Trajectory Comparison in a Vehicular Network II: Eliminating the Redundancy

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

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

Abstract. This paper investigates the truthfulness establishment problem between two nodes (vehicles) in a vehicular network. We focus more on the case when no interaction has been conducted and we use the Point of Interests (POIs) visited by the two nodes (vehicles) to establish the initial truthfulness. It turns out that this is a general version of a well-studied problem in compuational genomics called CMSR (Complementary Maximal Strip Recovery) in which the letters (similar to POIs) cannot be duplicated, while in our problem POIs could certainly be duplicated. We show that one version (when noisy POIs are deleted all the remaining POIs must be involved in some adjacency), is NP-hard; while the other version (with the adjacency involvement constraint is dropped), is as hard as Set Cover. We then design an ILP solution for the first problem. Simulations with various synthetic data show that the algorithm is very effective.

1 Introduction

In a vehicular network, an important problem is that when two vehicles (nodes) communicate, one needs to determine the trustfulness of the other while privacy is being preserved. 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 [14, 15].

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 keep away parking competitors. The latter could be a generation of fake identities to falter the functioning of the whole system [13, 18]. The public-key infrastructure might not be available over a road network; hence, some kind of trust management must be maintained [19, 16, 17].

One challenge to build such a trust management system is how to build the initial trustfulness between two vehicles while preserving the privacy. Recently a

method was proposed to compute the truth-telling probability, which is in turn decided by the opinions of adjacent vehicles [12]. Such information might not be available sometimes, for example, a man who lives remotely and works at home might not have a big chance to connect to a VANET on a regular basis.

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. 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 "home", "restaurant", "coffee shop", "gym", etc. (Sometimes, 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.)

Now suppose that we have the list of POIs visited by two vehicles over some time period, say two weeks. How do we compare these sequences (of POIs)? We adopt a classic concept called *adjacency*, which is used widely in computational genomics [1, 7, 8]. (We note that in computing genomic maps, this concept of adjacency is only applied on permutations, i.e., each letter appears exact once in the input and in the final solution [20]. This is different for our purpose.) While computing the number of adjacencies between two sequence or two permutations are relatively easy, in our applications (and also in computational genomics applications), a more important and practical problem is to throw away some noisy POIs to obtain the true similarity between two sequence of POIs. A variation of this problem has been studied in the biological community, as the Maximal Strip Recovery (MSR) and Complementary Maximal Strip Recovery (CMSR) problems. (Here a strip is a common substring appears in both sequence, one of which could be in reversed form. For both problem all the remaining letters must be involved in some adjacency.)

The MSR problem was first studied by the Sankoff group at University of Ottawa [5, 21]. The MSR problem, and its complement CMSR, have been shown NP-hard [20, 3, 4]. Late they have been shown to be APX-complete [3, 9]. MSR is known to admit a 4-approximation algorithm [4]. (This is achieved by converting the MSR problem to computing the maximum independent set in t-interval graphs, which admits a $2t$-approximation [2].) CMSR have been shown to admit a factor-3 approximation [6] and the best current approximation factor is 2.33 [10].

2 Preliminaries

At first, we make some necessary definitions. Given a set Σ of POIs (or just letters), 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 all POIs of A, which is a multi-set of elements in Σ. For example, Σ = {*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*. Throughout this paper, the relative order of the two letters in 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 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 use $b_A(A, B)$ and $b_B(A, B)$ to denote the set of breakpoints in A and B respectively. We illustrate the above definitions in Fig. 1.

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

Fig. 1. An example for adjacency, breakpoint and the related definitions.

Note that our breakpoint and adjacency definitions are more general than those for permutations. Let P, Q be permutations over Σ with length n. Let $\tilde{P}$ (resp. $\tilde{Q}$) be the *reversal* of P (resp. Q). Then $|b_P(P, Q)| = |b_Q(P, Q)| = n - 1 - |a(P, Q)|$; moreover, if $|b_P(P, Q)| = |b_Q(P, Q)| = 0$ then either $P = Q$, or $P = \tilde{Q}$. For general sequences, this is not necessarily true. For instance, $A = dabdcba$, $B = dcbabda$, and there is no breakpoint between A and B. But $A \neq B$ and $A \neq \tilde{B}$. Nonetheless, when there are more adjacencies between A and B, they are similar intuitively.

In essence, we need to compare the similarity between A and B when some redundant POIs are deleted. (To start with, any POI family not in A and B at the same time should deleted. Hence we assume that A and B are over the same set of POIs Σ.)

Now, we define the problems we study in this paper formally.

Definition 2. *Redundant POI Deletion (RPD).*

Input: *two sequences A and B of length at most n over a POI set Σ, and two positive integers k, ℓ.*
Question: *Can a total of k letters (POIs) be deleted from A and B to obtain A^* and B^* such that $c(A^*) = c(B^*)$, $|a(A^*, B^*)| \geq \ell$ and all letters in A^* and B^* are involved in some adjacency in $a(A^*, B^*)$?*

If two sequences X and Y are not inherently similar, requiring that the remaining letters in X and Y are all involved in some adjacency might be too much. Hence, we could have a different version of the problem.

Definition 3. *POI Deletion to Maximize the Number of (String) Adjacencies (PD-MNSA).*
Input: *two sequences X and Y of length at most n over a POI set Σ, and two positive integers k, ℓ.*
Question: *Can a total of k letters (POIs) be deleted from X and Y to obtain X^* and Y^* such that $c(X^*) = c(Y^*)$ and $|a(X^*, Y^*)| \geq \ell$?*

We comment that the two problems are very different. As we will show in the next section, the second problem is at least as hard as Set Cover (which cannot be approximated with $C \log n$ unless P=NP), while the first problem can only be shown to be NP-hard.

3 Hardness Results

3.1 RPD is NP-Complete

Note that in reality we are really interested in the optimization version of the RPD problem, i.e., deleting the minimum number of POIs in the input trajectories A, B to maximize the number of adjacencies between the resulting trajectories A', B' and each letter in A', B' is involved in some adjacency. For the hardness result we simply look at the decision of the RPD problem (RPD for short). This problem is a generalization of the Complementary Maximal Strip Recovery problem in which case no POI can be duplicated in the input T_1, T_2, and even this restricted version of the problem is NP-complete [20]. Here we give a simple proof.

Theorem 1. *The Redundant POI Deletion problem is NP-complete.*

Proof. It is easy to see that the Redundant POI Deletion (RPD) problem is in NP. We now show that Exact Cover by 3-Sets (X3C) can be reduced to RDP. Given a base set $X = \{x_1, x_2, ..., x_{3q}\}$ and a set $S = \{S_1, S_2, ..., S_m\}$, where $S_i \subseteq X$ and $|S_i| = 3$, the question is whether we could identify q sets in S which collectively cover all the $n = 3q$ elements in X. In our construction, for each x_i we create an x'_i. Let $S_i = \{x_{i1}, x_{i2}, x_{i3}\}$, we construct a string $T_i = x_{i1}x'_{i1}x_{i2}x'_{i2}x_{i3}x'_{i3}$. We construct two trajectories as follows.

$$A = (c_1c_2)^n a_1b_1T_1a_2b_2T_2 \cdots a_mb_mT_ma_{m+1}b_{m+1}\#\#,$$

$$B = x_1x'_1c_1c_2x_2x'_2c_1c_2 \cdots x_nx'_nc_1c_2 \cdot \#\#a_1b_1a_2b_2 \cdots a_mb_ma_{m+1}b_{m+1}.$$

Note that in B, all the 2-substrings c_1c_2's, $x_ix'_i$'s and a_ib_i's (before and after $\#\#$) are already forming $2n + (m + 1) + 1 = 2n + m + 2$ adjacencies with the correspondinging ones in A. Hence the problem is really to delete letters in T_i's to form new adjacencies in the form b_ja_{j+1}. We make the following claim: X3C has a solution of size q iff the RDP problem has a solution of size $\langle 6(m-q), 2n+2m-q+2\rangle$ (i.e., $6(m-q)$ letters can be deleted from A, B to form $2n + 2m - q + 2$ adjacencies and all the remaining letters are in some adjacency). □

We show a simple example for the reduction. $X = \{1, 2, 3, 4, 5, 6, a, b, c, d, e, f\}$. $S_1 = \{1, 2, 3\}$, $S_2 = \{1, 5, a\}$, $S_3 = \{4, 5, 6\}$, $S_4 = \{a, b, c\}$, $S_5 = \{4, b, e\}$, $S_6 = \{d, e, f\}$. Our construction is then

$$A = (c_1c_2)^{12}a_1b_111'22'33'\cdot a_2b_211'55'aa'\cdot a_3b_344'55'66'\cdot a_4b_4aa'bb'cc'\cdot a_5b_544'bb'ee'$$
$$\cdot a_6b_6dd'ee'ff'a_7b_7\#\#,$$
$$B = 11'c_1c_2 \cdot 22'c_1c_2 \cdot 33'c_1c_2 \cdot 44'c_1c_2 \cdot 55'c_1c_2 \cdot 66'c_1c_2 \cdot aa'c_1c_2 \cdot bb'c_1c_2 \cdot cc'c_1c_2$$
$$\cdot dd'c_1c_2 \cdot ee'c_1c_2 \cdot ff'c_1c_2 \cdot \#\# \cdot a_1b_1 \cdot a_2b_2 \cdot a_3b_3 \cdot a_4b_4 \cdot a_5b_5 \cdot a_6b_6 \cdot a_7b_7.$$

As the unique solution for this X3C instance is $\{S_1, S_3, S_4, S_6\}$, we need to delete the 12 letters in T_2 and T_5 from A to obtain A', and notice that $B' = B$.

$$A' = (c_1c_2)^{12}a_1b_111'22'33'\cdot a_2b_2\cdot a_3b_344'55'66'\cdot a_4b_4aa'bb'cc'\cdot a_5b_5\cdot a_6b_6dd'ee'ff'a_7b_7\#\#.$$

The two new adjacencies after T_2, T_5 are deleted are: b_2a_3 and b_5a_6. As there are already $2n + m + 2 = 32$ adjacencies between A and B and the deleted letters are all from A, the total number of adjacencies between A' and B is $2n + 2m - q + 2 = 34$.

Note that the following extensions can be obtained:

(1) The problem remains NP-complete when A' and B' must be permutations and when the deletions of letters only occur at one of A and B. This can be done by making all the c_1c_2's distinct and by changing $\#\#$ to $\#_1\#_2$.

In the above reduction, notice that all the letters in A' and B' are involved in some adjacency. If we drop this condition, then the problem is much harder.

3.2 PD-MNSA is As Hard As Set-Cover

We first recall the Set Cover problem. Here the input to set cover is $X = \{x_1, x_2, ..., x_n\}$ and $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, where $S_i \subset X$ and $\cup_i(S_i) = X$. The objective is to find minimum number (say k) of subsets in $\mathcal{S}$ such that they collectively cover X. We consider a special version of Set Cover, in which all S_i's have the same size Δ. We call this version Δ-Set Cover. The following lemma is not hard to obtain.

Lemma 1. *Δ-Set Cover is as hard as Set Cover, both in terms of polynomial time approximability and FPT tractability.*

Proof. We reduce Set Cover to Δ-Set Cover. Let the input to set cover be $X = \{x_1, x_2, ..., x_n\}$ and $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, where $S_i \subset X$ and $\cup_i(S_i) = X$. Let $\Delta = \max_i |S_i|$.

We construct a set of Δ new elements, $S' = \{y_1, ..., y_{\Delta-1}, z\}$. The instance for Δ-Set Cover is constructed as follows. First set $X' = X \cup S'$, and clearly $|X'| = |X| + \Delta = n + \Delta < 2n$. For all S_i, if $|S_i| < \Delta$, then we take a subset of $\Delta - |S_i|$ elements from $S' - \{z\}$ and union it with S_i to obtain S'_i; otherwise, set $S'_i \leftarrow S_i$. Then, we have $\mathcal{S}' = \{S'_1, S'_2, ..., S'_m, S'\}$. It is obvious that Set Cover has a solution of size k iff Δ-Set Cover has a solution of size $k + 1$. We omit the details as the only notice one should pay attention to is that S' must be included in any solution for Δ-Set Cover — as the elements $z \in X'$ is only covered by S'. □

We now reduce Δ-Set Cover to POI Deletion to Maximize the Number of (String) Adjacencies (PD-MNSA).

Theorem 2. *The POI Deletion to Maximize the Number of (String) Adjacencies (PD-MNSA) problem is NP-complete.*

Proof. We reduce Δ-Set Cover to POI Deletion to Maximize the Number of (String) Adjacencies (PD-MNSA). Let the input to Δ-Set Cover be $X = \{e_1, ..., e_n\}$ and $\mathcal{S} = \{S_1, ..., S_m\}$, with $S_i \subseteq X'$ and $|S_i| = \Delta$ for $i = 1..m$. For each e_i, let $f(i)$ be the number of sets in $\mathcal{S}$ which contains e_i.

In the following, i and $\overline{i}$ are all letters (POIs). Let $S_i = \{e_{i,1}, e_{i,2}, ..., e_{i,\Delta}\}$, we build a string $T_i = e_{i,1}e_{i,2}\cdots e_{i,\Delta}$. We construct two permutations: $Y(n) = (2n+1)(2n-1)\cdots 3\cdot 1\cdot (2n+2)(2n)\cdots 4\cdot 2$ and $Z(m,n) = (\overline{2n+1}\cdot\overline{2n-1}\cdots\overline{3}\cdot\overline{1})^m \cdot (\overline{2n+2}\cdot\overline{2n}\cdots\overline{4}\cdot\overline{2})^m$.

We construct two sequences G and H as follows.

$$G = \overline{1}^{m(n+1)}\overline{2}^{m(n+1)}\cdots\overline{2n+2}^{m(n+1)}\cdot a_1T_1b_1\cdot a_2T_2b_2\cdots a_mT_mb_m$$
$$\cdot 1^{n+1}2^{n+1}\cdots(2n+2)^{n+1},$$
$$H = Y(n)e_1^{f(1)}Y(n)\cdot e_2^{f(2)}Y(n)\cdots e_n^{f(n)}Y(n)$$
$$\cdot a_1b_1Z(m,n)\cdot a_2b_2Z(m,n)\cdots a_{m-1}b_{m-1}Z(m,n)\cdot a_mb_mZ(m,n).$$

Assuming that $k < n, k < m$, we have the following facts:

Fact 1: To obtain some adjacency in the form of $\overline{i}\cdot\overline{i+1}$ or $i(i+1)$, with $i \neq 2n+1$, one needs to delete at least $n+1$ letters.

Fact 2: To obtain some adjacency in the form of $a_i\overline{j}$ or $b_i\overline{j}$ or a_ij or b_ij, with $i \leq m$ and $j \leq 2n+2$, one needs to delete at least $n+1$ letters.

Fact 3: To obtain some adjacency in the form of $e_i\overline{j}$ or e_ij, with $i \leq n$ and $j \leq 2n+2$, one needs to delete at least $n+1$ letters.

Fact 4: To obtain some adjacency in the form of $e_i e_j$, with $i, j \leq n$, one needs to delete at least $2n + 2$ letters.

Hence, the only possibility to obtain an adjacency in the form $a_i b_i$ is to delete T_i, with is of length $\Delta < n$.

Therefore, we have the following claim: Δ-Set Cover has a solution of size k iff Δk letters are deleted from both G and H to obtain k adjacencies. (Note that to make sure that $c(G') = c(H')$, after each T_i is deleted, we must delete the corresponding e_j's in them from H as well.) □

Corollary 1. *The optimization version of PD-MNSA does not admit a factor $C \log n$ approximation unless P=NP.*

Proof. The above two reductions in Lemma 1 and Theorem 2 are both L-reductions. As Set Cover cannot be approximated with a factor of $C \log n$ for some constant C (unless P=NP) [11], the same claim holds for PD-MNSA. □

An intriguing problem is whether RPD admits a constant factor approximation. (Recall that the more restricted version of RPD, where the input sequences are permutations, does admit constant factor approximations [6, 10].) To solve the problem practically, we will design an ILP solution for the RPD problem.

4 A Practical Solution for Redundant POI Deletion

—we have the algorithm and implementation already, just need to obtain the empirical results —

Acknowledgments

This research is supported by NSF under project CNS-1761641.

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. R. Bar-Yehuda, M. M. Halldórsson, J. S. Naor, H. Shachnai, and I. Shapira. Scheduling split intervals. *SIAM Journal on Computing*, 36:1–15, 2006.
3. L. Bulteau, G. Fertin and I. Rusu. Maximal strip recovery problem with gaps: hardness and approximation algorithms. *Proceedings of the 20th Annual International Symposium on Algorithms and Computation (ISAAC'09)*, LNCS 5878, pages 710-719, 2009.
4. Z. Chen, B. Fu, M. Jiang, and B. Zhu. On recovering synthetic blocks from comparative maps. *Journal of Combinatorial Optimization*, 18:307–318, 2009.
5. V. Choi, C. Zheng, Q. Zhu, and D. Sankoff. Algorithms for the extraction of syntheny blocks from comparative maps. In *Proceedings of the 7th International Workshop on Algorithms in Bioinformatics (WABI'07)*, pages 277–288, 2007.

6. H. Jiang, Z. Li, G. Lin, L. Wang and B. Zhu. Exact and approximation algorithms for the complementary maximal strip recovery problem. *J. of Combinatorial Optimization*, 23(4):493-506, 2012.
7. 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.
8. 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.
9. M. Jiang. Inapproximability of maximal strip recovery. *Theoretical Computer Science*, 412(29):3759-3774, 2011.
10. G. Lin, R. Goebel, Z. Li and L. Wang. An improved approximation algorithm for the complementary maximal strip recovery problem. *J. Comput. Sys. Sci.*, 78(3):720-730, 2012.
11. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. *Proc. 29th ACM Symp. on Theory Comput. (STOC'97)*, pages 475-484, 1997.
12. R. Shrestha and S.Y. Nam. Trustworthy Event-Information Dissemination in Vehicular Ad Hoc Networks. *Mobile Information Systems*, 9050787:1-16, 2017.
13. J. Doucer. The Sybil attack. *Proc. IPTPS'01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251-260, 2002.
14. H. Hartenstein and L. Kenneth. *VANET: Vehicular Applications and Inter-Networking Technologies*. Wiley, New Jersey, USA, 2009.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. L. Wang and B. Zhu. On the tractability of maximal strip recovery. *J. of Computational Biology*, 17(7):907-914, 2010.
21. C. Zheng, Q. Zhu, and D. Sankoff. Removing noise and ambiguities from comparative maps in rearrangement analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:515–522, 2007.