Noname manuscript No.
(will be inserted by the editor)

Computing a Consensus Trajectory in a Vehicular Network

Peng Zou · Letu Qingge · Qing Yang · Binhai Zhu *

Received: date / Accepted: date

Abstract In this paper, we study the problem of computing a consensus trajectory of a vehicle given the history of Points of Interest (POIs) visited by the vehicle over a certain period of time. The problem arises when a system tries to establish the social connection between two vehicles in a vehicular network, where three versions of the problem are studied. Formally, given a set of m trajectories, the first version of 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 based on a greedy method. We implement the greedy algorithm and a variation of it which is based on a more natural greedy search on a new data structure called adjacency map. In the second version of the problem where the sequence T is restricted to be a permutation, we show that the problem remains NP-hard but the approximation factor can be improved to 1.5. In the third version where the sequence T has to contain all letters of Σ, we again prove that it is NP-hard. We implement a simple greedy algorithm and a variation of the 1.5-approximation algorithm for the second version, and which

* Corresponding author.

This research was partially supported by NSF under project CNS-1761641.

Peng Zou and Binhai Zhu
Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA
E-mail: peng.zou@student.montana.edu, bhz@montana.edu

Letu Qingge
Department of Computer Science, North Carolina A & T State University, Greensboro, NC 27411, USA
E-mail: lqingge@ncat.edu

Qing Yang
Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207, USA
E-mail: qing.yang@unt.edu

are used to construct solution for the third version. Our algorithms are tested on the simulation data and the empirical results are very promising.

Keywords Consensus Trajectory · NP-hardness · Approximation Algorithm · Heuristic Algorithm

1 Introduction

Comparing the similarity among different trajectories has a broad applications in different vehicular networks. For example, we can build a trustful system based on the similar behavior of different vehicles in a vehicular ad hoc network (VANET). In addition, we also can design a transportation network with a risk equity based on the similarity of different paths in a hazardous materials transportation.

In a vehicular ad hoc network (VANET), building a trustful communication environment is a critical privacy issue due to a property of high flexible topology of VANET. Vehicles can communicate to exchange useful information. However, to determine the trustfulness among vehicles is a challenging problem. A few trust models have been developed to address this security problem due to the special safey requirement of VANET. Its main functions include traffic control, accident avoidance and parking management (Hartenstein and Laberteaux, 2010; Papadimitratos et al., 2008).

A variety of attacks has been observed in VANET. For instance, fake message could be disseminated by the sender to compete the parking space during rush hour. The Sybil attacks could generate fake identities to falter the functioning of the whole system (Douceur, 2002; Shrestha et al., 2014). Thus, it is important to build a trust management system in VANET (Zhang, 2011; Liu et al., 2014; Patel and Jhaveri, 2015; Liu et al., 2017; Kchaou et al., 2018).

To build the initial trustfulness, we propose 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. Of course, to protect privacy, all sensitive information regarding the location of POIs, time a POI is visited, etc, could be erased for our purpose. Hence, the examples of POIs could be "home", "office", "restaurant", "coffee shop", "gym", etc. (To further protect privacy, we could use numbers or letters to represent POIs. An example is shown in Fig.1.)

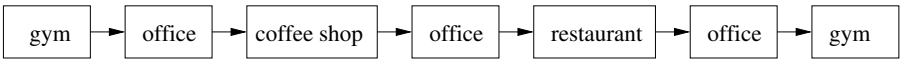

Fig. 1 The POIs visited by a white-collared professional during a typical working day. We use $1, 2, 3$, and 4 to represent 'gym', 'office', 'coffee shop', and 'restaurant' respectively, and the trajectory could be represented as a sequence $\langle 1, 2, 3, 2, 4, 2, 1\rangle$.

On the other hand, comparing the similarity between different paths is also useful for designing a hazardous materials transportation network with

risk equity. The hazardous materials (hazmat) transportation network design problem involves two parties, the government and the carrier. The government can restrict some edges of transportation network to minimize the overall risk over the hazardous materials transportation network. Then, the hazmat carriers choose the paths with the least transportation cost from the designated network by government. However, it is more likely that multiple hazmat carriers choose the same segment of paths in order to save their transportation cost. This leads to some edges in the transportation network having higher risks which makes it is more dangerous for surrounding areas once the accident happens (Xin et al., 2015; Kara and Verter, 2004).

Now suppose we have a list of POIs visited by each vehicle over certain period of time. The problem is how we compare these sequences of POIs to identify the similarity between these trajectories? Inspired by computing the consensus DNA sequence in the computational biology, we adopt an adjacency metric to determine the consensus path for each vehicle which can further be used to compare the similarities among different trajectories (Angibaud et al., 2009; Jiang et al., 2011, 2012). In a nutshell, once two consensus trajectories, T_1 and T_2, for each vehicle are computed, we can 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; and, to make the computation more accurate, we could even delete some redundant POIs — the latter research has been covered in (Qingge et al., 2019, 2021).) This paper focuses on the problem of computing the consensus trajectories.

In this formulation, the trustfulness level between vehicles is evaluated by the similarity of different trajectories. The more similar between trajectories, the more trustfulness given between vehicles. In the hazardous materials transportation network design problem, the government can avoid similarity paths when considering the risk equity. Because the more similar between paths, the more risk imposed around that paths.

Given a set of sequences, computing a consensus sequence is an important problem in computational biology, using different distance measures. For example, given a set of DNA sequences $\{S_1, S_2, ..., S_m\}$, computing a median sequence T of length at most L such that $\sum_i d_H(T, S_i)$ is minimized, where $d_H()$ is the famous Hamming distance, is an important problem in molecular biology for computing conserved regions (Li et al., 2002). When the input sequences are genomes (i.e., with a larger alphabet compared with DNA sqeuences) and the distance measure is the *breakpoint* distance, then a natural problem is the breakpoint median problem (Bryant, 1998; Pe'er and Shamir, 1998; Tannier et al., 2009). It is known that all these problems are NP-hard; in fact, for the breakpoint median problem it is NP-hard even if there are only three input genomes (Bryant, 1998). Nonetheless, this problem is different from ours. For example, if we have three permutations $A_1 = abcd, A_2 = bdca, A_3 = cbda$ then any permutation median like $M = abcd$ could only contribute at most 5 adjacencies between M and the input permutations A_i's. On the other hand, if the median does not have to be a permutation, as in our case, then $M' = bcdb$ would contribute 6 adjacencies instead.

In summary, while the problems we investigate in this paper are similar to some previous works, they are different in several different ways. Firstly, the number of adjacencies between two sequences is not a distance measure; instead, it is a similarity measure featuring "the more number of adjacencies between two sequences, the more similar they are". Secondly, let the alphabet be the set of POIs, then it is not necessarily a small constant; moreover, POIs are certainly allowed to repeat in any input sequence. For instance, we can use adjacencies *ab* and *bc* to represent two distinct edges in a vehicular network. Thirdly, the length of the median trajectory is bounded (otherwise, the problem would be trivial) and the median trajectory is not a permutation in general. All these make the design of efficient (approximation) algorithms more challenging; in fact, for one of the three versions we do not know how to design a constant-factor approximation at all.

We comment that the conference version of this paper appeared in (Zou et al., 2019), though a lot of details are omitted due to space constraint. In this paper, we fill all the omitted details. In addition, we add new heuristic algorithms for the third version of the problem, together with new empirical results.

The paper is organized as follows. In Section 2, we give necessary definitions and define three versions of the problem. In Section 3, we give NP-hardness proofs for all the three versions of the problem. In Section 4, we present two approximation algorithms for the first and second version of the problem, with approximation factor 2 and 1.5 respectively. In Section 5, we present two practical algorithms based on these approximation algorithms and some additional heuristic methods for the first and third version of the problem and show extensive empirical results. In Section 6, we conclude the paper.

2 Preliminaries

We first give some necessary definitions. Given a set Σ of POIs (which will also be called elements, letters, or nodes), a string P is called a *permutation* if each letter in Σ appears exactly once in P. We use *c(P)* to denote the set of elements in the permutation P. A string A is called a *sequence* (or a trajectory) if some POIs appear more than once in A, and $c(A)$ denotes the multi-set of POIs in A. (In this paper, any trajectory will be preprocessed so that consecutive identical letters will always be reduced as a single letter. For example, $A = bcdddaba$ would be preprocessed as $A = bcdaba$.) For instance, $\Sigma = \{a, b, c, d\}$, $A = abcdacdb$, $c(A) = \{a, a, b, b, c, c, d, d\}$. A substring with k letters (in a sequence A) is called a *k-substring*, and a 2-substring is also called a *pair*. Throughout this paper, 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 $A{=}a_1a_2a_3\cdots a_n$, we define $P_A = \{a_1a_2, a_2a_3, \ldots, a_{n-1}a_n\}$ as 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 $a_i a_{i+1}$ and $b_j b_{j+1}$ are matched to each other. (This naturally defines a bipartite graph $G(A, B)$, where the vertices are pairs in A and B.) In a maximum matching of $G(A, B)$, or 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.

Following the above definitions, clearly sequences (trajectories) A and B contain the same set of adjacencies but distinct breakpoints. The matched pairs in A and B, corresponding to the maximum matching, form the *adjacency set* between A and B, denoted as $a(A, B)$. The unmatched pairs in A (resp. B) form the breakpoints in A (resp. B), denoted as $b_A(A, B)$ (resp. $b_B(A, B)$). In Fig. 2, we show a detailed example with respect to these definitions.

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

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

We proceed to define the problems to be investigated in this paper. We use the decision versions in the definition, certainly in designing (approximation) algorithms we need to 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 ($|\Sigma| = n$), and two positive integers k and ℓ.
Question: Is there a trajectory T with ℓ elements over Σ such that $\sum_i |a(T, S_i)| \geq k$?

We call T as a *median trajectory* throughout the paper. When the median trajectory is restricted to be a permutation, we have a different version of the problem. We comment that this version does not seem to be directly related to vehicular networks, and we study it mainly for a theoretical purpose. But a solution of it could be used to solve the (more practical) median canonical trajectory problem (to be defined).

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

When the median trajectory must cover a superset of all the POIs in Σ, and some POIs must appear multiple times (resulting in a trajectory with length strictly longer than $|\Sigma|$), we have the third version of the problem as follows.

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

In the next section, we prove that all the three versions are NP-hard.

3 Hardness Results

First of all, as all the three decision problems are obviously in NP, in this section we will focus on the corresponding NP-hardness reductions.

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

Proof We reduce the Hamiltonian Path problem to the Median Trajectory problem with a bounded length $3n$. Without loss of generality, let $G = (V, E)$ be an undirected graph with vertex degree $deg(u) \geq 2$ for each $u \in V$. For each vertex $u \in V$, let $u^1, u^2, \cdots, u^{deg(u)}$ be the list of vertices adjacent to u ordered by their indices. For each u, we define a set $L(u)$ of 4 trajectories $\{L_1(u), L_2(u), L_3(u), L_4(u)\}$. We define $L_1(u)$ as follows:

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

Subsequently, we define three identical trajectories

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

for $j = 2, 3, 4$. Thirdly, 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 simply write $\#_{v_i}^k$ and $\$_{v_i}$ as $\#_i^k$ and $\$_i$, for $i = 1..n$. It should be noted that $\#_i^k$ are POIs which occur only once in all trajectories. Finally, we construct a set L of $4n$ trajectories as

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

Note 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 that $P(V)$ 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 $L(v_{\pi(i)})$ (which are three identical trajectories among the four in $L(v_{\pi(i)})$). In addition, 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 results in 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 to note is that T has length $3n$, therefore to form $8n-2$ adjacencies we cannot use any POI $\#_i^k$ since $\#_i^k v_i$ or $v_j\#_i^k$ in T could each form only one adjacency with the trajectories in L. Hence, to maximize the number of adjacencies between T and L, we must only use POIs in $V \cup \{\$_v | v \in V\}$. Since $v_{\pi(i)}\$v_{\pi(i)}v_{\pi(i)}$ 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 number of adjacencies between T and L to $8n-2$, we 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 be an edge in G, therefore the 2 adjacencies are formed with $L_1(v_{\pi(i)})$ and $L_1(v_{\pi(i+1)})$ respectively. Clearly, the order of such $v_{\pi(i)}v_{\pi(i+1)}$'s gives us a Hamiltonian path for G.

The reduction clearly takes $O(|V| + |E|)$ time. Therefore, 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. Without loss of generality, let $G = (V, E)$ be an undirected graph with vertex degree $deg(u) \geq 2$ for each $u \in V$. 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 a set of $2|V| + 2|E|$ 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 that G admits a Hamiltonian path M if and only if T is a median permutation for L in one of the two formats:

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

which is of length $\ell = 2n$ elements, is a median permutation trajectory with a total of $3n - 1$ adjacencies between T and L. Be reminded that n is the number of vertices in G.

"If part": If M is a Hamiltonian path, then in T each 2-substring $v_{\pi(i)}v'_{\pi(i)}, i = 1..n$, contributes 2 adjacencies with $L_1(v_{\pi(i)})$ and $L_2(v_{\pi(i)})$ in L (containing 2 trajectories). Moreover, each 2-substring $v_{\pi(i)}v'_{\pi(i+1)}$ or $v'_{\pi(i)}v_{\pi(i+1)}$, $i = 1..n-1$, contributes 1 adjacency (with either $L_3(v_{\pi(i)}v_{\pi(i+1)})$ or with $L_4(v_{\pi(i)}v_{\pi(i+1)})$). This gives us a total of $2n + n - 1 = 3n - 1$ adjacencies between T and all the trajectories in L.

"Only-if part" If T is a median permutation trajectory of length $2n$ for L forming a total of $3n-1$ adjacencies from T to $L(i)$'s, the first thing to notice is that T has length $2n$, in which each adjacent POIs $v_{\pi(i)}v'_{\pi(i)}, i = 1..n$ in T contributes 2 adjacencies with L. Thus, we have a total number of $2n$ adjacencies in T. In order to obtain the total adjacencies between T and L to $3n-1$, we need to form another $n-1$ adjacencies in T. According to our construction of trajectory L, only edges in graph G can contribute $n-1$ adjacencies. In other words, $v_{\pi(i)}v_{\pi(i+1)}$ must form an edge of G so that 1 adjacency is formed with $L_3(v_{\pi(i)}v_{\pi(i+1)})$ or with $L_4(v_{\pi(i)}v_{\pi(i+1)})$. Hence, a total number of $n-1$ adjacencies are formed and the corresponding $v_{\pi(i)}v_{\pi(i+1)}$'s give us a Hamiltonian path for G.

The reduction 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 This reduction is slightly different from the previous ones. Instead of Hamiltonian Path, we will reduce the Hamiltonian Cycle problem to the Median Canonical Trajectory problem. Given an undirected graph $G = (V, E)$, without loss of generality, we need to compute a Hamiltonian Cycle starting from some vertex, say v_1. For each edge $e = (v_i, v_j)$, we construct a trajectory

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

For the vertex 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 the set of trajectories be $L = \{L(e)|e \in E\} \cup \{L_1(v_1), L_2(v_1)\}$. Clearly, in L we have $m = |E| + 2$ trajectories, with $|\Sigma| = n + \gamma + 2$ (γ is some positive integer).

Let $\pi(i)$ be a permutation on $[n]$, with $\pi(1) = 1$. We claim that 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 formats:

$$\begin{aligned} T_1 &= \langle \$_s, \#_1, \#_2, \cdots, \#_\gamma, v_1, v_{\pi(2)}, \cdots, v_{\pi(n)}, \\ &\quad v_1, \#_\gamma, \#_{\gamma-1}, \cdots, \#_1, \$_t \rangle, \textit{or} \\ T_2 &= \langle \$_t, \#_1, \#_2, \cdots, \#_\gamma, v_1, v_{\pi(2)}, \cdots, v_{\pi(n)}, \\ &\quad v_1, \#_\gamma, \#_{\gamma-1}, \cdots, \#_1, \$_s \rangle; \end{aligned}$$

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

If G has a Hamiltonian Cycle, without loss of generality, let it be defined by some permutation π, with $\pi(1) = 1$, then the Hamiltonian cycle can be written as $\langle v_1, v_{\pi(2)}, \cdots, v_{\pi(n)}, v_1 \rangle$. Clearly, the n edges $v_{\pi(i)}v_{\pi(i+1)}$ (and $v_{\pi(n)}v_1$) in the Hamiltonian cycle each contributes one adjacency, with either T_1 or T_2. Between $L_1(v_1)$ (resp. $L_2(v_1)$) and either T_1 or T_2 there are $2\gamma+2$ adjacencies.

Therefore, the total number of adjacencies between L and either T_1 or T_2 is $n + 2(2\gamma + 2) = n + 4\gamma + 4$.

For the reverse direction, suppose that between the median T and L there are $n + 4\gamma + 4$ adjacencies. By the construction, the trajectories $L(e)$'s can contribute at most n adjacencies. As the median T must be canonical, it must contain the other $\gamma+2$ (non-vertex) letters in Σ. As the length of T is $n+2\gamma+3$, $L_1(v_1) = L_2(v_1)$ and their length is $2\gamma + 3$, all the pairs in $L_1(v_1)$ and $L_2(v_1)$ must form adjacencies with T. Therefore, there must be n trajectories in $L(e)$ which correspond to a permutation of n vertices, which forms a substring of length $n + 1$ in T. For enforcing this permutation to have the property of $\pi(1) = 1$ (equivalent to reordering the vertices in G), we have a Hamiltonian cycle in G.

The reduction obviously takes $O(|V| + |E|)$ time. Hence we have the theorem.

4 Approximation Algorithms

In this section we first present two simple approximation algorithms for the median trajectory and the median permutation problems respectively. (They will both be implemented later.) It is still open whether a constant factor approximation for the median canonical trajectory problem can be obtained.

4.1 A 2-Approximation for the Median Trajectory Problem

Recall that the median trajectory problem is defined as: 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 depict our approximation algorithm as follows.

Our algorithm selects $\lfloor \ell/2 \rfloor$ edges uv in a greedy way such that uv appears in $\mathcal{S}$ the maximum number of times, then we 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 finally repeat until $\lfloor \ell/2 \rfloor$ edges are selected. These edges are then concatenated arbitrarily into a trajectory T. If ℓ is odd, then we arbitrarily concatenate another POI at the end of T.

Since a POI could appear in T (and also 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 $\mathcal{S}$. Therefore, $\sum_{i=1..m} |a(T, S_i)| \geq \frac{1}{2} \sum_{i=1..m} |a(T^*, S_i)|$. We then have the following theorem.

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

We comment that even though the approximation algorithm is simple, the 2 factor is in fact tight for the algorithm. An example can be constructed as follows. Let $m = \ell$ and let ℓ be even. We first construct a set of ℓ (disjoint) ℓ-permutations $Q = \{Q_1, Q_2, \cdots, Q_\ell\}$, i.e., Q_i and Q_j share no letter at all and $|Q_i| = \ell$. Then for $i = 1..\ell$, we construct S_i as $a_j b_j$'s, $j = 1..\ell$, separated by $\ell - 1$ permutation segments in $Q - \{Q_i\}$. For instance,

$$S_1 = a_1 b_1 \cdot Q_2 \cdot a_2 b_2 \cdot Q_3 \cdot a_2 b_3 \cdots a_{\ell-1} b_{\ell-1} \cdot Q_\ell \cdot a_\ell b_\ell,$$

$$S_2 = a_1 b_1 \cdot Q_1 \cdot a_2 b_2 \cdot Q_3 \cdot a_3 b_3 \cdots a_{\ell-1} b_{\ell-1} \cdot Q_\ell \cdot a_\ell b_\ell,$$

etc. Since the approximation algorithm is greedy, it will select and concatenate $\ell/2$ number of $a_j b_j$'s as the median, which results in total of $\ell(\ell/2) = \ell^2/2$ adjacencies. The optimal algorithm is to select any Q_i, which appears $\ell - 1$ times in S_i's and results in a total of $(\ell - 1) \cdot (\ell - 1) = (\ell - 1)^2$ adjacencies. Clearly,

$$\lim_{\ell \to +\infty} \frac{(\ell - 1)^2}{\ell^2/2} = 2.$$

An interesting theoretical question is whether a better approximation can be designed for Median Trajectory.

4.2 A 1.5-Approximation for the Median Permutation Problem

Recall that the median permutation problem is defined as: given a set of m trajectories $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ over the same alphabet Σ, we are supposed to compute a median permutation M^* with $n = |\Sigma|$ nodes such that $\sum_{i=1..m} a(M^*, S_i)$ is maximized. Due to the restriction on this problem, we are able to give a factor-1.5 approximation. The details are given as follows.

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 (Edmonds and Johnson, 2001; Shiloach, 1981).

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,

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

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 theoretical meaning only: in practice, it is hardly the case that the median trajectory must be a permutation. However, we could use it as a subroutine for a heuristic solution for the median canonical trajectory problem. (Nonetheless, again, whether it is possible to design a better approximation for Median Permutation is an interesting theoretical problem.)

In the next section, we present some practical solutions, based on these two approximation algorithms, for the median trajectory and median canonical trajectory problems.

5 Empirical Results

5.1 Heuristic Method for the Median Trajectory Problem

The 2-approximation algorithm presented in the previous section might not be good enough practically. Hence we present a slightly different heuristic method based on the 2-approximation algorithm. Later, we compare the performance of these two algorithms using some simulated data presumed over a 2-month period.

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

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 compute and return $T = ab \cdot ab \cdot ac$, which incurs a total of 9 adjacencies with S_i's. The optimal solution would be *abacda*, 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 recursively defined as 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, one each from the corresponding S_i's containing ab or ba. In Fig. 3, we show an example to explain this definition.

Following this new concept, the original 2-approximation can be thought of a simple greedy search in the space (of adjacency maps) $\mathcal{M} = \{w_i(ab) | i > 0, ab \text{ is a 2-substring in some } S_i\}$. To be more precise, the algorithm repeatedly selects 2-substrings ab without replacement such that $w_i(ab)$ is the current maximum. Note that when ab is selected the corresponding $w_i(ab)$ is deleted from the search space. Our new heuristic algorithm is to extend this greedy search by searching more carefully to possibly extend existing solutions.

We first present the one-step implementation of the 2-approximation greedy search method, named as *Greedy1*. This implementation is mainly used for the comparison purpose. Throughout the heuristic algorithms, k is a global variable representing the total number of adjacencies between the selected 2-substrings in the current solution T and all the input sequences.

Algorithm 1 Greedy1

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

1: find a 2-substring ab with the maximum value $w_i(ab) = t$ in $\mathcal{M}$.
2: put ab in T and update $k \leftarrow k + t$.
3: remove $w_i(ab)$ from $\mathcal{M}$.

Note that in this implementation k does not contain the adjacencies incurred by additonal 2-substrings between the selected ones, e.g., we would count k as 3+3+2=8 for the solution $ab \cdot ab \cdot ac$ in Fig. 3 (without considering the contribution of ba). To improve the greedy search method, which adds a 2-substring to the solution at each step, we make the following observations. In principle, 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 specifically, 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 $\mathcal{M}$. (To simplify the presentation, we will also use $left(T_i)$ and $right(T_i)$ to represent $T_i[1]$ and $T_i[|T_i|]$ respectively.) With the above observation, if ab is first selected with Greedy1 and uv would be selected next with Greedy1, then we could select cd instead if $w_1(bc) + w_1(cd) > w_1(uv)$. This implies that if we could naturally 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, global integer variable k

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

Likewise, we can also present another one-step implementation which adds a new node x to the left or right of the current solution T. In this case, of course, x is selected of that the w-value for either $x \cdot left(T)$ or $right(T) \cdot x$ is maximized. We name this greedy method as Add1.

Algorithm 3 Add1

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

1: 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.
2: update $k \leftarrow k + temp1$.
3: update $S \leftarrow x \cdot S$ if $temp1 = w_i(\langle x, left(S)\rangle)$.
4: update $S \leftarrow S \cdot x$ if $temp1 = w_j(\langle right(S), x\rangle)$.
5: remove $temp1$ from $\mathcal{M}$.

We are now ready to give the heuristic algorithm for the median trajectory problem, based on Greedy1, Greedy2 and Add1. (Add1 will also be used a bit later for the other version.) 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 between 1.48 and 1.6.

Algorithm 4 Heuristic Algorithm for Median Trajectory

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

$T \leftarrow \varepsilon$, $k \leftarrow 0$.
compute the adjacency map $\mathcal{M}$ from S_i's.
$(\mathcal{M}, T, k) \leftarrow$ Greedy1$(\mathcal{M}, T)$
$\ell \leftarrow \ell - 2$
while $\ell \geq 2$ **do**
  if Greedy1$(\mathcal{M}, T)$.k $>$ Greedy2$(\mathcal{M}, T)$.k **then**
    $(\mathcal{M}, T, k) \leftarrow$ Greedy1$(\mathcal{M}, T)$
  else
    $(\mathcal{M}, T, k) \leftarrow$ Greedy2$(\mathcal{M}, T)$
  end if
  $\ell \leftarrow \ell - 2$
end while
if $\ell > 0$ **then**
  $(\mathcal{M}, T, k) \leftarrow$ Add1$(\mathcal{M}, T)$
end if

5.2 Empirical Results for the Median Trajectory Problem

For the empirical results, we first generate 60 sequences randomly (presumably for 60 days or 2 months), each with a length in the range $[2|\Sigma| - 5, 2|\Sigma| + 5]$. With different target lengths ℓ, we then 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 *frequency* (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 present additional empirical results in Table 2, using some variations to the simulated data (e.g., the range of ℓ is larger). 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 $\mathcal{R}$, with

$$\mathcal{R} = \frac{2 \cdot |\mathcal{T}_1|}{|\mathcal{T}_2|} \geq \frac{Opt}{|\mathcal{T}_2|},$$

which is always between 1.48 and 1.6 for the data in both tables.

Table 1 Comparison of the 2-approximation and the heuristic algorithms for the Median Trajectory problem. Here we have $|S_i| \in [2|\Sigma| - 5, 2|\Sigma| + 5]$. The results are averaged over 10 tries.

$\|\Sigma\|$	ℓ	Avg. of max frequency	$\|\mathcal{T}_1\|$ (2-App)	$\|\mathcal{T}_2\|$ (Heuristic)	$\mathcal{R}$
40	35	8.4	283	361	1.57
40	40	8.1	310	404	1.53
40	45	8.1	360	450	1.60
50	45	8.0	322	411	1.57
50	50	7.8	349	450	1.55
50	55	8.2	387	495	1.56
60	55	8.7	353	451	1.57
60	60	8.4	373	493	1.51
60	65	8.3	412	528	1.56
70	65	8.5	378	493	1.53
70	70	8.8	406	534	1.52
70	75	8.3	439	565	1.55
80	75	9.0	397	530	1.50
80	80	8.5	427	568	1.50
80	85	8.7	450	600	1.50

We next present heuristic algorithms for the Median Canonical Trajectory problem, which we believe is more practical and interesting compared with the Median Permutation Trajectory problem.

5.3 Heuristic Algorithms for the Median Canonical Trajectory Problem

First of all, it should be noted that the solution sequence for the median canonical trajectory problem needs to contain all the letters of Σ, and its length must be longer than the size of Σ. We design two heuristic algorithms for this problem. The first contains two steps. In the first step of the algorithm we construct a permutation sequence for the median permutation problem, using a greedy method. In the second step, we extend the permutation sequence to satisfy the required length of the problem.

Algorithm 5 The Greedy Permutation Algorithm (GPA)

Input: adjacency map $\mathcal{M}$, $I = |\Sigma|$
Output: sequence T with length I, number r of adjacencies between T and S_i's.

$T \leftarrow \varepsilon$, $T^{'} \leftarrow \varepsilon$
$(\mathcal{M}, T, r) \leftarrow$ Greedy1-c$(\mathcal{M}, T^{'})$
while $T \neq T'$ **do**
  $T^{'} \leftarrow T$
  if Greedy1-c$(\mathcal{M}, T^{'})$.r $>$ Greedy2-c$(\mathcal{M}, T^{'})$.r **then**
    $(\mathcal{M}, T, r) \leftarrow$ Greedy1-c$(\mathcal{M}, T^{'})$
  else
    $(\mathcal{M}, T, r) \leftarrow$ Greedy2-c$(\mathcal{M}, T^{'})$
  end if
end while
if $|T| < I$ **then**
  $(\mathcal{M}, T, r) \leftarrow$ Add1-c$(\mathcal{M}, T^{'})$
  while $T \neq T^{'}$ **do**
    $T^{'} \leftarrow T$
    $(\mathcal{M}, T, r) \leftarrow$ Add1-c$(\mathcal{M}, T^{'})$
  end while
end if
if $|T| < I$ **then**
  $T \leftarrow T + (I \setminus T)$
end if

The first heuristic algorithm is fully based on the greedy idea. We use an idea similar to that of Algorithm 4. As a permutation sequence needs to be generated in the first step, we add one constraint to the Greedy1, Greedy2 and Add1 algorithms to force that only the letters which are not in the temporary sequence T can be selected. These modified algorithms are called Greedy1-c, Greedy2-c and Add1-c respectively. These are summarized in Algorithm 5, which returns a sequence which is a permutation. After the permutation sequence is obtained, we use Greedy1, Greedy2 and Add1 to construct the final solution to meet the length constraint. The whole greedy algorithm is shown in Algorithm 6, which we also call it GPA+Greedy.

We also implement another heuristic algorithm which is similar to Algorithm 6 (GPA+Greedy). The difference is to replace the GPA algorithm with the 1.5-approximation to construct a permutation trajectory first, but it still uses the greedy idea to extend the sequence to a target length. We loosely call this algorithm the 1.5-approximation algorithm (1.5-App + Greedy, for short).

5.4 Empirical Results for the Median Canonical Trajectory Problem

We use the same setup to generate all the test sequences and test these two heuristic algorithms. All the test results are in Table 3 and Table 4 for the corresponding datasets. From these empirical results, we can see that the second heuristic algorithm has a little bit better performance (always 2.6-6.7% better).

Table 2 Comparison of the 2-approximation and the heuristic algorithms for the Median Trajectory problem. Here the data have a larger range, $|S_i| \in [2|\Sigma| - 10, 2|\Sigma| + 10]$ and similarly for ℓ. The results are averaged over 10 tries.

$\|\Sigma\|$	ℓ	Avg. of max frequency	$\|\mathcal{T}_1\|$ (2-App)	$\|\mathcal{T}_2\|$ (Heuristic)	$\mathcal{R}$
40	30	8.3	247	310	1.59
40	35	8.1	286	357	1.60
40	40	8.2	317	406	1.56
40	45	7.8	361	453	1.59
40	50	8.4	395	504	1.58
50	40	8.1	284	369	1.54
50	45	8.4	314	404	1.55
50	50	8.3	349	450	1.55
50	55	8.6	385	496	1.55
50	60	8.0	414	534	1.55
60	50	8.8	313	416	1.50
60	55	8.3	353	452	1.56
60	60	8.4	373	492	1.52
60	65	8.6	407	527	1.54
60	70	8.2	432	565	1.53
70	60	8.6	346	460	1.50
70	65	8.7	380	493	1.54
70	70	8.8	405	531	1.53
70	75	8.8	431	564	1.53
70	80	8.4	453	599	1.51
80	70	8.4	373	503	1.48
80	75	8.8	396	530	1.49
80	80	8.5	420	566	1.48
80	85	8.5	444	598	1.48
80	90	8.6	473	630	1.50

We comment that in this case the comparison is mainly on Algorithm 6 (i.e., GPA+Greedy) and the 1.5-approximation algorithm (1.5-App + Greedy). We are not able to say anything regarding the approximation factor in this case — since no approximation is known for the median canonical trajectory problem at the point.

Algorithm 6 The Greedy Median Canonical Trajectory Algorithm

Input: sequences $S_i (i = 1..m)$, $I = |\Sigma|$, integer $\ell > I$
Output: sequence T with length ℓ, 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.
$(\mathcal{M}, T, r) \leftarrow \text{GPA}(\mathcal{M}, I)$
$\ell \leftarrow \ell - I$
while $\ell > 1$ **do**
  if Greedy1$(\mathcal{M}, T)$.r > Greedy2$(\mathcal{M}, T)$.r **then**
    $(\mathcal{M}, T, r) \leftarrow \text{Greedy1}(\mathcal{M}, T)$
  else
    $(\mathcal{M}, T, r) \leftarrow \text{Greedy2}(\mathcal{M}, T)$
  end if
  $\ell \leftarrow \ell - 2$
end while
if $\ell > 0$ **then**
  $(\mathcal{M}, T, r) \leftarrow \text{Add1}(\mathcal{M}, T)$
end if

Table 3 Comparison of the greedy and the 1.5-approximation algorithms for the Median Canonical Trajectory problem. Here we have $|S_i| \in [2|\Sigma| - 5, 2|\Sigma| + 5]$. The results are averaged over 10 tries.

$\|\Sigma\|$	ℓ	Avg. of max frequency	$\|\mathcal{T}_1\|$ (GPA + Greedy)	$\|\mathcal{T}_2\|$ (1.5-App + Greedy)
40	45	7.0	346	361
40	50	6.7	385	399
40	55	6.8	422	437
50	55	7.1	370	389
50	60	6.9	410	428
50	65	6.7	440	458
60	65	7.3	399	423
60	70	7.6	429	453
60	75	7.3	470	490
70	75	7.3	428	453
70	80	7.3	456	480
70	85	7.2	487	513
80	85	7.4	455	480
80	90	7.3	480	508
80	95	7.3	513	540

6 Concluding Remarks

Applying (and extending) the concept of adjacency from computational genomics, we try to compare the similarity of trajectories in a vehicular network, which we propose to use as the first step to build trustfulness in vehicular networks. Naturally, a trajectory is a sequence of POIs visited by a vehicle in

Table 4 Comparison of the greedy and the 1.5-approximation algorithms for the Median Canonical Trajectory problem. Here the data have a larger range, $|S_i| \in [2|\Sigma| - 10, 2|\Sigma| + 10]$ and similarly for ℓ. The data are averaged over 10 tries.

$\|\Sigma\|$	ℓ	Avg. of max frequency	$\|\mathcal{T}_1\|$ (GPA + Greedy)	$\|\mathcal{T}_2\|$ (1.5-App + Greedy)
40	45	7.0	341	357
40	50	7.7	386	401
40	55	7.0	427	441
40	60	6.9	463	480
40	65	7.2	499	512
50	55	7.2	371	390
50	60	7.1	406	426
50	65	7.1	444	462
50	70	7.1	475	494
50	75	7.1	512	527
60	65	7.4	397	420
60	70	7.5	432	452
60	75	6.9	467	789
60	80	6.9	499	520
60	85	7.0	533	551
70	75	7.5	422	448
70	80	7.5	562	487
70	85	7.5	490	517
70	90	7.5	519	542
70	95	7.0	548	573
80	85	7.3	453	483
80	90	7.2	485	513
80	95	7.7	509	535
80	100	7.8	540	568
80	105	7.4	568	595

one day. Given a set of such trajectories, this paper focuses on computing a consensus trajectory, where the objective is to maximize the total number of adjacencies between the consensus trajectory and the input trajectories. We investigate three versions of the problem: Median Trajectory, Median Permutation and Median Canonical Trajectory, which we prove are all NP-hard. For the first two problems, we are able to give factor-2 and factor-1.5 approximation algorithms. To handle practical datasets, for the Median Trajectory problem, we also design a heuristic algorithm based on a new 'adjacency map' data structure. This heuristic outperforms the 2-approximation algorithm when tested over articulately generated simulated datasets. The actual approximation factor is in the range $[1.48, 1.6]$, even though in theory the factor 2 is what we can prove in this paper.

For the Median Canonical Trajectory problem, we implement and test two heuristic algorithms. The first heuristic algorithm is a simple greedy algorithm. The second heuristic algorithm is based on the 1.5-approximation algorithm for the Median Permutation problem. The empirical results show that the second heuristic algorithm has a slightly better performance, i.e., about 2.6-6.7% better.

There are still some open questions on this research. For instance, does Median Canonical Trajectory admit a constant factor approximation? Can the approximation factor for Median Trajectory be improved to be below 2? Can the approximation factor for Median Permutation be improved to be below 1.5? Are there any corresponding inapproximability lower bounds? These questions definitely deserve some further research.

Acknowledgments

We thank anonymous reviewers for their comments. Letu Qingge was additionally supported by College of Engineering Research Startup Funds at North Carolina A&T State University. Peng Zou was also supported by a COE Benjamin PhD Fellowship at Montana State University.

References

Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., and Vialette, S. (2009). On the approximability of comparing genomes with duplicates. *J. Graph Algorithms Appl.*, 13(1):19–53.

Bryant, D. (1998). The complexity of the breakpoint median problem technical report crm-2579. *Centre de recherches mathématiques, Université de Montréal*.

Douceur, J. R. (2002). The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Revised Papers*, volume 2429, pages 251–260. Springer.

Edmonds, J. R. and Johnson, E. L. (2001). Matching: A well-solved class of integer linear programs. In *Combinatorial Structures and Their Applications (Gordon and Breach, New York)*, volume 2570, pages 27–30. Springer.

Hartenstein, H. and Laberteaux, K. P. (2010). *VANET: Vehicular Applications and Inter-Networking Technologies*. Wiley.

Jiang, H., Zheng, C., Sankoff, D., and Zhu, B. (2012). Scaffold filling under the breakpoint and related distances. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 9(4):1220–1229.

Jiang, H., Zhong, F., and Zhu, B. (2011). Filling scaffolds with gene repetitions: Maximizing the number of adjacencies. In *Combinatorial Pattern Matching - 22nd Annual Symposium, CPM 2011, Proceedings*, volume 6661, pages 55–64. Springer.

Kara, B. Y. and Verter, V. (2004). Designing a road network for hazardous materials transportation. *Transp. Sci.*, 38(2):188–196.

Kchaou, A., Abassi, R., and Fatmi, S. G. E. (2018). Toward a distributed trust management scheme for VANET. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, volume 53, pages 1–6. ACM.

Li, M., Ma, B., and Wang, L. (2002). Finding similar regions in many sequences. *J. Comput. Syst. Sci.*, 65(1):73–96.

Liu, G., Chen, Q., Yang, Q., Zhu, B., Wang, H., and Wang, W. (2017). Opinionwalk: An efficient solution to massive trust assessment in online social networks. In *IEEE Conference on Computer Communications, INFOCOM 17*, pages 1–9. IEEE.

Liu, G., Yang, Q., Wang, H., Lin, X., and Wittie, M. P. (2014). Assessment of multi-hop interpersonal trust in social networks by three-valued subjective logic. In *IEEE Conference on Computer Communications, INFOCOM 14*, pages 1698–1706. IEEE.

Papadimitratos, P., Buttyán, L., Holczer, T., Schoch, E., Freudiger, J., Raya, M., Ma, Z., Kargl, F., Kung, A., and Hubaux, J. (2008). Secure vehicular communication systems: design and architecture. *IEEE Commun. Mag.*, 46(11):100–109.

Patel, N. J. and Jhaveri, R. H. (2015). Trust based approaches for secure routing in vanet: A survey. *Procedia Computer Science*, 45:592–601.

Pe'er, I. and Shamir, R. (1998). The median problems for breakpoints are np-complete. *Electron. Colloquium Comput. Complex.*, 5(71).

Qingge, L., Zou, P., Dai, L., Yang, Q., and Zhu, B. (2019). Trajectory comparison in a vehicular network II: eliminating the redundancy. In *Wireless Algorithms, Systems, and Applications - 14th International Conference, WASA 2019*, volume 11604, pages 260–271. Springer.

Qingge, L., Zou, P., Dai, L., Yang, Q., and Zhu, B. (2021). On comparing the similarity and dissimilarity between two distinct vehicular trajectories. *IEEE Access*, 9:34415–34422.

Shiloach, Y. (1981). Another look at the degree constrained subgraph problem. *Inf. Process. Lett.*, 12(2):89–92.

Shrestha, R., Djuraev, S., and Nam, S. Y. (2014). Sybil attack detection in vehicular network based on received signal strength. In *International Conference on Connected Vehicles and Expo, ICCVE 2014*, pages 745–746. IEEE.

Tannier, E., Zheng, C., and Sankoff, D. (2009). Multichromosomal median and halving problems under different genomic distances. *BMC Bioinform.*, 10.

Xin, C., Qingge, L., Wang, J., and Zhu, B. (2015). Robust optimization for the hazardous materials transportation network design problem. *J. Comb. Optim.*, 30(2):320–334.

Zhang, J. (2011). A survey on trust management for vanets. In *25th IEEE International Conference on Advanced Information Networking and Applications, AINA 2011*, pages 105–112. IEEE Computer Society.

Zou, P., Qingge, L., Yang, Q., and Zhu, B. (2019). Trajectory comparison in a vehicular network I: computing a consensus trajectory. In *Wireless Algorithms, Systems, and Applications - 14th International Conference, WASA 2019*, volume 11604, pages 533–544. Springer.