Time-Based Similar Trajectories on Graphs

Shima Moghtasedi

University of Pisa, Pisa, Italy
shima.moghtasedi@di.unipi.it

Abstract. Graphs can be used to represent not only road networks but also routing networks and computer networks. In such kind of applications, entities, such as packets, are moving on the network and the information associated with these movements are known regarding to the vertices they traverse. In such a context a trajectory for these entities can be defined as sequences of vertices and time intervals. We aim at designing fast algorithms for finding all the trajectories which are similar to (i.e. moving close to) a given trajectory in a specified time window.

Keywords: graph trajectory · similarity · data structures

1 Introduction

In this work, we are interested in algorithms to efficiently retrieve similar trajectories to a given one for an undirected graph. Many papers in literature have focused on extracting information from sets of trajectories, basically, looking at them as strings, so the similarity between trajectories corresponds to similarity between strings. In such a context, the similarity measure have been formalized by means of some concepts like longest common subsequences [8]. We aim at exploiting here the topology of the network, assessing that two trajectories are similar if they traverse "close" vertices in the same time interval. More formally, in our case, two trajectories are similar at time t if they are passing through either the same vertex or two neighbor vertices. We say that two trajectories are similar in a time interval T if they are similar for each time $t \in T$. We also say that a trajectory is close to a vertex v if the trajectory goes through either v or a neighbor of v. Using these notions of similarity, we aim at solving the following problems.

Problem 1. Given a graph, a set of trajectories, a trajectory x, and a time interval, find the subset of trajectories which are similar to x in the specified time interval.

Problem 2. Given a vertex v of a graph G, a set of trajectories, and a time interval, find all the trajectories close to v for the whole given time interval.

Related work. Some papers have focused on finding trajectories moving together in a certain area of the plane in some applications like traffic monitoring, e-marketing and route finding [9, 6, 7]. These proposed algorithms do not extend

to the case when the movements are constrained to be walks on a graph. Some papers in this area, focus to cluster the segments of trajectories on the plane [2, 4]. On the other hand, there are some papers answering topological based queries, specifically on trajectories and graphs [3, 5, 10], but they do not extend smoothly to our problem. Among them, the result by Luo et al. [3] finds all the trajectories arriving to a given vertex v during time interval $T = [a, b]$, under the restriction that each trajectory does not traverse the same vertex twice or more. In particular, they find all the trajectories y there exists a time $t \in [a, b]$ such that y passes by v. Although close, this result is not suitable for studying our problem, as we would need to do search in their structure for each $t \in [a, b]$, to find trajectories passing v or close to v for the whole time interval T.

Our contribution. In this paper, we propose a method, to solve our problems which consists of the following two phases.

The *preprocessing* phase: given a graph and a set of trajectories, prepare suitable data structures to maintain trajectories in order to answer *queries*. This phase is the same for both problems.

The *query* phase:

- given a vertex and a time interval, use the data structures built in the preprocessing phase to find all trajectories traversing close to the vertex within the given time interval, which corresponds to Problem 2
- given a trajectory and a time interval, use the solutions of Problem 2 to answer queries for Problem 1.

2 Definitions and Algorithm Overview

Definition 1 (Trajectory). *Given an undirected graph $G = (V, E)$, a trajectory x is a sequence $x = \langle (v_1, T_1), (v_2, T_2), \cdots, (v_l, T_l) \rangle$ such that for each $i \in \mathbb{N}, 1 \leq i \leq l-1$, we have $(v_i, v_{i+1}) \in E$ and T_i is a time interval $[a_i, b_i]$ with $a_i, b_i \in \mathbb{N}$, and $a_i \leq b_i < a_{i+1} \leq b_{i+1}, b_i + 1 = a_{i+1}$. We call l, i.e. the length of x, as number of movements.*

Given a set of trajectories $\mathcal{T}$, let us define the projection set S_v for each $v \in V$, as $\{(T, x) | (v, T) \in x\}$ (i.e. (v, T) appears in x). As we can see in the following example, there can be more than one pair associated with a vertex for a given trajectory, since each trajectory can traverse a vertex multiple times during its lifetime. For a trajectory x and a time $t \in \mathbb{N}$, the notation $x(t)$ indicates the vertex $v \in V$ such that x contains (v, T) and $t \in T$. For each vertex $v \in V$, we denote N_v as the set of neighbors of vertex v and $\{v\} \cup N_v$ as B_v.

Definition 2 (Time Constrained Similar Trajectories). *Given two trajectories x, y and time $t \in \mathbb{N}$, let u and v be vertices $x(t)$ and $y(t)$, respectively. We say x, y are similar at time t, if either $u = v$ or $u \in N_v$. For a given time interval T, x and y are similar in T if they are similar for each time $t \in T$.*

We explain the definitions by giving a simple example.

Example 1 *Graph* $G = (V, E), V = \{v_1, \dots v_4\}$ *and a set of two trajectories* $\mathcal{T} = \{x_1, x_2\}$ *are shown in Figure 1. Consider* $(v_1, [1, 7])$ *in* x_1*. This means that trajectory* x_1 *have traversed through the vertex* $v_1 \in G$ *for each time* $t \in [1, 7]$.

$x_1 = \langle (v_1, [1, 7]), (v_2, [8, 9]), (v_4, [10, 13]), (v_1, [14, 17]), (v_3, [18, 20]) \rangle$

$x_2 = \langle (v_1, [1, 4]), (v_2, [5, 7]), (v_3, [8, 9]) \rangle$

	x_1	x_2
v_1	$[1, 7], [14, 17]$	$[1, 4]$
v_2	$[8, 9]$	$[5, 7]$
v_3	$[18, 20]$	$[8, 9]$
v_4	$[10, 13]$	

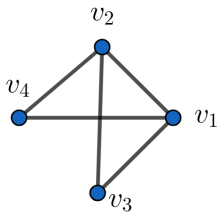

Fig. 1. An example of set of trajectories $\mathcal{T} = \{x_1, x_2\}$ on the graph G; Each row of the table indicates $S_v, v \in V$, implicitly.

For a given trajectory x_2 *and the time interval* $T = [3, 6]$ *in our example, the trajectory* x_1 *is similar to* x_2 *at time interval* T*, since for each time* $t \in T$ *they are at neighbor vertices* v_1 *and* v_2*. For instance,* x_1 *is at* v_1 *during the time interval* $[3, 6]$*, and* x_2 *is at* v_1 *at times* $t = 3, 4$ *and at* v_2 *for* $t = 5, 6$*. In our given example, in Figure 1, we have* $S_{v_1} = \{([1, 7], x_1), ([14, 17], x_1), ([1, 4], x_2)\}$.

Preprocessing and Query Definition. For both Problems 1 and 2, we define PREPROCESSING$(G, \mathcal{T})$, where $G = (V, E)$ is a graph and $\mathcal{T}$ is a set of trajectories. Given G and $\mathcal{T}$, we aim at answering the following queries, which corresponds to Problem 1 and 2, respectively.

QUERY$_1(x, T)$. Given a trajectory x, and a time interval $T = [a, b]$, find the set of trajectories which are similar to x in the time interval.

QUERY$_2(v, T)$. Given a vertex v, and a time interval $T = [a, b]$, find the set of trajectories, traversing vertices in B_v for each time $t \in T$.

Lemma 1. *Suppose we can solve* QUERY$_2(v_i, T_i)$ *in time* $O(Z)$ *for some* $Z(*)$*. By using its solution,* QUERY$_1(x, T = [a, b])$ *can be answered in time* $O(\log l + h \cdot Z)$*, where* h *is the number of movements in* x *within the time interval* T.

Proof. Let H be the set of pairs (v_i, T_i) in x, with $T_i = [a_i, b_i]$ such that $b \geq a_i$ and $a \leq b_i$. We have $h = |H|$. We run QUERY$_2(v_i, T_i)$ for each pair $(v_i, T_i) \in H$. Let U_{v_i} be the output set of QUERY$_2(v_i, T_i)$, then we report $\bigcap_{i=1}^{h} U_{v_i}$ as the solution set for QUERY$_1(x, T)$. We now prove that this algorithm is correct. For each pair (v_i, T_i) in x, trajectory x is at vertex v_i during time interval T_i (i.e. $\forall t \in T_i, x(t) = v_i$). This means that all trajectories, traversing either v_i or N_{v_i} for each time instance $t \in T_i$, are similar to x in a time interval T_i (Definition 2). By definition of QUERY$_2$, these are returned by QUERY$_2(v_i, T_i)$. As the set of trajectories similar to x in $T = \bigcup_{i=1}^{h} T_i$ is the intersection of trajectories similar to x in each T_i with $1 \leq i \leq h$, we return the intersection of the sets U_{v_i}. As x is

a time ordered sequence of pairs, we can find the set H easily in time $O(\log l+h)$. Since, we run $\text{QUERY}_2(v_i, T_i)$ for each pair $(v_i, T_i) \in H$, we pay $O(h \cdot Z)$. The cost of the intersection is dominated by this cost.

Hence, in the following we will concentrate on solving QUERY_2. In our solution $\text{PREPROCESSING}(G, \mathcal{T})$ is the same for both the problems.

Preprocessing Phase. Recall that for each vertex $u \in V$, S_u is the set of trajectories from $\mathcal{T}$ that pass through u with the corresponding time intervals. Similarly, here we define S_{B_u} for each vertex $u \in V$ as $S_{B_u} = \bigcup_{w \in B_u} S_w$. Let $|S_{B_u}|$ denotes the size of the set S_{B_u},which is the number of movements that trajectories have done within the vertices in B_u. Given a set S_{B_u}, we define $C(S_{B_u})$, a refinement of S_{B_u}, which basically considers time intervals spent by a trajectory in B_u as a unique interval in $O(|S_{B_u}| \cdot |N_u|)$ time. More formally, assume y is a trajectory passing through B_u, i.e. y appears in S_{B_u}. If we have consecutive pairs $\langle (u_i, T_i), (u_{i+1}, T_{i+1}) \rangle$ belong to y, such that $u_i, u_{i+1} \in B_u$, we have $\langle (T_i, y), (T_{i+1}, y) \rangle \in S_{B_u}$, so we consider $(T_i \cup T_{i+1}, y) \in C(S_{B_u})$. For instance, assume we are given $v_3 \in V$ in our graph example in Figure 1, and $B_{v_3} = \{v_1, v_2, v_3\}$. For the trajectory $x_1 = \langle (v_1, [1, 7]), (v_2, [8, 9]) \rangle$, we consider $([1, 9], x_1) \in C(S_{B_{v_3}})$. Our preprocessing phase builds an *interval tree* [1] representation of $C(S_{B_u})$ for each $u \in V$, considering all time intervals $\{T_i | \exists x \in \mathcal{T} s.t\ (T_i, x) \in C(S_{B_u})\}$. Briefly, interval tree is a binary tree to store a set of intervals based on the median of the intervals endpoints. In this structure all the intervals intersect the median point are stored at the root of the tree. Those intervals lying completely to the left and right of the median point, are stored at the left subtree and right subtree, respectively. These subtrees are constructed recursively in the same way. The interval tree associated with a given vertex u helps to find for any given query interval $T = [a, b]$, the set of intervals in $C(S_{B_u})$ that are contained in $T = [a, b]$. Moreover, intervals are endowed with the trajectories they belong to, by definition of $C(S_{B_u})$. Letting $|C(S_{B_u})| = r_u$, an interval tree on the set of intervals in $C(S_{B_u})$ can be built in $O(r_u \log r_u)$ time, using $O(r_u)$ space. The interval tree has depth $O(\log r_u)$. As a result, we obtain the following lemma.

Lemma 2. $\text{PREPROCESSING}(G, \mathcal{T})$ *takes* $O(\Sigma_{u \in V}(|S_{B_u}| \cdot |N_u|) + r_u \log r_u)$ *time and uses* $O(\Sigma_{u \in V} r_u)$ *space.*

2.1 Solving $\text{QUERY}_2(v, T)$

Given a vertex v, and a time interval $T = [a, b]$, we want to find the set of trajectories, traversing the vertices in B_v at each time instance $t \in T$. We will make use of the interval trees built in $\text{PREPROCESSING}(G, \mathcal{T})$.

Query Phase. Consider the interval tree associated with $C(S_{B_v})$ which maintains the time intervals endowed with the trajectories passing B_v they belong to. We obtain all the trajectories traversing B_v for each time $t \in T$, by finding all the

intervals which are fully contained in T. As the interval tree associated with v has depth $O(\log r_v)$, the cost of retrieving all the intervals fully contained in T is $O(\log r_v + |U_v|)$, where $|U_v|$ is the number of reported trajectory id's. Indeed, notice that we cannot have two time intervals belonging to the same trajectory in the output set, since we are reporting the intervals fully contained in T.

Theorem 1. *For a vertex v and a time interval T, we can answer* $\text{QUERY}_2(v, T)$ *in* $O(\log r_v + |U_v|)$ *time, where* $|U_v|$ *is the number of reported trajectories.*

Let $K = \Sigma_{v_i|(v_i,T_i)\in H}|U_{v_i}|$, by applying Lemma 1, we obtain the following.

Corollary 1. $\text{QUERY}_1(x, T)$ *can be answered in* $O(\log l + \Sigma_{v_i|(v_i,T_i)\in H} \log r_{v_i} + K)$ *time.*

As future work, we plan to enhance our algorithm to reduce the storage usage and consider different similarity measures.

Acknowledgments. The author wishes to thank R. Grossi and A. Marino for helpful discussions.

References

1. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.C.: Computational geometry. In: Computational geometry, pp. 1–17. Springer (2000)
2. Li, Y., Han, J., Yang, J.: Clustering moving objects. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 617–622. ACM (2004)
3. Luo, W., Tan, H., Chen, L., Ni, L.M.: Finding time period-based most frequent path in big trajectory data. In: Proceedings of the 2013 ACM SIGMOD international conference on management of data. pp. 713–724. ACM (2013)
4. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. Journal of Intelligent Information Systems **27**(3), 267–289 (2006)
5. Shang, S., Chen, L., Wei, Z., Jensen, C.S., Zheng, K., Kalnis, P.: Trajectory similarity join in spatial networks. Proceedings of the VLDB Endowment **10**(11), 1178–1189 (2017)
6. Surynek, P.: An application of pebble motion on graphs to abstract multi-robot path planning. In: Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on. pp. 151–158. IEEE (2009)
7. Tao, Y., Papadias, D.: Efficient historical r-trees. In: Scientific and Statistical Database Management, 2001. SSDBM 2001. Proceedings. Thirteenth International Conference on. pp. 223–232. IEEE (2001)
8. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: Data Engineering, 2002. Proceedings. 18th International Conference on. pp. 673–684. IEEE (2002)
9. Wang, L., Zheng, Y., Xie, X., Ma, W.Y.: A flexible spatio-temporal indexing scheme for large-scale gps track retrieval. In: Mobile Data Management, 2008. MDM'08. 9th International Conference on. pp. 1–8. IEEE (2008)
10. Wei, L.Y., Zheng, Y., Peng, W.C.: Constructing popular routes from uncertain trajectories. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 195–203. ACM (2012)