Trajectory Comparison in a Vehicular Network I: Computing a Consensus Trajectory

Peng Zou[1], Letu Qingge[2], Qing Yang[3], and Binhai Zhu[1]

[1] Gianforte School of Computing, Montana State University, Bozeman, MT, 59717-3880, USA. Email: peng.zou@msu.montana.edu, bhz@montana.edu
[2] College of Computing and Informatics, University of North Carolina at Charlotte, Charlotte, NC 28223, USA. Email: qingge1231@gmail.com.
[3] Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207, TX 76207-7102, USA. Email: qing.yang@unt.edu.

Abstract. In this paper, we investigate the problem of computing a consensus trajectory of a vehicle giving the history of Points of Interest (POIs) visited by the vehicle over certain period of time. The problem originates from building the social connection between two vehicles in a vehicular network. Formally, given a set of m trajectories (sequences S_i's over a given alphabet Σ, each with length at most $O(n)$, with $n = |\Sigma|$), the problem is to compute a target (median) sequence T over Σ such that the sum of similarity measure (i.e., number of adjacencies) between T and all S_i's is maximized. For this version, we show that the problem is NP-hard and we present a simple factor-2 approximation. If T has to be a permutation, then we show that the problem is still NP-hard but the approximation factor can be improved to 1.5. We implement the greedy algorithm and a variation of it which is based on a more natural greedy search. Using simulated data over two months (e.g., $n = 60$) and variants of m and Σ (e.g., $30 \leq m \leq 50$ and $30 \leq |\Sigma| \leq 60$), the empirical results are very promising and with the local adjustment algorithm the actual approximation factor is bounded by 1.14 for all the cases.

1 Introduction

In a vehicular network, before two vehicles (nodes) communicate useful information, they need to determine the trustfulness among them while preserving their privacy. (This problem is getting more important with the advent of autonomous driving.) This security issue must be addressed due to the special safety requirement of vehicular ad hoc networks (VANET), which has a wide range of applications in traffic control, accident avoidance and parking management [5, 11].

Unique to VANET, false information dissemination and Sybil attacks are some of the critical security issues. The former could be a false information like "parking garage is full" so that the sender can turn away parking competitors. The latter could be a generation of fake identities to falter the functioning of the whole system [3, 14]. The public-key infrastructure might not be available over

a road network; hence, some kind of trust management must be maintained [16, 8, 9].

To build the initial trustfulness, the solution we propose is to use the Point of Interests (POIs) visited by each vehicle to establish some level of similarity, which serves as a starting point for determining the trustfulness of vehicles. (Later, the trustfulness computation can be enhanced using some other methods [8, 9], or simply some standard method in game theory.) In this case, to protect privacy, all sensitive information regarding the location of POIs, time a POI is visited, etc, are erased for our computation and comparison. Hence, an example of POIs is "office", "restaurant", "coffee shop", "gym", etc. (See Fig.1 for a simple example.) Certainly, we could make POIs slightly more specific without sacrificing privacy, for instance, a restaurant could be more specific, e.g., "Wendy's" or "MacDonalds" could be used.

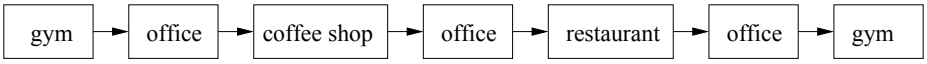

Fig. 1. The POIs visited by a white-collared professional during a typical working day. If we use A, B, C, and D to represent 'gym', 'office', 'coffee shop', and 'restaurant' respectively, then the trajectory would be represented as a sequence $\langle A, B, C, B, D, B, A\rangle$.

Now suppose that we have two list of POIs, $\mathcal{S}_1$ and $\mathcal{S}_2$, visited by two vehicles over some time period, say two months. How do we compare these sequences (of POIs)? We adopt a classic concept called *adjacency*, which is commonly used in computational genomics [1, 6, 7]. Our idea is to compute two consensus trajectories, T_1 and T_2, for each vehicle and then compare T_1 and T_2 directly. (The latter comparison could be done naively by counting the number of adjacencies between T_1 and T_2; of course, to make the computation more accurate, we could delete some redundant POIs — the latter research will be presented in a companion paper.)

Computing the consensus trajectory (sequence) is a traditional problem in computational biology, under different distance measures. For instance, given a set of DNA sequences $\{S_1, S_2, ..., S_m\}$, computing a median sequence T of length L such that $\sum_i d_H(T, S_i)$ is minimized — where $d_H()$ is the Hamming distance, is an important problem in computing conserved regions in many molecular biology problems [10]. When the input sequences are genomes and the distance measure is the *breakpoint* distance, then we have the breakpoint median problem [2, 12, 15]. It is not surprising that all these problems are NP-hard; in fact, for the breakpoint-median problem the problem is NP-hard even if there are only three input genomes [2].

The problem we investigate in this paper, while similar in some sense to these previous works, are different in several aspects. First of all, the similarity measure we use is not a distance measure; instead, it is a similarity measure featuring "the more similar, the more number of adjacencies". Secondly, the

alphabet (i.e., set of POIs) we use is not necessarily a small constant and POIs are allowed to repeat in any input sequence. Thirdly, the length of the median trajectory is bounded (as otherwise the problem becomes trivial to solve). These make the design of algorithms more challenging.

The paper is organized as follows. In Section 2, we give necessary definitions and define the corresponding problems. In Section 3, we give NP-hardness proofs for these problems. In Section 4, we present two approximation algorithms for these problems, with approximation factor 2 and 1.5 respectively. In Section 5, we implement two methods based on one of these approximation algorithms and show the empirical results. In Section 6, we conclude the paper with several open questions.

2 Preliminaries

At first, we make some necessary definitions. Given a set Σ of POIs (or just letters or nodes), a string P is called *permutation* if each element in Σ appears exactly once in P. We use *c(P)* to denote the set of elements in permutation P. A string A is called *sequence* if some POIs appear more than once in A, and $c(A)$ denotes POIs of A, which is a multi-set of elements in Σ. (Throughout this paper, we will mix the use of sequences and trajectories, with the understanding that consecutive identical letters will always be preprocessed as a single letter. For instance $A = abcdddab$ would be preprocessed as $A = abcdab$.) For example, $\Sigma = \{a, b, c, d\}$, $A = abcdacd$, $c(A) = \{a, a, b, c, c, d, d\}$. A substring with m letters (in a sequence A) is called an *m-substring*, and a 2-substring is also called a *pair*. The relative order of the two letters of a pair does not matter, i.e., the pair xy is equal to the pair yx. Given a sequence $A{=}a_1a_2a_3\cdots a_n$, let $P_A = \{a_1a_2, a_2a_3, \ldots, a_{n-1}a_n\}$ be the set of pairs in A.

Definition 1. *Given two sequences $A{=}a_1a_2\cdots a_n$ and $B{=}b_1b_2\cdots b_m$, if $a_ia_{i+1} = b_jb_{j+1}$ (or $a_ia_{i+1}{=}b_{j+1}b_j$), where $a_ia_{i+1} \in P_A$ and $b_jb_{j+1} \in P_B$, we say that a_ia_{i+1} and b_jb_{j+1} are matched to each other. In a maximum matching of pairs in P_A and P_B, a matched pair is called an* ***adjacency****, and an unmatched pair is called a* ***breakpoint*** *in A and B respectively.*

It follows from the definition that sequences (trajectories) A and B contain the same set of adjacencies but distinct breakpoints. The maximum matched pairs in B (or equally, in A) form the *adjacency set* between A and B, denoted as $a(A, B)$. We illustrate the above definitions in Fig. 2.

We now define the problems to be studied in this paper. We use the decision versions in the definition, though in designing approximation algorithms we will focus on the corresponding optimization versions.

Definition 2. *Median Trajectory Problem.*
Input*: A set of m sequences/trajectories S_i's, $i = 1..m$, over a POI set Σ (with $|\Sigma| = n$), and two positive integers k and ℓ.*
Question*: Can a trajectory T with ℓ elements over Σ be computed such that $\sum_i |a(T, S_i)| \geq k$?*

$$
\begin{aligned}
\textit{sequence} \quad A &= \langle a\ b\ a\ e\ c\ a\ b\ d\ \rangle \\
\textit{sequence} \quad B &= \langle c\ b\ d\ a\ b\ a\ d \rangle \\
P_A &= \{ab, ba, ae, ec, ca, ab, bd\} \\
P_B &= \{cb, bd, da, ab, ba, ad\} \\
\textit{matched pairs} &: (ab \leftrightarrow ab), (ba \leftrightarrow ab), (bd \leftrightarrow bd) \\
a(A,B) &= \{ab, ba, bd\} \\
b_A(A,B) &= \{ae, ec, ca, ab\} \\
b_B(A,B) &= \{cb, da, ad\}
\end{aligned}
$$

Fig. 2. An example for adjacency, breakpoint and the related definitions.

T is typically called a *median trajectory*. When the median trajectory is restricted to be a permutation. We have a variation of the problem.

Definition 3. *Median Permutation Problem.*
Input*: A set of m sequences/trajectories S_i's, $i = 1..m$, over a POI set Σ (with $|\Sigma| = n$), and a positive integers k.*
Question*: Can a permutation M (with n elements) over Σ be computed such that $\sum_i |a(M, S_i)| \geq k$?*

When the median trajectory must cover a superset of the POIs in Σ (but some POIs could appear multiple times), we have the following problem.

Definition 4. *Median Canonical Trajectory Problem.*
Input*: A set of m sequences/trajectories S_i's, $i = 1..m$, over a POI set Σ (with $|\Sigma| = n$), and two positive integers k and $\ell > n$.*
Question*: Can a trajectory Z with ℓ elements over Σ be computed such that $\sum_i |a(Z, S_i)| \geq k$ and $c(Z) \supseteq \Sigma$?*

3 Hardness Results

In this section we prove that all the three problems are NP-hard. As the proofs are in general similar (certainly with some twist), for the first one we give all the details and for the second and third one we only give the most important ideas. The details will be given in the full version of this paper.

Theorem 1. *The decision version of the Median Trajectory problem is NP-complete.*

Proof. We reduce the Hamiltonian Path problem to the Median Trajectory with a bounded length $3n$. WLOG, let $G = (V, E)$ be an undirected graph with vertex degree $deg(u) \geq 2$ for $u \in V$. For each vertex $u \in V$, let $u^1, u^2, \cdots, u^{deg(v)}$ be

the list of vertices adjacent to u ordered by their indices. We define $L_1(u)$ as follows:

$$L_1(u) = \#_u^1 \cdot uu^1\#_u^2 \cdot uu^2 \cdot \#_u^3 \cdot ... \cdot \#_u^{deg(u)} \cdot uu^{deg(u)}.$$

Then, we define

$$L_j(u) = u\$_u u,$$

for $j = 2, 3, 4$. Finally, we define

$$L(u) = \{L_1(u), L_2(u), L_3(u), L_4(u)\}.$$

Let $V = \{v_1, v_2, ..., v_n\}$, and we write $\#_{v_i}^k$ and $\$_{v_i}$ simply as $\#_i^k$ and $\$_i$ respectively. Note that $\#_i^k$ are POIs which occur only once in all trajectories. Then we construct $4n$ trajectories as

$$L = L(v_1) \cup L(v_2) \cup L(v_3) \cup \cdots \cup L(v_n).$$

Notice that $\Sigma = V \cup \{\$_v | v \in V\} \cup \{\#_i^j | i = 1..n, j = 1..deg(v_i)\}$.

Let $P(V) = \langle v_{\pi(1)}, v_{\pi(2)}, \cdots, v_{\pi(n)} \rangle$ be a permutation of V. We claim the following: $P(V)$ is a Hamiltonian path for G if and only if

$$T = \langle v_{\pi(1)}, \$_{\pi(1)}, v_{\pi(1)}, v_{\pi(2)}, \$_{\pi(2)}, v_{\pi(2)}, \cdots, v_{\pi(n)}, \$_{\pi(n)}, v_{\pi(n)} \rangle,$$

which is of $\ell = 3n$ elements, is a median trajectory with a total of $8n - 2$ adjacencies between T and L.

"If part": If $P(V)$ is a Hamiltonian path, then in T each 3-substring $v_{\pi(i)}\$v_{\pi(i)}v_{\pi(i)}, i = 1..n$, contributes 6 adjacencies with $L_2(v_\pi(i))$, $L_3(v_\pi(i))$ and $L_4(v_\pi(i))$ in $L(v_\pi(i))$ (containing 4 trajectories). Moreover, each 2-substring $v_{\pi(i)}v_{\pi(i+1)}, i = 1..n-1$, contributes 2 adjacencies (one with $L_1(v_{\pi(i)})$, the other with $L_1(v_{\pi(i+1)})$. This gives us a total of $6n + 2(n-1) = 8n - 2$ adjacencies between T all the trajectories in L.

"Only-if part" If T is a median trajectory of length $3n$ for L forming a total of $8n - 2$ adjacencies from T to $L(i)$'s, the first thing to notice is that T has length $3n$, hence to form $8n - 2$ adjacencies we cannot use any POI $\#_i^k$ as $\#_i^k v_i$ or $v_j\#_i^k$ in T each could form only one adjacency with trajectories in L. Hence, to maximize the number of adjacencies between T and L, we must only use POIs in $V \cup \{\$_v | v \in V\}$. As $v_{\pi(i)}\$v_{\pi(i)}v_{\pi(i)}$ contributes 6 adjacencies with $L(v_\pi(i))$, including all of them in the median trajectory naturally gives us $6n$ adjacencies. Now, to increase the total adjacencies between T and L to $8n - 2$, we must make use of the $2(n-1)$ adjacencies in the form of $v_{\pi(i)}v_{\pi(i+1)}$ — which implies that $v_{\pi(i)}v_{\pi(i+1)}$ must form an edge of G, hence the 2 adjacencies are formed with $L_1(v_\pi(i))$ and $L_1(v_\pi(i+1))$ respectively. Obviously, the order of such $v_{\pi(i)}v_{\pi(i+1)}$'s gives us a Hamiltonian path for G.

The reduction takes $O(|V| + |E|)$ time. Hence the theorem is proven. □

Theorem 2. *The decision version of the Median Permutation problem is NP-complete.*

Proof. Again, we reduce the Hamiltonian Path problem to the Median Permutation problem, this time with a bounded length $2n$. WLOG, let $G = (V, E)$ be an undirected graph with vertex degree $deg(u) \geq 2$ for $u \in V$. For each vertex $u \in V$, we define $L_1(u) = L_2(u) = uu'$. For each edge $e = (u, w) \in E$, we define $L_3(e) = u'w$ and $L_4(e) = uw'$. L contains $2|V| + 2|E|$ trajectories. Let $\pi(i)$ be a permutation on $[1..n]$, then for $V = \{v_1, v_2, \cdots, v_n\}$, $M = \langle v_{\pi(1)}, v_{\pi(2)}, \cdots, v_{\pi(n)}\rangle$ is a permutation on V. We claim the following: G admits a Hamiltonian path M if and only if T is a median permutation for L in one of the two forms:

1. $T = \langle v_{\pi(1)}, v'_{\pi(1)}, v_{\pi(2)}, v'_{\pi(2)}, \cdots, v_{\pi(n)}, v'_{\pi(n)}\rangle$, or
2. $T = \langle v'_{\pi(1)}, v_{\pi(1)}, v'_{\pi(2)}, v_{\pi(2)}, \cdots, v'_{\pi(n)}, v_{\pi(n)}\rangle$.

We leave the detailed arguments as an exercise for the readers. The reduction obviously takes $O(|V| + |E|)$ time. Hence the theorem is proven. □

Theorem 3. *The decision version of the Median Canonical Trajectory problem is NP-complete.*

Proof. The reduction is slightly different from the previous two. We will reduce the Hamiltonian Cycle problem to the Median Canonical Trajectory problem. Given an undirected graph $G = (V, E)$, WLOG, we want to compute a Hamiltonian Cycle starting from some vertex v_1. For each edge $e = (v_i, v_j)$, we construct a trajectory

$$L(e) = v_i v_j.$$

For v_1, we construct two trajectories

$$L_1(v_1) = L_2(v_1) = \langle \$_s, \#_1, \#_2, \cdots, \#_\gamma, v_1, \#_\gamma, \#_{\gamma-1}, \cdots, \#_1, \$_t\rangle.$$

Let $L = \{L(e)|e \in E\} \cup \{L_1(v_1), L_2(v_1)\}$. Then we have $m = |E| + 2$ trajectories, with $|\Sigma| = n + \gamma + 2$.

Let $\pi(i)$ be a permutation on $[n]$, with $\pi(1) = 1$. We claim the following: G has a Hamiltonian Cycle if and only if T (of $\ell = n + 2\gamma + 3$ elements) is a median canonical trajectory for L in one of the two forms:

1. $T = \langle \$_s, \#_1, \#_2, \cdots, \#_\gamma, v_1, v_{\pi(2)}, \cdots, v_{\pi(n)}, v_1, \#_\gamma, \#_{\gamma-1}, \cdots, \#_1, \$_t\rangle$, or
2. $T = \langle \$_t, \#_1, \#_2, \cdots, \#_\gamma, v_1, v_{\pi(2)}, \cdots, v_{\pi(n)}, v_1, \#_\gamma, \#_{\gamma-1}, \cdots, \#_1, \$_s\rangle$;

moreover, there are $n + 4\gamma + 4$ adjacencies between T and L.

Again, we leave the detailed arguments as an exercise for the readers. The reduction obviously takes $O(|V| + |E|)$ time. Hence we have the theorem. □

4 Approximation Algorithms

In this section we present two simple approximation algorithms for the median trajectory and the median permutation problems respectively. It is open whether a constant factor approximation for the median canonical trajectory problem can be designed.

4.1 A 2-Approximation for the Median Trajectory Problem

Given a set of m trajectories $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ over the same alphabet Σ, we need to compute a median trajectory T^* with ℓ nodes such that $\sum_{i=1..m} |a(T^*, S_i)|$ is maximized.

We use a greedy method to select $\lfloor \ell/2 \rfloor$ edges uv as follows: select uv that appear in $\mathcal{S}$ the maximum number of times, then subtract one copy of the edge uv (or vu) from the corresponding S_i's that uv or vu appears, update S_i's and then repeat until $\lfloor \ell/2 \rfloor$ edges are selected. We then concatenate these edge arbitrarily into a trajectory T. If ℓ is odd, then we arbitrarily concatenate another POI at the end of T.

As a POI could appear in T (and in T^*) multiple times, by the greedy search, the selected $\lfloor \ell/2 \rfloor$ edges form the maximum number of adjacencies with $\mathcal{S}$. This in turns implies that $\sum_{i=1..m} |a(T, S_i)|$ is greater than or equal to the adjacencies formed between any $\lfloor \ell/2 \rfloor$ edges in T^* with the trajectories in $\mathcal{S}$. Hence, $\sum_{i=1..m} |a(T, S_i)| \geq \frac{1}{2} \sum_{i=1..m} |a(T^*, S_i)|$. We thus have the following theorem.

Theorem 4. *The Median Trajectory problem admits a factor-2 polynomial-time approximation.*

4.2 A 1.5-Approximation for the Median Permutation Problem

Given a set of m trajectories $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ over the same alphabet Σ, we need to compute a median permutation M^* with $n = |\Sigma|$ nodes such that $\sum_{i=1..m} a(M^*, S_i)$ is maximized.

Note that $|M^*| = |\Sigma| = n$. In this case, we first construct a weighted graph G such that the vertices are all the (distinct) POIs. Two vertices u and v form an edge $e = (u, v)$ if uv is an adjacency in some L_i; moreover, the weight of e, $W(e)$, is the total number of L_i's that e appears at least once.

The algorithm is to compute the maximum path-cycle cover $PC(G)$ of G, which is a set of disjoint paths/cycles with the maximum total edge weights. It is well-known that $PC(G)$ can be reduced to the maximum weight matching problem, hence can be computed in polynomial time [4, 13].

Let $PC(G) = \{P_1, P_2, \cdots, P_q\} \cup \{C_1.C_2, \cdots, C_r\}$, where P_i's are paths and C_j's are cycles in $PC(G)$. Let $|P_i|$ be the total weight of the edges in P_i, and let $|C_j|$ be the total weight of the edges in C_j'. Let $|PC(G)|$ be the total weight of the edges in P_i's and C_j's.

Then we just delete the edge with the minimum weight in each C_j to have a path C_j^-. We then concatenate all P_i's and C_j^-'s arbitrarily to have an approximate median M.

Note that the optimal solution M^*, with $OPT = \sum_{i=1..m} |a(M^*, S_i)|$, provides a feasible solution for the corresponding path-cycle cover for G. Then, by the optimality of $PC(G)$, which contains a disjoint set of q paths and r cycles, we have

$$OPT \leq |PC(G)| = \sum_{i=1..q} |P_i| + \sum_{j=1..r} |C_j|.$$

As each cycle C_j has at least 3 edges, and the minimum weight edge is deleted to obtain C_j^-, we have

$$|C_j^-| \geq \frac{2}{3}|C_j|,$$

for $j = 1..r$. Hence,

$$|M| = \sum_{i=1..q} |P_i| + \sum_{j=1..r} |C_j^-| \geq \sum_{i=1..q} |P_i| + \frac{2}{3} \sum_{j=1..r} |C_j| \geq \frac{2}{3} OPT.$$

In other words, this gives us a factor-1.5 approximation for the median permutation problem.

Theorem 5. *The Median Permutation problem admits a factor-1.5 polynomial-time approximation.*

We comment that the median permutation problem is mainly of a theoretical meaning, in practice, it is hardly the case that the median trajectory must be a permutation. In the next section, we present a practical solution, based on the greedy 2-approximation algorithm, for the median trajectory problem.

5 Empirical Results

5.1 Heuristic Method for the Median Trajectory Problem

The 2-approximation algorithm presented in the previous section is probably not practical. We present a slightly different heuristic method based on the 2-approximation algorithm. Later, we compare the performance of these two algorithms using simulated data over a 2-month period.

$$\begin{aligned} \textit{sequence} \quad & S_1 = \langle a\ b\ a\ e\ c\ a\ b\ d\ \rangle \\ \textit{sequence} \quad & S_2 = \langle c\ b\ d\ a\ b\ a\ d \rangle \\ \textit{sequence} \quad & S_3 = \langle a\ b\ e\ c\ a\ b\ c\ a\ f \rangle \\ & AM(ab) = \langle 3, 3, 1 \rangle \\ & AM(ac) = \langle 2, 1 \rangle \end{aligned}$$

Fig. 3. An example for the adjacency map, with $m = 3$ and $\Sigma = \{a, b, c, d, e, f\}$. If $\ell = 6$, the 2-approximation would return $T = ab \cdot ab \cdot ac$, which incurs a total of 9 adjacencies with S_i's. The optimal solution would be *abacbd*, which incurs 12 adjacencies.

To make the presentation more clear, we formally define the concept of *adjacency map* as follows. Let ab be a 2-substring which appears in some $S_i, 1 \leq i \leq m$. The *adjacency map* of ab, denoted as $AM(ab)$, is a vector

$$AM(ab) = \langle w_1(ab), w_2(ab), \cdots, w_q(ab) \rangle$$

with $w_1(ab) \geq w_2(ab) \geq \cdots \geq w_q(ab) > 0$ and $w_0(ab) = 0$, such that $w_i(ab)$ is the number of S_i's that contains either ab or ba as a 2-substring after $w_{i-1}(ab)$ number of 2-substrings in the form of ab or ba have been removed from each of these S_i's. In Fig. 3, we show an example for this definition.

With this concept, the original 2-approximation is simply a greedy search in the space (of adjacency maps) $\mathcal{M} = \{w_i(ab)|i > 0, ab \text{ is a 2-substring in some } S_i\}$. More precisely, the algorithm repeatedly selects 2-substrings ab without replacement such that $w_i(ab)$ is the current maximum; moreover, when ab is selected the corresponding $w_i(ab)$ is deleted from the search space. Our new heuristic method is based on this greedy search, and searches more carefully by possibly extending existing solutions.

We first present the one-step implementation of this greedy search method, named as *Greedy1*, which is a subroutine used in the final algorithm.

Algorithm 1 Greedy1

Input: adjacency map $\mathcal{M}$, temporary solution T
Output: adjacency map $\mathcal{M}$, temporary solution T, integer $k > 0$
1: find a 2-substring ab with the maximum $k = w_i(ab)$ in $\mathcal{M}$.
2: put ab in T.
3: remove $w_i(ab)$ from $\mathcal{M}$.

To improve the simple greedy search method, which adds a 2-substring to the solution at each round, we make the following observations. In general, the final solution T is a concatenation of *maximal* substrings $T_1, T_2, \cdots, T_q$ such that breakpoints only exist between T_i and T_{i+1}. More precisely, let $T_i = \langle T_i[1], T_i[2], \cdots, T_i[|T_i|]\rangle$, then $w_x(T_i[|T_i|]T_{i+1}[1]) = 0$ for any x — in other words, $T_i[|T_i|]T_{i+1}[1]$ does not even exist in $\mathcal{M}$. (For convenience, we will also use $left(T_i)$ and $right(T_i)$ to represent $T_i[1]$ and $T_i[|T_i|]$ respectively.) With this observation in mind, if ab is first selected with Greedy1 and uv would be selected next with Greedy1, then we could select cd instead if $w_1(bc) + w_1(cd) > w_1(uv)$. Naturally, this means if we could extend ab into $abcd$, then it is better than $ab \cdot uv$, where $b \cdot u$ could be a breakpoint.

Algorithm 2 Greedy2

Input: adjacency map $\mathcal{M}$, temporary solution T
Output: adjacency map $\mathcal{M}$, temporary solution T, integer $k > 0$

for a maximal substring S in the temporary solution T **do**
  find a 2-substring ab s.t. $L = w_i(ab) + w_j(\langle b, left(S)\rangle)$ is maximum.
  find a 2-substring cd s.t. $R = w_{i'}(\langle right(S), c\rangle) + w_{j'}(cd)$ is maximum.
  update $temp \leftarrow \max(L, R)$.
  update $k \leftarrow \max(k, temp)$.
end for
update $S \leftarrow ab \circ S$ if $temp = L$, and update $S \leftarrow S \circ cd$ if $temp = R$.
remove $w_i(ab)$ and $w_j(\langle b, left(S)\rangle)$ from $\mathcal{M}$ if $temp = L$.
remove $w_{i'}(\langle right(S), c\rangle)$ and $w_{j'}(cd)$ from $\mathcal{M}$ if $temp = R$.

We are now ready to give the heuristic algorithm based on Greedy1 and Greedy2.

Algorithm 3 The Heuristic Algorithm

Input: sequences $S_i (i = 1..m)$, integer ℓ
Output: sequence T with ℓ nodes, number r of adjacencies between T and S_i's.

$T \leftarrow \varepsilon$, $r \leftarrow 0$.
compute the adjacency map $\mathcal{M}$ from S_i's.
$(\mathcal{M}, T, r) \leftarrow$ Greedy1$(\mathcal{M}, T)$
$\ell \leftarrow \ell - 2$
while $\ell \geq 2$ **do**
  if Greedy1$(\mathcal{M}, T)$.r $>$ Greedy2$(\mathcal{M}, T)$.r **then**
    $(\mathcal{M}, T, r) \leftarrow$ Greedy1$(\mathcal{M}, T)$
  else
    $(\mathcal{M}, T, r) \leftarrow$ Greedy2$(\mathcal{M}, T)$
  end if
  $\ell \leftarrow \ell - 2$
end while
if $\ell > 0$ **then**
  extend a maximal substring S of T by a new node x such that $temp1 = \max(w_i(\langle x, left(S)\rangle), w_j(\langle right(S), x\rangle))$ is maximized over all S and x.
  update $r \leftarrow r + temp1$.
  update $S \leftarrow x \circ S$ if $temp1 = w_i(\langle x, left(S)\rangle)$.
  update $S \leftarrow S \circ x$ if $temp1 = w_j(\langle right(S), x\rangle)$.
end if

Note that as the heuristic algorithm subsumes the 2-approximation algorithm, it provides a performance guarantee of at most 2 as well. Unfortunately, this is the best we could say regarding its theoretical performance. We show next

that with randomly generated simulated data, the actual performance (approximation factor) is always bounded by 1.14.

5.2 Empirical Results

For the empirical results, we first generate 60 sequences randomly (presumably for 60 days or 2 months), each with a length in the range $[1.5|\Sigma| - 5, 1.5|\Sigma| + 5]$. Then for different target length ℓ, we run the 2-approximation algorithm and the heuristic algorithm to obtain the median trajectories $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively. We run this 10 times to obtain the average of the maximum *appearance* (the maximum number of time a node, or POI, appearing in any sequence S_i), and the averages of $\mathcal{T}_1$ and $\mathcal{T}_2$ — the last two being rounded to the largest integers below. The result is summarized in Table 1.

We also obtain the following Table 2, using some variations to the simulated data. With the 2-approximation algorithm, we have $App_1 \geq Opt/2$, or equivalently, $Opt \leq 2 \cdot App_1$. (For practical reason, here we can take App_1 roughly the same as $|\mathcal{T}_1|$.) Hence, the actual performance (aka. approximation factor) can be bounded from above by

$$\frac{|\mathcal{T}_2|}{2 \cdot |\mathcal{T}_1|} \leq \frac{|\mathcal{T}_2|}{Opt},$$

which is always less than 1.14 for the data in both tables.

Table 1. Results for $|S_i| \in [1.5|\Sigma| - 5, 1.5|\Sigma| + 5]$, averaged over 10 tries.

$\lvert\Sigma\rvert$	ℓ	Avg. of max appearance	$\lvert\mathcal{T}_1\rvert$ (2-App)	$\lvert\mathcal{T}_2\rvert$ (Heuristic)
30	40	6.7	215	384
30	45	6.8	243	427
30	50	6.7	261	467
40	55	6.9	251	445
40	60	7.1	268	482
40	65	7.0	294	521
50	70	7.3	280	501
50	75	6.8	303	540
50	80	7.1	316	570
60	85	7.5	318	558
60	90	7.5	330	587
60	95	7.1	352	620

6 Concluding Remarks

Using the concept of adjacency from computational genomics, we try to compare the similarity of trajectories from a vehicular network, which we propose to use as the first step to build trustfulness in vehicular networks. Here, a trajectory is

Table 2. Results for $|S_i| \in [1.5|\Sigma| - 10, 1.5|\Sigma| + 10]$, averaged over 10 tries.

$\|\Sigma\|$	ℓ	Avg. of max appearance	$\|\mathcal{T}_1\|$ (2-App)	$\|\mathcal{T}_2\|$ (Heuristic)
30	35	6.3	194	337
30	40	6.7	214	383
30	45	6.2	244	425
30	50	6.5	258	458
30	55	6.5	288	511
40	50	7.2	231	411
40	55	7.5	255	450
40	60	6.9	270	481
40	65	7.0	297	524
40	70	6.7	308	550
50	65	7.2	267	470
50	70	7.2	278	503
50	75	7.1	308	543
50	80	7.5	318	571
50	85	7.4	344	611
60	80	7.4	298	530
60	85	7.0	316	556
60	90	7.5	334	591
60	95	7.0	343	611
60	100	7.2	365	652

a sequence of POIs visited by a vehicle in one day. This paper focuses on computing a consensus trajectory given a set of such trajectories, the objective being maximizing the total number of adjacencies between the consensus trajectory and the input trajectories. We consider three versions of the problem: Median Trajectory, Median Permutation and Median Canonical Trajectory, which are all NP-hard. We also give factor-2 and factor-1.5 approximation algorithms for the first two problems. For the Median Trajectory problem, we also design a heuristic algorithm which greatly outperforms the 2-approximation algorithm using simulated data articulately generated. The actual approximation factor is bounded from above at 1.14, even though in theory it is 2 in the worst case.

There are still many open questions along this line. Theoretically, does Median Canonical Trajectory admit a constant factor approximation? Can the approximation factor for Median Trajectory be improved to be below 2? These questions definitely need further research.

Acknowledgments

This research was supported by NSF under project CNS-1761641. Peng Zou was also supported by a COE Benjamin PhD Fellowship at Montana State University.

References

1. S. Angibaud, G. Fertin, I. Rusu, A. Thevenin, and S. Vialette. On the approximability of comparing genomes with duplicates. *J. Graph Algorithms and Applications*, 13(1):19-53, 2009.
2. D. Bryant. The complexity of the breakpoint median problem. Technical Report CRM-2579. Centre de Recherches en Mathématiques, Université de Montréal. 1998.
3. J. Doucer. The Sybil attack. *Proc. IPTPS'01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251-260, 2002.
4. J. Edmonds and E. Johnson. Matching: a well-solved class of integer linear programs. In *Combinatorial Structures and Their Applications (Gordon and Breach, New York)*, pages 89–92, 1969.
5. H. Hartenstein and L. Kenneth. *VANET: Vehicular Applications and Inter-Networking Technologies*. Wiley, New Jersey, USA, 2009.
6. H. Jiang, F. Zhong, and B. Zhu. Filling scaffolds with gene repetitions: maximizing the number of adjacencies. *Proc. 22nd Annual Combinatorial Pattern Matching Symposium (CPM'11)*, LNCS 6661, pp. 55-64, 2011.
7. H. Jiang, C. Zheng, D. Sankoff, and B. Zhu. Scaffold filling under the breakpoint and related distances. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1220-1229, July/August, 2012.
8. G. Liu, Q. Yang, H. Wang, X. Lin and M. Wittie. Assessment of multi-hop interpersonal trust in social networks by Three-Valued Subjective Logic. *Proc. INFOCOM'14*, pages 1698-1706, 2014.
9. G. Liu, Q. Chen, Q. Yang, B. Zhu, H. Wang and W. Wang. OpinionWalk: An efficient solution to massive trust assessment in online social networks. *Proc. INFOCOM'17*, pages 1-9, 2017.
10. M. Li, B. Ma and L. Wang. Finding similar regions in many sequences. *J. Comput. Sys. Sci.*, 65(1):73-96, 2002.
11. P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung and J-P. Hubaux. Secure vehicular communication systems: design and architecture. *IEEE Communications Magazine*, 46(11):100-109, 2008.
12. I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Elec. Colloq. Comput. Complexity*, TR-98-071. 1998.
13. Y. Shiloach. Another look at the degree constrained subgraph problem. *Info. Process. Lett.*, 12(2):89-92, 1981.
14. R. Shrestha, S. Djuraev and S.Y. Nam. Sybil attack detection in vehicular network based on received signal strength. *Proc. 3rd Intl. Conf. on Connected Vehicles and Expo (ICCVE'14)*, pages 745-746, 2014.
15. E. Tannier, C. Zheng and D. Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics* 10:120. 2009.
16. J. Zhang. A survey on trust management for VANETS. *Proc. 2011 IEEE Intl. Conf. Advanced Information Networking and Applications (AINA'11)*, pages 105-115, 2011.