{i+1}), v)$ have to hold for all $i < l$, because of how G_v^O was defined.

We now have $w(\rho) = \sum_{i=1}^{l}(w(o_i) + w(\rho_i)) = \sum_{i=1}^{l-2}(w(o_i) + w(\rho_i)) + w(o_{l-1}) + \mathcal{D}(t(o_{l-1}), s(o_l)) + w(o_l) + \mathcal{D}(t(o_l), v) = \sum_{i=1}^{l-2}(w(o_i) + w(\rho_i)) + \mathcal{D}(s(o_{l-1}), v) = \sum_{i=1}^{l-3}(w(o_i) + w(\rho_i)) + w(o_{l-2}) + \mathcal{D}(t(o_{l-2}), s(o_{l-1})) + w(o_{l-1}) + \mathcal{D}(t(o_{l-1}), v) = \sum_{i=1}^{l-3}(w(o_i) + w(\rho_i)) + \mathcal{D}(s(o_{l-2}), v) = \ldots = \mathcal{D}(s(o_1, v)$. This shows that ρ is a shortest walk that ends at v and is consistent with O, because all observations occur in ρ at least once.

We know there exists o_1 such that $\mathcal{D}(u, v) = \mathcal{D}(u, s(o_1)) + \mathcal{D}(s(o_1), v)$. Let ρ' be the shortest walk from u to $s(o_1)$ and consider $\rho' \cdot \rho$. We can see it is the shortest because of the just mentioned equality, and so $(u, v) \in H(O)$.

($\Leftarrow$) Suppose $(u, v) \in H(O)$, which means that there exists a shortest walk, ρ, that starts at u and terminates at v and is consistent with the set of observations O. Thanks to Theorem 3 when applied to each single observation in ρ, we know that the conditions in line 4 hold. Note that the condition for an edge in G_v^O to exist is the same as in Theorem 3 when applied to a sequence of two observations. We claim that G_v^O has a walk that visits all nodes at least once. This is because ρ visits all observations and we can simply follow the sequence of observations as they occur in ρ and just argued an edge has to exist between them as the condition holds.

Let o_1 be the first observation visited in ρ. We clearly have to have $\mathcal{D}(u, v) = \mathcal{D}(u, s(o_1)) + \mathcal{D}(s(o_1), v)$, because ρ is a shortest walk between u and v and any point along such a walk has to be a concatenation of two shortest walks. Also, o_1 has to belong to the SCC of G_v^O that is first in topological order, because otherwise we would not be able to visit some observations in earlier SCCs.

All of this shows that Algorithm 5 will return (u, v) as a source-target pair as it should. □

6.1 Computational Complexity

We can no longer use Dijkstra algorithm for calculating $\mathcal{D}(u, \cdot)$, because it requires all edge weights to be non-negative. Instead we make use of Bellman-Ford algorithm for which the complexity of a single call is $\mathcal{O}(nm)$ time. To calculate all-pairs shortest distances, it is best to use Johnson's algorithm instead whose running time is $\mathcal{O}(nm + nm \log n) = \mathcal{O}(nm \log n)$.

In the case of POSSIBLE-TARGET-PROBLEM and POSSIBLE-SOURCE-PROBLEM, we just need to calculate $\mathcal{D}(u, \cdot)$ for all u that are either a start or end of an observation. That gives $\mathcal{O}(nm \cdot \min(k, \log n))$ complexity when we use Bellman-Ford for $k \leq \log n$

Table 1: Average size of the set of sources and targets for different graphs and observations set sizes

			Average size of sources set vs #observations				**Average size of targets set vs #observations**			
Graph	$\|N\|$	$\|E\|$	4	8	16	32	4	8	16	32
# Minnesota	2642	6606	179.32	93.76	42.96	30.00	187.44	128.80	105.72	78.24
# Europe	1174	2834	79.48	25.96	8.00	2.56	52.16	12.28	3.56	1.88
# Luxembourg	114,599	239,332	7742.96	3928.60	981.56	232.64	33,631.08	15,351.40	7304.84	1283.28
# US roads	129,164	330,870	46,881.24	38,077.00	23,520.68	17,775.20	1260.00	682.88	342.36	65.68
# Pennsylvania	1,087,562	3,083,028	24,993.56	794.24	342.88	196.76	42,036.40	8333.28	6653.00	615.76
# Belgium	1,441,295	3,099,940	92,749.72	49,993.20	1602.04	966.76	67,817.32	46,420.00	3295.84	272.32
# Italy	6,686,493	14,027,956	8332.24	6451.40	3365.88	2015.60	2,181,188.88	516,536.32	70,806.88	30,569.40
# Great britain	7,733,822	16,313,034	142,739.12	39,085.08	34,745.28	5285.60	139,910.20	59,266.68	43,329.44	19,857.60

and use Johnson's algorithm otherwise. For sparse graphs that becomes $\mathcal{O}(n^2 \cdot \min(k, \log n))$. In case of DAGs we can compute these in $\mathcal{O}(km)$ time. In case of sparse DAGs that becomes $\mathcal{O}(kn)$. For SOURCE-TARGET-PROBLEM using Johnson's algorithm gives $\mathcal{O}(nm \log n)$, which is $\mathcal{O}(n^2 \log n)$ for sparse graphs. In case of DAGs we can compute these in $\mathcal{O}(nm)$ time, and for sparse DAGs we get $\mathcal{O}(n^2)$.

As for the other computation steps, we build $\mathcal{O}(n)$ graphs G_v^O that each takes $\mathcal{O}(k^2)$ time. Also, we call $\mathcal{O}(n)$ times ONE-WALK-COVER that each takes $\mathcal{O}(m)$. We decompose $\mathcal{O}(n)$ graphs G_v^O, each of which takes $\mathcal{O}(k^2)$. Finally, we check at most $\mathcal{O}(n^2 k)$ times the equality in line 17. Overall, that gives $\mathcal{O}(nk^2 + nm + nk^2 + n^2 k) = \mathcal{O}(nk^2 + nm + n^2 k)$. For sparse graphs that becomes $\mathcal{O}(nk(n+k))$.

The overall running time is just the sum of these two parts.

7 Experiments

In this section we evaluate the effectiveness of the proposed methods across real-world road networks of varying scales from 1K to 7.7 millions nodes. Datasets details can be found in the Appendix. Experiments were conducted on eight datasets from the Network Data Repository [23]. All datasets originate from the DIMACS10 challenge and were initially unweighted undirected road networks. The smallest instance, denoted as #Europe, comprises 1,174 nodes, while the largest network #Great Britain contains over 7 million nodes and 14 million edges. We first transformed the graphs into equivalent directed ones, by replacing each undirected edge with bidirectional edges between the same two nodes. Next, we assigned every edge with a weight uniformly at random between 1 and 1000. All algorithms were executed multiple times to estimate overall running time performance with standard deviation measurements. The Python-based implementation ran on an Intel Core i7-14650HX CPU (2.20 GHz, 32GB RAM) using a single core. We first identified random shortest walks containing more than 64 edges. For the inference tasks, we randomly sampled 4, 8, 16, or 32 observations from these walks to serve as input for each execution of the algorithm. In the end, the average running time and average size of sources and targets are evaluated.

We did not compare against, e.g. a simple enumeration of all shortest paths approach, as our graphs are too large for that to be feasible.

7.1 Results

Figure 2 compares algorithm running times across datasets grouped into 4 levels by their size: Level 1 (Europe, Minnesota): thousands of nodes, Level 2 (Luxembourg, US road): about 10K nodes, Level 3 (Pennsylvania, Belgium): around 1M nodes and Level 4 (Italy, Great Britain): over 7M nodes. The first row in Figure 2 shows Sources search time, and the second row shows Targets search time. At Level 1, both methods perform similarly. As graphs grow larger, both show nearly identical growth rates. However, Sources search requires graph reversal, making it slightly slower. All the plots demonstrate near-linear dependence on the number of observations, aligning with theoretical expectations.

Table 1 illustrates the average size of possible sources set $S(O)$ and targets set $T(O)$ for a given observation set sizes 4, 8, 16 and 32. Note that, as one would expect, the average size of $S(O)$ and $T(O)$ decrease as the number of observations increase, because that reduces ambiguity. Sometimes the drop in these sizes can be very sharp for the larger graphs. Interestingly, there is notable asymmetry between source and target set sizes for some of these graphs, suggesting differences in structural directionality (recall that each undirected edge was replaced by two directed edges but possibly with a very different weight). Moreover, graph size alone does not determine inference difficulty: small graphs sometimes yield similar set sizes as larger ones, indicating that topology plays a critical role.

8 Conclusions

In this paper we addressed the source and target prediction problems in a weighted graph under the shortest walk assumption. We showed that these problems can be solved in polynomial time. Furthermore, we performed a detailed computational complexity analysis and considered various classes of graphs, such as positive weighted graphs, directed acyclic graphs (DAGs), sparse graphs, and sparse DAGs. Experiments on real-world road networks with millions of nodes demonstrate the scalability and practicality of our approach.

As mentioned earlier, our problems can be defined in many different ways. One can consider the set of observations to consists of nodes rather than edges. It could also be a mixture of the two. Also, the order of observations can be known or partially known. Straightforward adaptations of our algorithms can handle all such different cases.

Our work makes the first step in the algorithmic approach to trajectory prediction-type problems and their computational complexity, laying a foundation for further research into this direction. Future work could extend this framework to incorporate additional real-world constraints, such as dynamic edge weights that reflect changing conditions (e.g., traffic or delays) or probabilistic modeling of uncertainties in the observation set. Another promising direction is to explore incorporating efficient methods for shortest-path computation at scale. In large networks, such as those used in real-time navigation systems, techniques such as Contraction Hierarchies [9], Hub Labeling [1], or Customizable Route Planning [5] enable near-instantaneous shortest-path queries despite theoretical complexity limits. Integrating such techniques into our inference framework could potentially significantly improve its performance and make it applicable to real-time, large-scale trajectory analysis scenarios.

Ethical Statement

There are no ethical issues.

Acknowledgements

This work was supported by the EPSRC Centre for Doctoral Training in Distributed Algorithms and it is also partly supported by EPSRC though grants number EP/X017796/1 and EP/X03688X/1. We would like to thank the anonymous reviewers for their valuable comments, and Sven Schewe and Stephen Pasteris for insightful discussions on some aspects of this work.

References

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. Hierarchical hub labelings for shortest paths. In *European Symposium on Algorithms*, pages 24–35. Springer, 2012.

[2] F. Amin, K. Gharami, and B. Sen. Trajectoformer: Transformer-based trajectory prediction of autonomous vehicles with spatio-temporal neighborhood considerations. *International Journal of Computational Intelligence Systems*, 17(1), Apr 2024. doi: 10.1007/s44196-024-00410-1.

[3] B. Chang, Y. Park, D. Park, S. Kim, and J. Kang. Content-aware hierarchical point-of-interest embedding model for successive poi recommendation. In *IJCAI*, volume 20, pages 3301–3307, 2018.

[4] Z. Cheng, M. Trépanier, and L. Sun. Probabilistic model for destination inference and travel pattern mining from smart card data. *Transportation*, 48(4):2035–2053, 2021.

[5] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10*, pages 376–387. Springer, 2011.

[6] N. Deo, E. Wolff, and O. Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. In *Conference on Robot Learning*, pages 203–212. PMLR, 2022.

[7] J. Eisner, S. Funke, A. Herbst, A. Spillner, and S. Storandt. Algorithms for matching and predicting trajectories. In *2011 Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 84–95. SIAM, 2011.

[8] J.-G. Fages and X. Lorca. Improving the asymmetric tsp by considering graph structure. *arXiv preprint arXiv:1206.3437*, 2012.

[9] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

[10] J. He, X. Li, and L. Liao. Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking. In *IJCAI*, volume 17, pages 1837–1843, 2017.

[11] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.

[12] A. Ip, L. Irio, and R. Oliveira. Vehicle trajectory prediction based on lstm recurrent neural networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5. IEEE, 2021.

[13] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path prediction and predictive range querying in road network databases. *The VLDB Journal*, 19:585–602, 2010.

[14] G. Li, Y. Chen, Q. Liao, and Z. He. Potential destination discovery for low predictability individuals based on knowledge graph. *Transportation Research Part C: Emerging Technologies*, 145:103928, 2022.

[15] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang, and L. Pan. Predicting path failure in time-evolving graphs. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1279–1289, 2019.

[16] Y. Li, Y. Jiang, and X. Wu. Trajpt: A trajectory data-based pre-trained transformer model for learning multi-vehicle interactions. *Transportation Research Part C: Emerging Technologies*, 171:105013, 2025. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2025.105013. URL https://www.sciencedirect.com/science/article/pii/S0968090X25000178.

[17] C. Liao, C. Chen, C. Xiang, H. Huang, H. Xie, and S. Guo. Taxi-passenger's destination prediction via gps embedding and attention-based bilstm model. *IEEE Transactions on Intelligent Transportation Systems*, 23(5):4460–4473, 2021.

[18] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1043–1051, 2013.

[19] D. D. Ndunguru, F. Xing, C. Z. Oroni, A. Ani, and C. Li. Stg-lstm: Spatial-temporal graph-based long short-term memory for vehicle trajectory prediction. *Multimodal Transportation*, 4(3):100222, 2025. ISSN 2772-5863. doi: https://doi.org/10.1016/j.multra.2025.100222. URL https://www.sciencedirect.com/science/article/pii/S277258632500036X.

[20] M. O'Connell, moreiraMatias, and W. Kan. Ecml/pkdd 15: Taxi trajectory prediction (i). https://kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i, 2015. Kaggle.

[21] G. Qin, L. Song, Y. Yu, C. Huang, W. Jia, Y. Cao, and J. Dong. Graph structure learning on user mobility data for social relationship inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4578–4586, 2023.

[22] Y. Ren, Y. Xiao, Y. Zhou, Z. Zhang, and Z. Tian. Cskg4apt: A cybersecurity knowledge graph for advanced persistent threat organization attribution. *IEEE Transactions on Knowledge and Data Engineering*, 35 (6):5695–5709, 2022.

[23] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL https://networkrepository.com.

[24] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[25] X. Song, K. Chen, X. Li, J. Sun, B. Hou, Y. Cui, B. Zhang, G. Xiong, and Z. Wang. Pedestrian trajectory prediction based on deep convolutional lstm network. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3285–3302, 2020.

[26] D. Su, X. Li, Z. Li, Y. Liao, R.-H. Li, and G. Wang. Dirw: Path-aware digraph learning for heterophily. *arXiv preprint arXiv:2410.10320*, 2024.

[27] H. Su, J. Zhu, Y. Dong, and B. Zhang. Forecast the plausible paths in crowd scenes. In *IJCAI*, volume 1, page 2, 2017.

[28] H. Sun, J. Xu, K. Zheng, P. Zhao, P. Chao, and X. Zhou. Mfnp: A meta-optimized model for few-shot next poi recommendation. In *IJCAI*, volume 2021, pages 3017–3023, 2021.

[29] K. Sun, T. Qian, T. Chen, Y. Liang, Q. V. H. Nguyen, and H. Yin. Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 214–221, 2020.

[30] Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao. M2i: From factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6543–6552, 2022.

[31] L. Ullrich, A. McMaster, and K. Graichen. Transfer learning study of motion transformer-based trajectory predictions*. In *2024 IEEE Intelligent Vehicles Symposium (IV)*, pages 110–117, 2024. doi: 10.1109/IV55156.2024.10588422.

[32] X. Wang, G. Sun, X. Fang, J. Yang, and S. Wang. Modeling spatio-temporal neighbourhood for personalized point-of-interest recommendation. In *IJCAI*, pages 3530–3536, 2022.

[33] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang. Modeling trajectories with recurrent neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 3083–3090. AAAI Press, 2017. ISBN 9780999241103.

[34] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut. Taxi dispatch planning via demand and destination modeling. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 377–384. IEEE, 2018.