Offline Map Matching Using Time-Expanded Graph for Low-Frequency GPS Data

田中, 智

KYUSHU UNIVERSITY

DOCTORAL THESIS

Offline Map Matching Using Time-Expanded Graph for Low-Frequency GPS Data

---

Author:
Akira TANAKA

Supervisor:
Dr. Katsuki FUJISAWA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Mathematics

in the

Graduate School of Mathematics

January 7, 2022

KYUSHU UNIVERSITY

Abstract

Graduate School of Mathematics

Doctor of Mathematics

Offline Map Matching Using Time-Expanded Graph for Low-Frequency GPS Data

by Akira TANAKA

Map matching is an essential preprocessing step for most trajectory-based intelligent transport system services. Due to device capability constraints and the lack of a high-performance model, map matching for low-sampling-rate trajectories is of particular interest. Therefore, we developed a time-expanded graph matching (TEG-matching) that has three advantages (1) high speed and accuracy, as it is robust for spatial measurement error and a pause such as at traffic lights; (2) being parameter-free, that is, our algorithm has no predetermined hyperparameters; and (3) only requiring ordered locations for map matching. Given an entire low-frequency GPS data, we construct a time-expanded graph (TEG) whose path from source to sink represents a candidate route. We find the shortest path on TEG to obtain the matched route with a small area between the vehicle trajectory. Additionally, we introduce two general speedup techniques (most map matching methods can apply) bottom-up segmentation and fractional cascading. Numerical experiments with worldwide vehicle trajectories in a public dataset show that TEG-matching outperforms state-of-the-art algorithms in terms of accuracy and speed, and we verify the effectiveness of the two general speedup techniques. Moreover, we propose an upgraded model, called NewTEG-matching, to solve a theoretical limitation and complex calculation of TEG-matching. NewTEG-matching is more straightforward, intuitive, and high-speed, but the comprehensive experiments are left for our future work.

Acknowledgements

First and foremost, I would like to show my extreme gratitude to my supervisor, Professor Katsuki Fujisawa, who gave me invaluable advice, continuous support, and words of encouragement during my MS and Ph.D. study. His immense knowledge and ample experience have educated and encouraged me in my academic research and daily life. I would also like to thank my lab mates for their significant discussions and fantastic leisure time. I would like to thank my lab's technical staff, Mrs. Tomoko Sakai and Mrs. Kyoko Ikebe, for their consideration, such as simplifying office procedures and organizing refresh events. Additionally, my gratitude extends to the cooperative research partner companies for providing an opportunity to develop real-world applications. I also appreciate colleagues in the National Institute of Information and Communications Technology (NICT). They support me in succeeding in a new academic field and spending better life in a relocation site. Finally, I would like to express my gratitude to my family and friends for their support and warm encouragement.

Contents

List of Figures

List of Tables

List of Symbols

V	set of *nodes* and *junctions* (see Figure 1.1)
A	set of *arcs* (see Figure 1.1)
$\mathrm{tail}(a)\ (a \in A)$	tail of an *arc a*
$\mathrm{head}(a)\ (a \in A)$	head of an *arc a*
G	*road network*
$\boldsymbol{P} = (P_i)_{i=1}^{n+1}$	a *vehicle trajectory* from time stamp 1 to $n+1$
$d(X, Y)$	Euclidean distance between X and Y where X (Y) is either an element or a subset of Euclidean space.
$d(X, Y)_\infty$	Chebyshev distance between X and Y where X (Y) is either an element or a subset of Euclidean space.
$\mathbb{R}^n$	n-dimensional Euclidean space
$\lVert\boldsymbol{x}\rVert_p$	ℓ^p norm of a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \mathbb{R}^n$, that is, $\lVert\boldsymbol{x}\rVert_p := \left(\sum_{i=1}^n \lvert x_i\rvert^p\right)^{1/p}$
$\lVert\boldsymbol{x}\rVert_\infty$	maximum norm of a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, that is, $\lVert\boldsymbol{x}\rVert_\infty := \max\{\lvert x_1\rvert, \ldots, \lvert x_n\rvert\}$
$T(G, \boldsymbol{P})$	time-expanded graph

Chapter 1

Introduction

1.1 Background

Map-matching algorithms determine the user or vehicle travel route by aligning the discrete positioning data to the road network and are driven by the ubiquity and improvement of positioning devices. According to functional scenarios and applications, current map-matching solutions can be categorized into online and offline map-matching. Online map-matching processes the current sample with a limited number of preceding or succeeding samples (Goh et al., 2012; Yin et al., 2018). The process is often fast and straightforward for interactive performance and is used for route guidance, autonomous cars, collision avoidance systems, lane departure warning, emergency response, and enhanced driver awareness systems (White, Bernstein, and Kornhauser, 2000; Toledo-Moreo, Betaille, and Peyret, 2010; Sathiaseelan, 2011). In contrast, offline map matching is performed after the entire trajectory is obtained, aiming for an optimal matching route with fewer processing time constraints. Offline map matching is utilized for traffic flow analysis, road pricing, traffic surveillance, and transport operations (Velaga and Pangbourne, 2014).

1.2 Related work

To review the current status of map-matching and determine future research directions, Chao et al. (2020) and Huang et al. (2021) classified map-matching models based on the technical perspective (core matching model) and the sampling frequency of positioning data, respectively. Chao et al. (2020) focused on categorizing only competitive algorithms while including new models that appear after the last comprehensive survey. These algorithms are classified into four categories: *similarity*, *state-transition*, *candidate-evolving*, and *scoring* models.

The *similarity* models return the vertices and arcs that are geometrically *closest* to the trajectory. Based on the definition of *closest*, the *similarity* models are divided into two subcategories. Distance-based models regard the spatial distance as the closeness between a trajectory and a matched path, and Fréchet distance (Alt et al., 2003; Wei et al., 2013) and the longest common subsequence (Zhu, Holden, and Gonder, 2017) are commonly used approaches. Meanwhile, the pattern-based algorithm assumes that people tend to travel on the same paths given a pair of origin and destination points. Historical map-matched data were utilized to determine similar travel patterns (Zheng et al., 2012).

The *state-transition* model builds a weighted topological graph whose vertices represent the possible *state* where the vehicle may be located at a particular moment. The arcs represent the *transitions* between states at different timestamps. The matched path is then obtained from the optimal path in the graph globally. There are three primary ways to build a graph and solve the optimal path problem: the hidden Markov model (HMM) (Newson and Krumm, 2009; Goh et al., 2012), conditional random field (CRF) (Hunter, Abbeel, and Bayen, 2014), and weighted graph technique (WGT) (Hsueh and Chen, 2018; Hu et al., 2017). In the HMM, each trajectory sample is regarded as the observation, while the actual location of the vehicle on the road, which is unknown, is the hidden state. The optimal path is obtained by the Viterbi algorithm, which is a dynamic programming approach. CRF is a statistical model and considers interactions among observations. HMM is also a statistical model and focuses only on the relationship between an observation and the state at the same stage.

The WGT enumerates candidate points for each positioning data and selects the most probable point sequence after creating a weighted candidate graph. A candidate point corresponds to any of the following things: (1) a road, (2) an endpoint of a road, and (3) a point on a road. For example, the spatio-temporal based matching algorithm (STD-matching) (Hsueh and Chen, 2018) and the AntMapper algorithm (Gong et al., 2018) use "(3) a point on a road" as a candidate point. Their algorithms calculate the shortest path problems between any pair of two consecutive candidate points to set the candidate graph's weight, but this incurs a high computational cost. Focusing on high-sampling-rate global positioning system (GPS) trajectories, Tang et al. (2016) introduced a time-dependent graph to address this problem. For each position fix, their algorithm utilizes the potential path area rather than the candidate points; hence, they do not need to solve the shortest path problems between two consecutive candidate points. Their algorithm finally produces a reasonable network-time path, representing the expected arc travel times and dwell times at possible intermediate stops. Although they succeeded in building both offline and real-time map-matching algorithms for high-sampling-rate trajectories, few studies have developed their model for low-sampling-rate trajectories. Therefore, we have developed a time-expanded graph matching (TEG-matching) for low-sampling-rate trajectories.

The *candidate-evolving* model holds a set of candidates (also known as particles or hypotheses) during map matching. The candidate set is initiated based on the first trajectory sample. It continues to evolve by adding new candidates propagated from the old ones close to the latest measurements while pruning the irrelevant ones. Compared to the *state-transition* model, the *candidate-evolving* type is more robust for off-track matching issues, and the particle filter (PF) (Wang and Ni, 2016; Bonnifait and Laneurit, 2009) and the multiple hypothesis technique (Taguchi, Koide, and Yoshimura, 2019; Knapen et al., 2018) are two representative solutions. The PF is a state estimation technique that combines Monte Carlo sampling methods with Bayesian inference. The PF model's general idea is to recursively estimate the probability density function (PDF) of the road network section around the observation as time advances. According to the moving status, the particles with higher weights are more likely to propagate to feed particles for the next cycle, while those with low weights are likely to die. The multiple hypothesis technique determines the scores to the candidate road edge (or point) rather than approximating the complicated PDF of the neighboring map area, which leads to a reduction in computation.

The *scoring* models (Quddus and Washington, 2015; Toledo-Moreo, Betaille, and Peyret, 2010) assign a group of candidates to each trajectory segment (or location observation) and finds an arc from each group that maximizes the predefined scoring function. According to the working scenario, every timestamp's discovered segment is either returned or joined with other matched segments.

On the other hand, Huang et al. (2021) classified map-matching algorithms based on the sampling frequency and insisted that GPS information is desired to be recorded at a lower frequency to save energy consumption and communication cost produced by large-scale GPS devices. Yuan et al. (2010) collected positioning data of more than 10,000 taxis in Beijing and revealed that 66% occupied the low-frequency data (sampling frequency is more than one minute). Therefore, map-matching algorithms based on low-frequency sampling data have attracted much attention in recent years (Huang et al., 2021; Chen et al., 2014; Yuan et al., 2010; Hsueh and Chen, 2018; Gong et al., 2018). Chen et al. (2014) proposed an online candidate-evolving map-matching algorithm for large-scale low-frequency data using multi-criteria dynamic programming (MDP). The MDP technique reduces the number of candidate routes when candidate routes stored from the previous position are extended to candidate routes at the current position. As for offline map-matching, (Yuan et al., 2010) developed an interactive voting-based map-matching algorithm. The authors consider spatial information, temporal information, and the mutual influence between matched points for neighboring GPS points. Recent works for tackling the same issue are STD-matching (Hsueh and Chen, 2018) and AntMapper algorithm (Gong et al., 2018), which utilizes WGT. STD-matching and the AntMapper algorithm enumerate candidate roads for each position data and use the longest path problem and an ant colony algorithm, respectively, to find the best combination of candidate roads.

Although these studies contribute to developing an offline map-matching algorithm for low-frequency data, there are the following shortcomings:

1. Few models achieve both high speed and accuracy.

2. Some methods require additional information such as velocity and angle, which are not obtained by some vehicles.

3. Some algorithms suffer from the tuning of hyperparameters.

Therefore, we utilized a concept from previous research (Tang et al., 2016) and developed a time-expanded graph matching (TEG-matching). We leverage the concept of TEG and formulate the map-matching problem as the shortest path problem, which achieves high speed and accuracy with only positioning data (Chapter 2). The elaborate weight on TEG makes the proposed model parameter-free (Section 2.2).

1.3 Problem setting

In this study, we handle a 2D *road network* containing two types *nodes*: *junctions* and *shape nodes*. A one-way road (hereinafter called *arc*) is represented by a polyline whose first and last points are *junctions*, and the other points are *shape nodes*. Let V be the set of *junctions* and *shape nodes*, A be the set of *arcs*, G be a *road network* that has *arc* set A. Then, *arc* a is represented as $a := (v_1, v_2, \ldots, v_m)$, $\{v_j\}_{j=1}^{m} \subset V$, and every two-way road is expressed as two one-way roads such as $(v_1, \ldots, v_m) \in A$ and $(v_m, v_{m-1}, \ldots, v_1) \in A$. A pair of two consecutive points of an *arc* is referred to as a *shape arc* and denoted by (v_i, v_{i+1}), where $v_i, v_{i+1} \in V$. A vehicle *trajectory* is a chronologically ordered position fixes $\boldsymbol{P} = (P_1, P_2, \ldots, P_{n+1})$ produced by a GPS device mounted on the vehicle. For each time step i, P_i includes only the east and north coordinates. The proposed algorithm aims to restore the most likely path $(a_1, a_2, \ldots, a_m)$ $(a_j \in A)$ of the vehicle under a given vehicle trajectory $\boldsymbol{P} = (P_i)_{i=1}^{n+1}$. The predicted path is referred to as the *matched path*, and the *arc* included in the *matched path* is called the *matched arc*. Figure 1.1 represents an example of map-matching for a vehicle trajectory $\boldsymbol{P} = (P_1, P_2, \ldots)$, and the *matched path* is $((v_0, v_1), (v_1, v_4, v_5), (v_5, v_6), \ldots)$. A *correct path* denotes a finite sequence of *arcs* on which the vehicle actually travels, and an *arc* included in the *correct path* is called a *correct arc*.

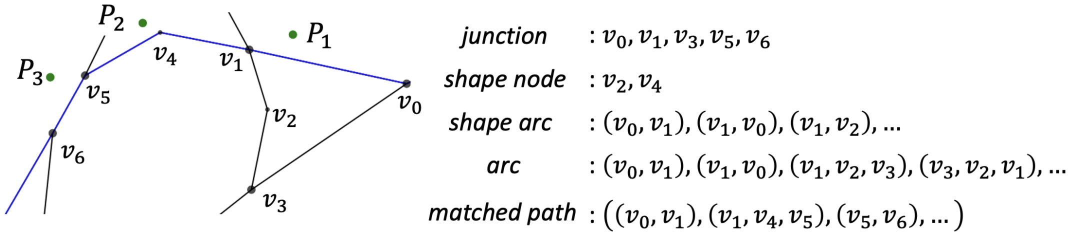

FIGURE 1.1: Example of a *road network*, a *vehicle trajectory*, and the corresponding *matched path*.

1.4 Contribution

Intuitively, if the area between a vehicle *trajectory* $(P_i)_{i=1}^{n+1}$ and a *matched path* is sufficiently small, the *match path* almost coincides with the *correct path*; hence, our algorithm finds the *matched path* with a small area. We first specify a potential area $L(G, P_i)$, where the vehicle may pass through from time step i to $i + 1$. Then, each pair of two consecutive position fixes (P_i, P_{i+1}) $(i \in \{1, 2, \ldots, n\})$ is simultaneously matched to a path (referred to as a *partial path*) on $L(G, P_i)$ while maintaining the connectivity of the two consecutive *partial paths*. A selected *partial path* tends to have few abrupt direction changes and creates a small area between the two position fixes. We formulate the matching as the shortest path problem on a time-expanded graph (TEG). In summary, our contributions are listed as follows:

- We propose a parameter-free offline map-matching approach called TEG-matching that requires only the vehicle's ordered locations. TEG-matching is robust for spatial measurement errors and pauses such as at traffic lights. We experimentally achieved a higher Jaccard index of 0.098 and a 5.6x faster outcome than two state-of-the-art algorithms, namely, the STD-matching (Hsueh and Chen, 2018) and the AntMapper algorithm (Gong et al., 2018).
- We applied fractional cascading (FC) to candidate *shape node* search and verified a 2.5x speedup compared with the k-d tree. To use FC, we also conducted a geometric analysis and answered "how big a square is needed to obtain all *arcs* within a radius r from a position fix."
- We applied the existing bottom-up segmentation to a road network and achieved a 64% reduction of shape nodes, resulting in a 1.78x speedup with only a 0.0074 accuracy drop for map matching.
- We propose an upgraded model, called newTEG-matching, to solve a theoretical limitation and complex calculation of TEG-matching in Appendix B.

1.5 Overview of TEG-matching

Proposed map-matching model follows these steps:

1. Preprocess

   (a) Define *junctions* and *shape nodes*
   Some datasets do not distinguish between *junctions* and *shape nodes*; hence, nodes on the road network are classified into these two types according to the topology of the road network. We also split long *shape arcs* (add *shape nodes*) such that the length of each *shape arc* is less than or equal to the predefined parameter ℓ_{max}. This is because that we may overlook *arcs* close to a certain point if the *arc* contains some long *shape arcs*. This situation occurs when both endpoints of a long *shape arc* are far from the point because we report an *arc* if at least one *shape node* of the *arc* is in a square centered on the point. The road network that completes this process is referred to as the *processed road network*.

   (b) Fractional cascading (Chapter 4)
   The first step of map matching is to obtain the *arcs* close to a vehicle *trajectory*, and FC data structure accelerates the process. We want to report *arcs* within r meter from a certain point, and an elementary geometry determines whether or not *arcs* are in the circle. However, this process is computationally expensive if the road network has vast *arcs*. We speed up this step by utilizing FC that reports every *arc* that has at least one *shape node* belonging to a square centered on a certain point. Based on the Theorem 2, if we set the side length of the square $c = 2 \cdot \max\left(r, \frac{\ell_{max}+2r}{2\sqrt{2}}\right)$ if $\ell_{max} \leq 2\left(1+\sqrt{2}\right) r$ (otherwise $c = \ell_{max}$), we obtain either of the endpoints of the *shape arc* within a radius r meter from a certain point. This implies we acquire all *arcs* within r meter from the point. Before performing map-matching, we construct FC data structure with all the *shape nodes* of *processed road network*; and the FC is utilized for the step "2. TEG-matching >(a) Obtain neighborhood arcs" as mentioned below.

   (c) Bottom-up segmentation (Chapter 3)
   Processed road network have some redundant *shape nodes* to represent the shape of the road. Bottom-up segmentation reduces these nodes and contributes to memory reduction and speedup in map matching.

2. TEG-matching

   (a) Obtain neighborhood *arcs*
   Given a vehicle *trajectory* $\boldsymbol{P} = (P_i)_{i=1}^{n+1}$, we first obtain *arcs*, where the vehicle may travel from time stamp i to $i+1$ $(i \in \{1,\ldots,n\})$. We assume that these *arcs* lie within $r' = d(P_i, P_{i+1})/2 + r_{\mathrm{GPS}}$ from the midpoint of P_i and P_{i+1} (hearinafter denoted by $P_{i,i+1}$), where r_{GPS} is the upper bound of the spatial measurement error, and $d(x,y)$ is the Euclidean distance between x and y. From the above discussion, if we set $\ell_{max} = 2(1+\sqrt{2})r_{\mathrm{GPS}} \left(\leq 2(1+\sqrt{2})r'\right)$ and $c_i = 2 \cdot \max\left(r', \frac{\ell_{max}+2r'}{2\sqrt{2}}\right)$, we can obtain all *arcs* that lie within r' from $P_{i,i+1}$. FC speed up the query.

   (b) Construct a time-expanded graph (TEG) and find the shortest path on TEG (Chapter 2)
   We construct a TEG that represents the space-time movement of the vehicle. To attain the most plausible *matched path*, we find the shortest path on the TEG and restore the *matched path* from the shortest path.

Chapter 2

Time-Expanded Graph (TEG)

Given a time series of GPS recording, our model first builds a time-expanded graph (TEG) and obtains a *matched path* by solving the shortest path problem on the TEG. This section starts with the topology construction of the TEG in Section 2.1 and then explains three types of weights on TEG in Section 2.2.

We first introduce some notation used in this section. Given $X, Y \subset \mathbb{R}^2$ or $X, Y \in \mathbb{R}^2$, $d(X, Y)$, and $d_\infty(X, Y)$ are the Euclidean and Chebyshev distances between X and Y, respectively. For example, the distance between a *shape arc* (v_1, v_2) and a position fix P_i is denoted by $d\left((v_1, v_2), P_i\right)$. $d\left((v_1, v_2), P_i\right)$ is the perpendicular distance if the perpendicular distance is achieved on the segment (v_1, v_2); otherwise, $\min\left\{d\left((v_1, P_i\right), d\left((v_2, P_i\right)\right\}$. Similarly, if $a = (v_1, \ldots, v_m)$ is an *arc*, $d(a, P_i) = \min_{1 \le j \le m-1} d\left((v_j, v_{j+1}), P_i\right)$. The first and last *node* (*junction*) of an *arc* a are denoted by $\mathrm{tail}(a) := v_1$ and $\mathrm{head}(a) := v_m$, respectively.

2.1 Topology construction of the TEG

Given a *road network* G that has *arc* set A, we define the line graph (Ray-Chaudhuri, 1967) $L(G)$ of G as the directed graph whose vertex set $V\left(L(G)\right) := A$ and whose directed edge set $E\left(L(G)\right) := \{(a, \tilde{a}) \mid a, \tilde{a} \in A, \mathrm{head}(a) = \mathrm{tail}(\tilde{a})\}$. Given G and a vehicle *trajectory* $\boldsymbol{P} = (P_i)_{i=1}^{n+1}$, the corresponding time-expanded graph (TEG), referred to as $T(G, \boldsymbol{P}) = \left(V\left(T(G, \boldsymbol{P})\right), E\left(T(G, \boldsymbol{P})\right)\right)$, consists of n subgraphs of $L(G)$ denoted by

$$L(G, P_i) = \left(V\left(L(G, P_i)\right), E\left(L(G, P_i)\right)\right) \;\; (1 \le i \le n) \tag{2.1}$$

. The i-th *layer graph* $L(G, P_i)$ represents a partial road network where the vehicle may travel from time stamp i to $i + 1$; hence $L(G, P_i)$ includes the area near P_i and P_{i+1}. Specifically, $V\left(L(G, P_i)\right)$ includes all *arcs* that lie within $r' = d(P_i, P_{i+1})/2 + r_{\mathrm{GPS}}$ from the $P_{i,i+1}$ where r_{GPS} is the upper bound of measurement error, and $P_{i,i+1}$ is the midpoint of P_i and P_{i+1}. Based on the previous discussion in Section 1.5, we define

$$V\left(L(G, P_i)\right) := \left\{a^i \mid a = (v_1, \ldots, v_m) \in A, \min_{1 \le j \le m} d_\infty(v_j, P_{i,i+1}) \le c_i/2\right\} \text{ and} \tag{2.2}$$

$$E\left(L(G, P_i)\right) := \left\{\left(a^i, \tilde{a}^i\right) \in V\left(L(G, P_i)\right) \times V\left(L(G, P_i)\right) \mid \mathrm{head}(a) = \mathrm{tail}(\tilde{a})\right\}, \tag{2.3}$$

where c_i is defined in Section 1.5. For any notation x, x^i is the copy of x related to the time stamp i, and the superscript i is used for distinguishing copies related to different time stamps. For instance, a^i exists in i-th *layer graph* and is the copy of the *arc* $a \in A$. $a^i \in V\left(L(G, P_i)\right)$ and $(a^i, \tilde{a}^i) \in E\left(L(G, P_i)\right)$ are referred to as a *layer vertex* and *layer edge*, respectively. $(a^i, \tilde{a}^i)$ implies that the vehicle moves from the *arc* a to the *arc* $\tilde{a}$ between the time stamps i and $i+1$. To express the travel of the vehicle from time stamp i to $i+2$, we define the set of *layer-to-layer edges* $E_{\mathrm{LtL}}\left(L(G, P_i, P_{i+1})\right)$ as

$$E_{\mathrm{LtL}}\left(L(G, P_i, P_{i+1})\right) := \left\{(a^i, a^{i+1}) \in V\left(L(G, P_i)\right) \times V\left(L(G, P_{i+1})\right) \mid d(a, P_{i+1}) \leq r_{\mathrm{GPS}}\right\} \tag{2.4}$$

for $i \in \{1, 2, \ldots, n-1\}$. (a^i, a^{i+1}) implies that the vehicle lies on *arc* a at the time stamp $i+1$. We also define *source* s and *sink* t so that we formulate the map-matching problem as a shortest path problem from s to t. The set of *source edges* $E_{\mathrm{source}}\left(L(G, \boldsymbol{P})\right)$ and the set of *sink edges* $E_{\mathrm{sink}}\left(L(G, \boldsymbol{P})\right)$ are defined as follows:

$$E_{\mathrm{source}}\left(L(G, \boldsymbol{P})\right) := \left\{(s, a^1) \mid a^1 \in V\left(L(G, P_1)\right)\right\} \tag{2.5}$$

$$E_{\mathrm{sink}}\left(L(G, \boldsymbol{P})\right) := \left\{(a^n, t) \mid a^n \in V\left(L(G, P_n)\right)\right\}. \tag{2.6}$$

In conclusion, the TEG $T(G, \boldsymbol{P}) = \left(V\left(T(G, \boldsymbol{P})\right), E\left(T(G, \boldsymbol{P})\right)\right)$ is defined as follows:

$$V\left(T(G, \boldsymbol{P})\right) := \left(\bigcup_{i=1}^{n} V\left(L(G, P_i)\right)\right) \cup \{s, t\} \tag{2.7}$$

$$E\left(T(G, \boldsymbol{P})\right) := \left(\bigcup_{i=1}^{n} E\left(L(G, P_i)\right)\right) \cup \left(\bigcup_{i=1}^{n-1} E_{\mathrm{LtL}}\left(L(G, P_i)\right)\right) \tag{2.8}$$

$$\cup\, E_{\mathrm{source}}\left(L(G, \boldsymbol{P})\right) \cup E_{\mathrm{sink}}\left(L(G, \boldsymbol{P})\right) \tag{2.9}$$

A path from s to t represents the vehicle's travel from time stamp 1 to $n+1$, and the next subsection assigns weight to $T(G, \boldsymbol{P})$ to find the most reasonable space-time path. The following theorem insists that TEG has the path corresponding to the *correct path* under a certain condition.

Theorem 1. *Let G be a road network that has* arc *set A, $\boldsymbol{P} = (P_i)_{i=1}^{n+1}$ be a vehicle trajectory. If the vehicle lies within $r' = d(P_i, P_{i+1})/2 + r_{GPS}$ from the $P_{i,i+1}$ between the time stamp i and $i+1$ $(1 \leq \forall i \leq n)$, the TEG has the* path *from* source *s to* sink *t that corresponds to the* correct path.

Proof. Let $\left(a_1^i, a_2^i, \ldots, a_{L_i}^i\right)$ be the *correct path* (the sequence of *arcs* that the vehicle actually travels) between the time stamp i and $i+1$. Then, the equation $a_{L_i}^i = a_1^{i+1}$ $(1 \leq i \leq n-1)$ satisfies. As $V\left(L(G, P_i)\right)$ includes all *arcs* that lie within $r' = d(P_i, P_{i+1})/2 + r_{\mathrm{GPS}}$ from the $P_{i,i+1}$, especially $V\left(L(G, P_i)\right)$ includes all the *arcs* that lie within r_{GPS} from P_i and P_{i+1}. At the time step 1, as the vehicle lies within r_{GPS} from P_1, $a_1^1 \in V\left(L(G, P_1)\right)$, which implies $\left(s, a_1^1\right) \in E_{\mathrm{source}}\left(L(G, \boldsymbol{P})\right)$. For the same reason, $\left(a_{L_n}^n, t\right) \in E_{\mathrm{sink}}\left(L(G, \boldsymbol{P})\right)$. We also know $a_k^i \in V\left(L(G, P_i)\right)$ $(1 \leq \forall i \leq n, 1 \leq \forall k \leq L_i)$ from the assumption. From the characteristic of the *(correct) path*, head$\left(a_k^i\right) =$ tail$\left(a_{k+1}^i\right)$ $(1 \leq \forall i \leq n, 1 \leq \forall k \leq L_i - 1)$ holds. Hence $\left(a_k^i, a_{k+1}^i\right) \in E\left(L(G, P_i)\right)$. If we are concern about $a_{L_i}^i \in V\left(L(G, P_i)\right)$, $a_1^{i+1} \in V\left(L(G, P_{i+1})\right)$, $a_{L_i}^i = a_1^{i+1}$ and $d\left(a_1^{i+1}, P_{i+1}\right) \leq r_{\mathrm{GPS}}$, we have $\left(a_{L_i}^i, a_1^{i+1}\right) \in E_{\mathrm{LtL}}\left(L(G, P_i)\right)$ for $1 \leq \forall i \leq n-1$. In summary,

$$\left\{\left(s, a_1^1\right), \left(a_1^1, a_2^1\right), \ldots, \left(a_{L_i-1}^1, a_{L_i}^1\right), \left(a_{L_i}^1, a_1^2\right), \left(a_1^2, a_2^2\right), \ldots, \left(a_{L_n-1}^n, a_{L_n}^n\right), \left(a_{L_n}^n, t\right)\right\} \tag{2.10}$$

is the *path* from source s to sink t on TEG that corresponds to the *correct path*. $\square$

2.2 Weight of the TEG

The path from *source* s to sink t on the TEG represents the space-time travel of the vehicle, and we formulate map matching as a shortest path problem from s to t. To quantify the improbability of the path, we consider three points: (1) area between the path and the vehicle trajectory, (2) abrupt direction changes, and (3) spatial measurement error. The corresponding weight functions are *area weight* $w_{\text{area}}(\cdot)$, *direction change weight* $w_d(\cdot)$, and *spatial weight* $w_s(\cdot)$, respectively. The following subsections explain the details of these weights and their motivations using mathematical expressions. The last subsection integrates these three weights into the weight of TEG.

We introduce some notations and definitions, which are illustrated in Figure 2.1. Let $p, p', v_j \in \mathbb{R}^2$ $(j \in \{1, 2, \dots\})$ be a two-dimensional point coordinate. For a polyline $a = (v_1, \dots, v_m)$, the $l(a) := \sum_{j=1}^{m-1} d(v_j, v_{j+1})$ denotes the length of the polyline a. We define $\tilde{d}\left(p, (v_1, v_2)\right)$ as the perpendicular distance between p and the segment (v_1, v_2), that is, $d\left(p, (v_1, v_2)\right)$ is the Euclidean distance between p and the line containing the two points v_1 and v_2. The point on the "line" (not segment) that achieves the perpendicular distance is called the *perpendicular point* and is denoted by $\zeta_{p,(v_1,v_2)}$. If $\zeta_{p,(v_1,v_2)}$ lies on (v_1, v_2), we say that "p achieves perpendicular distance on (v_1, v_2)". The *projection* of p onto (v_1, v_2) is denoted by $\eta_{p,(v_1,v_2)}$. If p achieves a perpendicular distance on (v_1, v_2), $\zeta_{p,(v_1,v_2)} = \eta_{p,(v_1,v_2)}$ satisfies; otherwise, $\eta_{p,(v_1,v_2)} = \underset{v \in \{v_1, v_2\}}{\operatorname{argmin}}\ d(p, v)$. Similarly, we say that "p achieves perpendicular distance on a polyline $a = (v_1, \dots, v_m)$" if at least one segment (v_j, v_{j+1}) $(1 \le j \le m-1)$ achieves the perpendicular distance on (v_j, v_{j+1}). If p achieves perpendicular distance on a, the *perpendicular point* $\zeta_{p,a}$ is defined as the nearest *perpendicular point* for (v_j, v_{j+1}) $(1 \le j \le m-1)$, that is, $\zeta_{p,a} := \zeta_{p,(v_j,v_{j+1})}$ such that p achieves perpendicular distance on (v_j, v_{j+1}) and $\tilde{d}\left(p, (v_j, v_{j+1})\right) \le \tilde{d}\left(p, (v_{j'}, v_{j'+1})\right)$ $\forall j' \in \{j' \mid p \text{ achieves perpendicular distance on } (v_{j'}, v_{j'+1})\}$. Otherwise, $\zeta_{p,a} := \text{nil}$, where "nil" implies that the position fix p does not have the *perpendicular point* on the polyline a. The tail and head of the segment on which the *perpendicular point* $\zeta_{p,a}$ lies on are denoted by $v_{j(\text{tail},p,a)}$ and $v_{j(\text{head},p,a)}$, respectively. Specifically, we define $v_{j(\text{tail},p,a)} := v_j$ and $v_{j(\text{head},p,a)} := v_{j+1}$ if $\zeta_{p,a} = \zeta_{p,(v_j,v_{j+1})}$; otherwise, if $\zeta_{p,a} = \text{nil}$, $v_{j(\text{tail},p,a)} = v_{j(\text{head},p,a)} = \text{nil}$. The *perpendicular distance* between p and a polyline a is denoted by $\tilde{d}(p, a)$ and is defined as

$$\tilde{d}(p, a) := \min\left\{\tilde{d}\left(p, (v_j, v_{j+1})\right) \mid 1 \le j \le m-1, p \text{ achieves } \textit{perpendicular distance} \text{ on } (v_j, v_{j+1})\right\} \tag{2.11}$$

if p achieves *perpendicular distance* on a. The *projection* of p onto a polyline a is denoted by

$$\eta_{p,a} := \underset{\eta_{p,(v_j,v_{j+1})}}{\operatorname{argmin}}\ d\left(p, \eta_{p,(v_j,v_{j+1})}\right) \tag{2.12}$$

subject to $1 \le j \le m-1$. Given two point coordinates $p, p' \in \mathbb{R}^2$, we say that the polyline $a = (v_1, \dots, v_m)$ "lies on p' side" if $\zeta_{v_j,(p,p')} \in \{p' + (p' - p)\theta \mid \theta \ge 0\}$

for all $j \in \{1,\ldots,m\}$. The closest *perpendicular point* is called the *nearest perpendicular point* and denoted by $\zeta^*_{a,(p,p')} := \underset{\zeta_{v_j,(p,p')}}{\operatorname{argmin}}\ d\left(\zeta_{v_j,(p,p')}, (p,p')\right)$ subject to $j \in \{1,\ldots,m\}$, and the corresponding point on the polyline is denoted by $v^*_{a,(p,p')} := \underset{v_j}{\operatorname{argmin}}\ d\left(\zeta_{v_j,(p,p')}, (p,p')\right)$ subject to $j \in \{1,\ldots,m\}$. Figure 2.1 visualizes these notations.

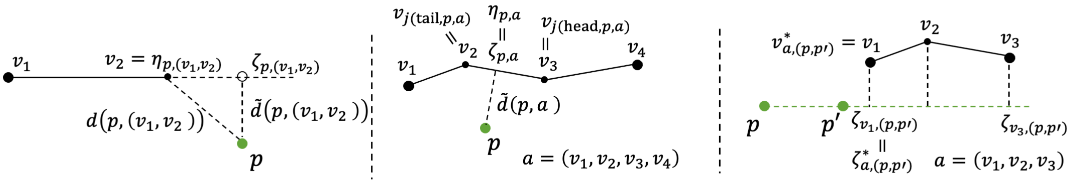

FIGURE 2.1: (Left and center) Visualization of a distance $d(\cdot)$, perpendicular distance $\tilde{d}(\cdot)$, a *projection* η, and a *perpendicular point* ζ. (Right) Example of a polyline that "lies on the p' side."

2.2.1 Area weight

If the area between a *vehicle trajectory* and the *matched path* is sufficiently small, the *matched path* is probably the same as the *correct path*. We attempt to obtain the path with a small area, but lining up probable candidate paths is difficult. To address the issue, the *area weight* of an *arc* is defined as the area between the *arc* and a *vehicle trajectory*; then, the area between a path and a *vehicle trajectory* is defined as the total *area weight* overall arcs contained in the path. The *area weight* facilitates the acquisition of the path that has a small area with a *vehicle trajectory*. This section defines the *area weight*, and the assignment is explained in Section 2.2.4. Figure 2.2 illustrates the *area weights* as the sum of the green and yellow areas. We calculate the *area weights* in various ways according to the relative positions of the *arc* and position fixes, and the color corresponds to these situations. We categorize the relative positions and explain how to calculate the *area weight* for each case.

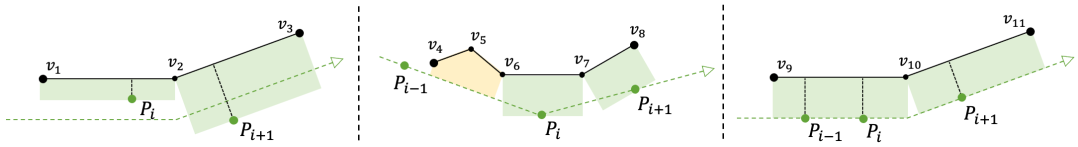

FIGURE 2.2: Examples of *area weight* $w_{\text{area}}(\cdot)$. *Area weight* is represented as the sum of the areas of green and yellow.

The *area weight* $w_{\text{area}}(\cdot)$ is defined using $w_{\text{area0}}(\cdot)$, $w_{\text{area1}}(\cdot)$, and $w_{\text{area2}}(\cdot)$, which are illustrated in Figure 2.3; hence, we first provide these three definitions. Let $p, p', p_i, v_j \in \mathbb{R}^2\ (i, j \in \{1, 2, \ldots\})$ be a two-dimensional point coordinate. Suppose that (p, p') and $(p_1, \ldots, p_n)$ are subsequences of a *vehicle trajectory*, and $a = (v_1, \ldots, v_m)$ is a subsequence of an *arc*. Then, $w_{\text{area0}}\left((p_1, \ldots, p_n), a\right)$ defines the area between $(p_1, \ldots, p_n)$ and a under the assumption that the $\{p_i\}_{i=1}^n$ are obtained when the vehicle travels on a. Formally, if every p_i $(1 \leq i \leq n)$ achieves *perpendicular distance* on

a, $w_{\text{area0}}\left((p_1,\ldots,p_n),a\right)$ is defined as follows:

$$w_{\text{area0}}\left((p_1,\ldots,p_n),a\right) := \left(\frac{1}{n}\sum_{i=1}^{n}\tilde{d}(p_i,a)\right) l(a) = \left(\frac{1}{n}\sum_{i=1}^{n}\tilde{d}(p_i,a)\right)\left(\sum_{j=1}^{m-1} d(v_j,v_{j+1})\right) \tag{2.13}$$

$w_{\text{area1}}(p,p',a)$ defines the area between a and the segment (p,p') under the assumption that the position fix p (p') is acquired before (after) the vehicle travels on a. More precisely, $w_{\text{area1}}(p,p',a)$ is normally the area between a and the line containing p and p'. We define $w_{\text{area1}}(p,p',a)$ as the sum of the areas between the *shape arc* (v_j,v_{j+1}) $(j=1,\ldots,m-1)$ and the line containing p and p'; that is, $w_{\text{area1}}(p,p',a) := \sum_{j=1}^{m-1} w_{\text{area1}}(p,p',v_j,v_{j+1})$, where $w_{\text{area1}}(p,p',v,v')$ is defined as follows. $w_{\text{area1}}(p,p',v,v')$ is the improbability estimation of the vehicle passing through the *shape arc* (v,v') during the interval of p and p'. $w_{\text{area1}}(p,p',v,v')$ becomes large if the angle difference between (v,v') and (p,p') is large, or if the (v,v') is far from the (p,p'). Normally, $w_{\text{area1}}(p,p',v,v')$ becomes the area between (v,v') and the line containing p and p'. However, the exact definition depends on the two spatial conditions: (1) whether the line containing (p,p') separates the (v,v'); and (2) whether the angle between the (p,p') and (v,v') is less than $\pi/2$. The exact definition and the corresponding figures are summarized in Table 2.1 and Figure 2.4, respectively.

$w_{\text{area2}}(p,p',a)$ denotes the penalty that the vehicle passes through a far from the two consecutive position fixes p and p' during the interval of p and p'. Formally, $w_{\text{area2}}(p,p',a)$ is defined as follows:

$$w_{\text{area2}}(p,p',a) := \begin{cases} d\left(\zeta^*_{a,(p,p')},(p,p')\right) d\left(v^*_{a,(p,p')},\zeta^*_{a,(p,p')}\right) & \text{if } a \text{ lies on } p \text{ or } p' \text{ side} \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

, as shown in the right side on 2.3.

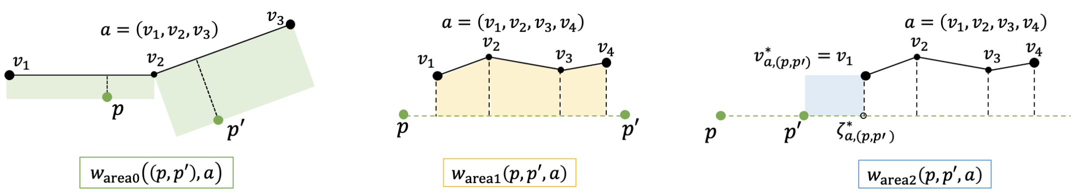

FIGURE 2.3: Visualization of $w_{\text{area0}}\left((p,p'),a\right)$, $w_{\text{area1}}(p,p',a)$, and $w_{\text{area2}}(p,p',a)$.

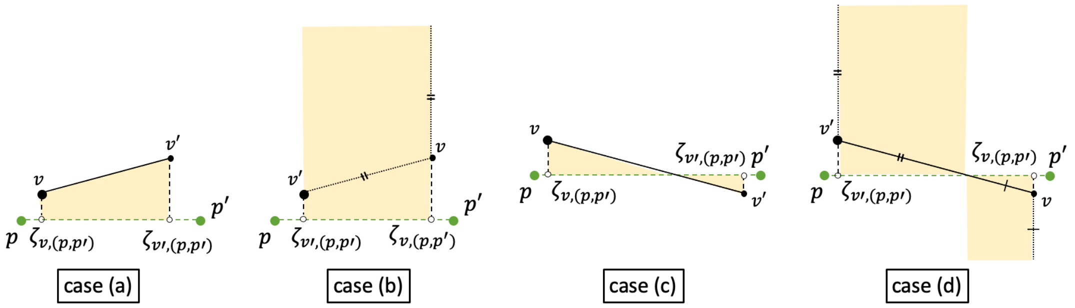

FIGURE 2.4: Visualization of $w_{\text{area1}}(p, p', v, v')$.

case	(1)	(2)	$w_{\text{area1}}(p, p', v, v')$
(a)		✓	$\left(d\left(v, \zeta_{v,(p,p')}\right) + d\left(v', \zeta_{v',(p,p')}\right)\right) d\left(\zeta_{v,(p,p')}, \zeta_{v',(p,p')}\right)/2$
(b)			$d\left(\zeta_{v',(p,p')}, \zeta_{v,(p,p')}\right)\left(d\left(v, \zeta_{v,(p,p')}\right) + d(v', v)\right)$
(c)	✓	✓	$\frac{\left(d\left(v,\zeta_{v,(p,p')}\right)^2 + d\left(v',\zeta_{v',(p,p')}\right)^2\right) d\left(\zeta_{v',(p,p')},\zeta_{v,(p,p')}\right)}{2\left(d\left(v,\zeta_{v,(p,p')}\right) + d\left(v',\zeta_{v',(p,p')}\right)\right)}$
(d)	✓		$\frac{\left(d\left(v,\zeta_{v,(p,p')}\right) + d\left(v',\zeta_{v',(p,p')}\right) + d(v',v)\right)\left(d\left(v,\zeta_{v,(p,p')}\right)^2 + d\left(v',\zeta_{v',(p,p')}\right)^2\right) d\left(\zeta_{v',(p,p')},\zeta_{v,(p,p')}\right)}{\left(d\left(v,\zeta_{v,(p,p')}\right) + d\left(v',\zeta_{v',(p,p')}\right)\right)^2}$

TABLE 2.1: Definition of $w_{\text{area1}}(p, p', v, v')$. (1) implies whether the line containing (p, p') separates the (v, v'); and (2) represents whether the angle between the (p, p') and (v, v') is less than $\pi/2$.

Consider the *area weight* $w_{\text{area}}(a^i)$ for the *layer vertex* $a^i \in V(L(G, P_i))$, where the vehicle may travel from the time step i to $i+1$. If we denote the *arc* by $a = (v_1, \ldots, v_m)$, $w_{\text{area}}(a^i)$ indicates the improbability that the vehicle travels on the *arc* a from the time step i to $i+1$. The improbability is basically expressed as the area between the arc a and the vehicle trajectory. However, the exact definition depends on the relative positions of the *arc* a, position fixes P_{i-1}, P_i, and P_{i+1}. Figure 2.5 illustrates the $w_{\text{area}}(a^i)$, and Table 2.2 summarizes the definition of $w_{\text{area}}(a^i)$ by cases, and the detailed explanation is given below:

1. (case1) Both P_i and P_{i+1} achieve *perpendicular distance* on a. ($\zeta_{P_i,a} \neq \text{nil}$, and $\zeta_{P_{i+1},a} \neq \text{nil}$)
   We can suppose that the vehicle trajectory is $\left(\tilde{d}(P_i, a) + \tilde{d}(P_{i+1}, a)\right)/2$ away from the *arc* a on average; hence, we define $w_{\text{area}}(a^i) := w_{\text{area0}}\left((P_i, P_{i+1}), a\right)$.

2. (case2) P_i achieves perpendicular distance on a, but P_{i+1} does not. ($\zeta_{P_i,a} \neq \text{nil}$, and $\zeta_{P_{i+1},a} = \text{nil}$)
   When a vehicle passes through the *shape arc* $(v_{j(\text{tail},P_i,a)}, v_{j(\text{head},P_i,a)})$, the vehicle trajectory is likely $\tilde{d}(P_i, a)$ away from the *shape arc*. Hence, we add

$$w_{\text{area0}}\left((P_i), (v_{j(\text{tail},P_i,a)}, v_{j(\text{head},P_i,a)})\right)$$

   to $w_{\text{area}}(a^i)$. Because P_{i+1} does not achieve a perpendicular distance on a, we can suppose that the vehicle passes through the polyline

$$(v_{j(\text{head},P_i,a)}, v_{j(\text{head},P_i,a)+1}, \ldots, v_m)$$

during the interval of P_i and P_{i+1}. This implies that we add

$$w_{\text{area1}}\left(P_i, P_{i+1}, (v_{j(\text{head},P_i,a)}, v_{j(\text{head},P_i,a)+1}, \ldots, v_m)\right)$$

to $w_{\text{area}}(a^i)$. For the polyline $\left(v_1, v_2, \ldots, v_{j(\text{tail},P_i,a)}\right)$, we consider two cases. If P_{i-1} achieves a perpendicular distance on a, we can suppose that the vehicle trajectory $\left(\tilde{d}(P_{i-1}, a) + \tilde{d}(P_i, a)\right)/2$ away from the polyline on average. Hence, we add

$$w_{\text{area0}}\left((P_{i-1}, P_i), \left(v_1, \ldots, v_{j(\text{tail},P_i,a)}\right)\right)$$

to $w_{\text{area}}(a^i)$. In the other case, that is, if P_{i-1} does not achieve a perpendicular distance on a, we suppose that the P_{i-1} is obtained before the vehicle passes through the a. Hence, we add

$$w_{\text{area1}}\left(P_{i-1}, P_i, (v_1, \ldots, v_{j(\text{tail},P_i,a)})\right)$$

to $w_{\text{area}}(a^i)$.

3. (case3) P_i does not achieve perpendicular distance on a.
Because $w_{\text{area}}(a^i)$ represents the improbability of the vehicle passing through the arc a from the time stamp i to $i+1$, we suppose that P_i is acquired before the vehicle travels on *arc a*. Hence, we add $w_{\text{area1}}(P_i, P_{i+1}, a)$ to $w_{\text{area}}(a^i)$. Moreover, we assign the penalty $w_{\text{area2}}(P_i, P_{i+1}, a)$ to $w_{\text{area}}(a^i)$ to address the situation in which the *arc a* is far from the segment (P_i, P_{i+1}).

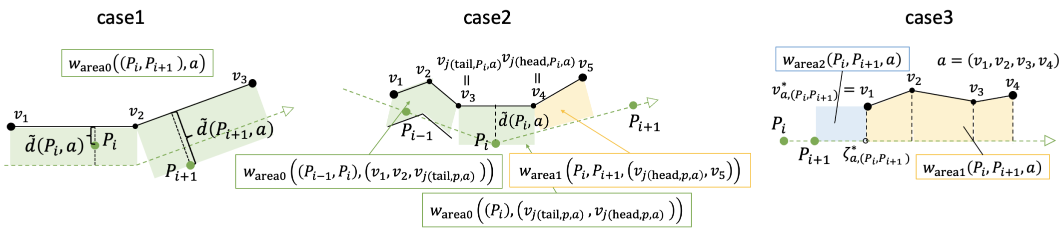

FIGURE 2.5: Visualization of $w_{\text{area}}(a^i)$ by cases.

(a)	(b)	(c)	case	$w_{\text{area}}(a^i)$
✓	✓	✓	case1	$w_{\text{area0}}\left(\left(P_i, P_{i+1}\right), a\right)$
✓	✓			
✓		✓	case2	$w_{\text{area0}}\left(\left(P_{i-1}, P_i\right), \left(v_1, \ldots, v_{j(\text{tail},P_i,a)}\right)\right)$ $+w_{\text{area0}}\left(\left(P_i\right), \left(v_{j(\text{tail},P_i,a)}, v_{j(\text{head},P_i,a)}\right)\right)$ $+w_{\text{area1}}\left(P_i, P_{i+1}, \left(v_{j(\text{head},P_i,a)}, v_{j(\text{head},P_i,a)+1}, \ldots, v_m\right)\right)$
✓			case2	$w_{\text{area1}}\left(P_{i-1}, P_i, \left(v_1, \ldots, v_{j(\text{tail},P_i,a)}\right)\right)$ $+w_{\text{area0}}\left(\left(P_i\right), \left(v_{j(\text{tail},P_i,a)}, v_{j(\text{head},P_i,a)}\right)\right)$ $+w_{\text{area1}}\left(P_i, P_{i+1}, \left(v_{j(\text{head},P_i,a)}, v_{j(\text{head},P_i,a)+1}, \ldots, v_m\right)\right)$
	✓	✓	case3	$w_{\text{area1}}\left(P_i, P_{i+1}, a\right) + w_{\text{area2}}\left(P_i, P_{i+1}, a\right)$
	✓			
		✓		

TABLE 2.2: Definition of $w_{\text{area}}(a^i)$. (a)$\zeta_{P_i,a} \neq$ nil, (b) $\zeta_{P_{i+1},a} \neq$ nil, and (c) $\zeta_{P_{i-1},a} \neq$ nil.

2.2.2 Direction change weight

We assume that the position fix is obtained every time the vehicle makes a significant direction change, and the *direction change weight* $w_d(\cdot)$ reflects this assumption. $w_d(\cdot)$ becomes large if there is no position fix near a significant direction change when the vehicle moves from one *arc* to another. For an *layer edge* $(a_1^i, a_2^i) \in A^i$, the $w_d\left((a_1^i, a_2^i)\right)$ is defined as the square of the distance between the common *node* of these two arcs (a_1 and a_2) and the segment whose endpoints are η_{P_i,a_1} and η_{P_i,a_2}.

$$w_d\left(\left(a_1^i, a_2^i\right)\right) := \left\{d\left(\text{head}\left(a_1\right), \left(\eta_{P_i,a_1}, \eta_{P_i,a_2}\right)\right)\right\}^2 \tag{2.15}$$

Figure 2.6 shows two cases where the position fix is far and near from a large direction change at the intersection of two *arcs*. We assume that the vehicle travels $(\ldots, v_1, v_2, v_3, v_4, v_5, \ldots)$, and P_i is acquired when the vehicle passes through v_3.

2.2.3 Spatial weight

Spatial weight $w_s(\cdot)$ is the penalty when the vehicle lies on the *arc* far from the position fix P_{i+1} at the time stamp $i+1$. Because *layer-to-layer edge* (a^i, a^{i+1}) indicates that the vehicle lies on the *arc* a at the time step $i+1$, $w_s\left((a^i, a^{i+1})\right)$ is defined as the square

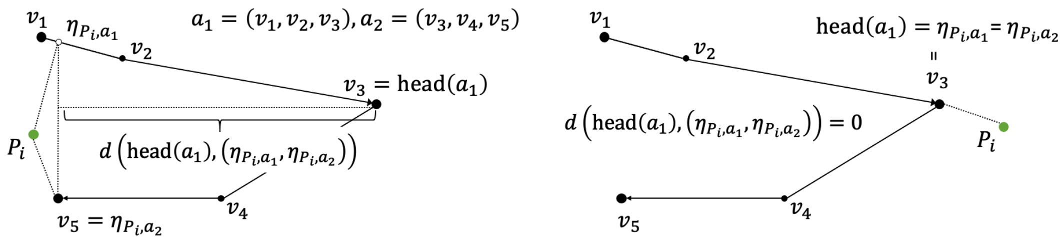

FIGURE 2.6: Visualization of $w_d\left((a_1^i, a_2^i)\right)$. The left (right) side is the case where the position fix is far (near) from a significant direction change at the intersection of two *arcs*.

of the distance between the position fix P_{i+1} and arc a:

$$w_s\left((a^i, a^{i+1})\right) := \{d(P_{i+1}, a)\}^2 \tag{2.16}$$

We define similar but large weights for the first and last position fix (P_1 and P_{n+1}) to prevent the *matched path* from being shorter than the *correct path*. The shorter path is caused by the property of the shortest path problem. Formally, for a *source arc* (s, a^1) and a *sink arc* (a^n, t), we define

$$w_s\left((s, a^1)\right) := d(P_1, a) l_{\text{mean}} \tag{2.17}$$

$$w_s\left((a^n, t)\right) := d(P_{n+1}, a) l_{\text{mean}} \tag{2.18}$$

, where l_{mean} is the average *arcs* length.

2.2.4 Weight of the edge in TEG

This section integrates *area*, *direction change*, and *spatial weights* into the weight of the TEG. Let G be a *road network*, $\boldsymbol{P} = (P_i)_{i=1}^{n+1}$ be the vehicle *trajectory*, $T(G, \boldsymbol{P}) = (V(T(G, \boldsymbol{P})), E(T(G, \boldsymbol{P})))$ be the corresponding TEG, where $V(T(G, \boldsymbol{P}))$ is the vertex set and $E(T(G, \boldsymbol{P}))$ is the edge set. We assign a weight to each edge of the TEG by transferring the vertex weight to the edge weight. Formally, for each edge $e = (e_{\text{tail}}, e_{\text{head}}) \in E(T(G, \boldsymbol{P}))$, *edge weight* is defined as follows:

$$w(e) := \begin{cases} w_{\text{area}}(e_{\text{tail}}) + w_d(e) & (\text{if } e \text{ is a } \textit{layer edge}) \\ w_s(e) & (\text{if } e \text{ is a } \textit{source edge} \text{ or } \textit{layer-to-layer edge}) \\ w_s(e) + w_{\text{area}}(e_{\text{tail}}) & (\text{if } e \text{ is a } \textit{sink edge}) \end{cases} \tag{2.19}$$

Our model solves the shortest path problem from *source* to *sink* on the TEG, thus obtaining the *matched path*.

Chapter 3

Bottom-up Segmentation

A segmentation method is applied to both (1) a *road network* to reduce the graph size and (2) vehicle *trajectory* to generate the evaluation data (low-sampling-rate trajectories). Keogh et al. (2002) undertook an extensive review and empirical comparison of several piecewise linear approximation techniques for the time series databases and proposed a new algorithm. Although such an approach does not initially develop for a directed graph, the approach is transferred to graph simplification while maintaining the topology of the graph. This section introduces three basic segmentation techniques, and we apply the most suitable one to the *road network* and the vehicle *trajectory*. We further visualized the *arcs* applied and did not apply the segmentation technique using different hyper-parameters.

Given a time series data $T_1, T_2, \ldots, T_n$, each of which includes only coordinates (spatial data), these algorithms aim to create a polyline similar to the time series data using a smaller number of line segments. According to Keogh et al. (2002), we essentially categorize time-series segmentation algorithms into three groups: (1) sliding windows (Koski, Juhola, and Meriste, 1995; Park, Kim, and Chu, 2001), (2) top-down (DOUGLAS and PEUCKER, 1973; Park, Lee, and Chu, 1999), and (3) bottom-up (Keogh and Pazzani, 1998; Luebke, 2001). The sliding window algorithm is a simple and intuitive online algorithm. It works by anchoring the first data point as the tail of a potential segment and then approximates the data with increasingly longer segments. At some point i, the potential segment's error is greater than the user-specified threshold; thus, the subsequence from the anchor to $i - 1$ is transformed into a segment. The top-down algorithm works by considering every possible partitioning of the time series and splitting it at the best location. We then tested whether the approximation error of each subsection was below a user-specified threshold. A failed subsection was recursively split until all the segments had approximation errors below the threshold. The bottom-up segmentation is a natural complement to the top-down algorithm. The algorithm begins by creating the finest possible approximation of the time series. The merge cost of two adjacent segments is calculated, and the lowest cost pair is iteratively merged until a stopping criterion is met. Keogh et al. (2002) concluded that the sliding window algorithm shows a generally poor quality, and the bottom-up algorithm often significantly outperforms the other two algorithms. The properties of the three algorithms are summarized in Table 3.1.

Because we apply a segmentation method to a *road network* and *vehicle trajectories* offline, bottom-up segmentation is the most suitable approach. We introduce some notation to describe the procedure of bottom-up segmentation. Let $T = (t_1, t_2, \ldots)$ be a finite sequence of points, where t_i is a coordinate. $T[a : b] := (t_a, t_{a+1}, \ldots, t_b)$ denotes the contiguous subsequence of T from a-th point to b-th point. A piecewise linear approximation of T is the output of bottom-up segmentation and is denoted by Seg_TS. Seg_TS is defined as the sequence of *approximate segments*, each of which is a two-element subsequence of T. An *approximate segment* "Seg" (=

TABLE 3.1: Feature summary for the three widely-known algorithms.

Algorithm	User can specify[1]	Online
Top-Down	E,ME,K	No
Bottom-Up	E,ME,K	No
Sliding Window	E	Yes

[1] E and ME are the maximum errors for a given segment and for an entire time series, respectively, where K represents the number of segments.

$(t_{\text{tail}}, t_{\text{head}})$) approximates a contiguous subsequence of T, and $\text{tail}(\text{Seg}) := t_{\text{tail}}$ and $\text{head}(\text{Seg}) := t_{\text{head}}$ denote the endpoints of the "Seg." The *approximation error* of "Seg," denoted by calculate_error(Seg), is the maximum distance between the "Seg" and one of the approximated contiguous subsequence points. merge(Seg, Seg') := $(\text{tail}(\text{Seg}), \text{head}(\text{Seg}'))$ is the rough *approximate segment* integrating two *approximate segments* Seg and Seg'. Hence, the corresponding two approximated contiguous subsequences are also merged.

Using these symbols, we show the pseudocode in Algorithm 1, and the right side of Figure 3.1 shows how the algorithm works. The *max_error* is the parameter that determines the approximation accuracy, and the influence is visualized on the left side in Figure 3.1. The figure indicates that too large a *max_error* destroys the road shape; thus, we have to choose an appropriate *max_error* based on the complexity and density of a *road network*.

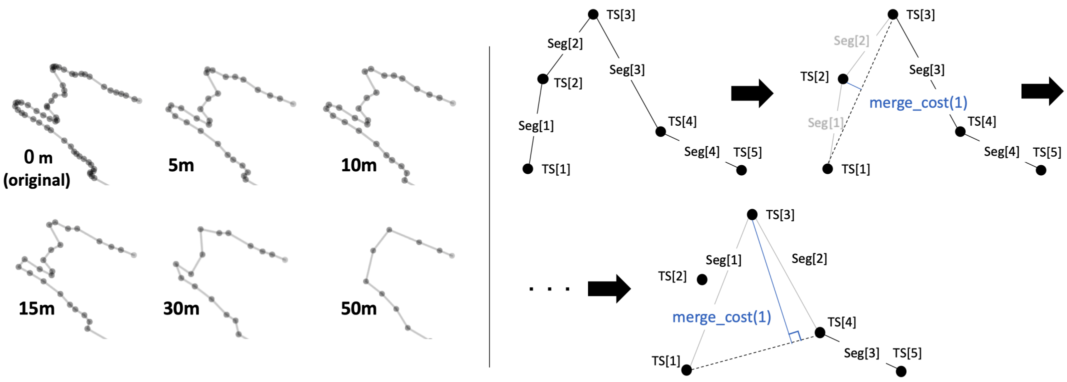

FIGURE 3.1: (Left) Influence of max_error on the bottom-up segmentation. The max_error is written at the lower left of each drawing. (Right) Procedure for the bottom-up segmentation.

Algorithm 1: *bottom-up segmentation*$(T, \text{max\_error})$

Input : a sequence of points $T = (t_1, t_2 \ldots, t_n)$, $n \geq 3$; max_error that decides the approximation accuracy

Output: A piecewise linear approximation of T, denoted by Seg_TS

```
Function calculate_merge_cost(Seg1, Seg2):
    merge_seg ← merge(Seg1, Seg2)
    merge_cost ← calculate_error(merge_seg)
    return merge_cost

Function bottom_up_segmentation(T,max_error):
    // Initialization
    Seg_TS ← [(t_1, t_2), (t_2, t_3), ..., (t_{n-1}, t_n)]
    for i ← 1 to n − 2 do
        merge_costs[i] ← calculate_merge_cost(Seg_TS[i], Seg_TS[i + 1])
    while min(merge_costs) < max_error and len(Seg_TS) > 1 do
        // Find the minimum pair to merge
        i ← arg min_i(merge_costs[i])
        Seg_TS[i] ← merge(Seg_TS[i], Seg_TS[i + 1])
        delete(Seg_TS[i + 1])
        delete(merge_costs[i])
        if 0 < i − 1 then
            merge_costs[i-1] ← calculate_merge_cost(Seg_TS[i − 1],
             Seg_TS[i]))
        if i + 1 <len(Seg_TS)+1 then
            merge_costs[i] ← calculate_merge_cost(Seg_TS[i],
             Seg_TS[i + 1])
    return Seg_TS
```

Chapter 4

Fractional Cascading

All map-matching algorithms first restrict candidate nodes and arcs near positioning data. Because this operation is often repeated for each position fix, it can be a computational bottleneck. To speed up this operation, we introduced the fractional cascading (FC), proposed by Chazelle and Guibas (1986).

FC is a data structure for an orthogonal range query with a query time of $\mathcal{O}(\log n + k)$ in the 2-dimensional space, where n is the total number of points in the data structure, and k is the number of the points lying in the orthogonal range. The range tree, an existing data structure, has a query time $\mathcal{O}\left((\log n)^2 + k\right)$, and the k-d tree (Bentley, 1975) has a query time $\mathcal{O}\left(n^{1/2} + k\right)$. Thus the FC is found to be an efficient algorithm, although it requires significant memory (Table 4.1).

	Query time	Space complexity
Fractional Cascading	$\mathcal{O}(\log n + k)$	$\mathcal{O}(n \log n)$
range-tree	$\mathcal{O}\left((\log n)^2 + k\right)$	$\mathcal{O}(n \log n)$
kd-tree	$\mathcal{O}\left(n^{1/2} + k\right)$	$\mathcal{O}(n)$

TABLE 4.1: Complexities of each data structure for 2-dimensional data

For 1-dimensional range queries, we commonly utilize the simple binary search tree. The set of points is split into two subsets of approximately equal sizes: one subset contains points smaller than or equal to the splitting value, while the other subset contains the points larger than the splitting value. The splitting value is stored at the root, and the two subsets are stored recursively in the two subtrees. This structure can be expanded to higher-dimensional range queries using FC.

The construction of the FC consists of two stages: *creating a binary tree* and *setting minmax and maxmin pointers*. *Creating a binary tree* is remarkably similar to the aforementioned one-dimensional case. The set of points is recursively split into two subsets of roughly equal size according to x-value. The only difference is that every vertex v of an FC tree contains not only the split point v_{split} but also a sorted list v_{list}, a sub-list of points in lexicographical order for (y, x). The pseudocode is described as "create_fctree" in Algorithm 2, where v_{left} and v_{right} represent the left and right children of vertex v, respectively. An example of an FC tree is illustrated in Figure 4.1 along with the original data points. At the root vertex v_0, the lexicographical order for (x, y) is C, A, E, F, D, G, B; hence, F is selected as *split point*. All the points smaller than or equal to F (i.e., C, A, E, F) are held by the left child v_1 and sorted in lexicographical order for (y, x). The *minmax (maxmin) pointers* facilitate specifying points smaller (larger) than or equal to a specific point in lexicographical order for (y, x). Let $v_{\text{list}}[i]$ be the i-th point of v_{list} and v_{child} be the child of v. The *minmax (maxmin)* *pointer* minmax$(v_{\text{list}}[i], v_{\text{child}})$ (maxmin $(v_{\text{list}}[i], v_{\text{child}})$) is defined as the pointer pointing to the smallest (largest) element of v_{child} larger (smaller) than or equal to $v_{\text{list}}[i]$ in lexicographical order (y, x), respectively. If no element satisfies the condition, the pointer becomes nil. For example, Figure 4.1 shows that all the points of v_1 larger than or equal to B in lexicographical order (y, x) are C, E, F which are located on the right side of the head of the minmax(B, v_1) (blue arc). The pseudocode of *setting minmax and maxmin pointer* is described as "set_pointer" in Algorithm 2, and the overall procedure of building FC tree is also represented as "main" in Algorithm 2.

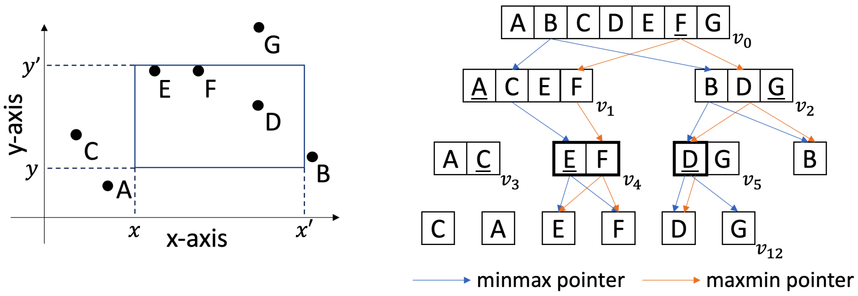

FIGURE 4.1: (Left) Two-dimensional point coordinates with a rectangular range query. (Right) Data structure for fractional cascading. The *minmax and maxmin pointers* used for the rectangular range query are only drawn, and the split points are underlined in each vertex. Reported points are bold.

We explain how the FC answers a two-dimensional orthogonal range query $[x, x'] \times [y, y']$. FC implements a binary search on the x-axis, whereas *minmax* and *maxmin pointers* automatically performs y-axis search. We search all the points lying in $[y, y']$ at only the root vertex, and *minmax and maxmin pointers* perform the y-axis check for other vertices automatically. Algorithm 3 presents the pseudocode of orthogonal range searching, and an example is drawn in Figure 4.1. We first enumerate the candidate points in terms of y-axis at the root node v_0; that is, the points lying in $[y, y']$ are specified (B, C, D, E, and F remain). If the x-coordinate of $v_{0_{\text{split}}} (= F)$ is greater than or equal to x, we continue searching for the left child of v_0 by following *minmax* and *minmax pointers*. Precisely, we follow the *minmax pointer* of the left-end point among the remaining points. In this case, we follow *minmax*(B, v_1) and reach C in v_1. We also follow *maxmin pointer* of the right-end point F in v_0 and reach F in v_1. Notably, each point in v_1 lies in $[y, y']$ if and only if the point is located between C and F (heads of the two pointers). This fact is directly deduced from the definitions of *minmax* and *maxmin pointers* and implies that the y-axis search is sufficient to be performed only at the root. The operation, deciding whether the child needs to be explored and following *minmax* and *minmax pointers*, is performed recursively until all the remaining points are included in $[x, x']$ in the current vertex. With such a situation, we report all these points. For example, all the points of v_4 lie in $[x, x']$; hence, E and F are reported without searching for children of v_4.

Algorithm 2: *constructing a fractional cascading tree*(S)

Input : a set of 2-dimensional data points S
Output: a data structure for fractional cascading

Function `fctree(list)`:
    $v_{\text{list}} \leftarrow$ list
    $v_{\text{split}} \leftarrow$ median point of "list" in lexicographical order (x, y)
    leftlist $\leftarrow$ sub-list consisting of all the points less than or equal to v_{split} in lexicographical order (x, y)
    rightlist $\leftarrow$ sub-list consisting of all the points greater than v_{split} in lexicographical order (x, y)
    if len(leftlist) > 0 **then**
        $v_{\text{left}} \leftarrow$ `fctree`(leftlist)
    if len(rightlist) > 0 **then**
        $v_{\text{right}} \leftarrow$ `fctree`(rightlist)
    return v

Function `create_fctree(`S`)`:
    list $\leftarrow$ sorted array of points S in lexicographical order (y, x)
    $v_{\text{root}} \leftarrow$ `fctree(list)`
    return v_{root}

Function `set_pointer(`v_{root}`)`:
    V $\leftarrow \{v_{\text{root}}\}$
    while $V \neq \emptyset$ **do**
        $v \leftarrow$ V.pop()
        for side $\in$ *{* left, right *}* **do**
            if there exists v_{side} **then**
                for $p \in v_{\text{list}}$ **do**
                    minmax$(p, v_{\text{side}}) \leftarrow$ the smallest element of v_{side} larger than or equal to p in lexicographical order (y, x) if it exists; otherwise, *nil*
                    maxmin$(p, v_{\text{side}}) \leftarrow$ the largest element of v_{side} smaller than or equal to p in lexicographical order (y, x) if it exists; otherwise, *nil*
                V.add(v_{side})

Function `main(`S`)`:
    $v_{\text{root}} \leftarrow$ `create_fctree(`S`)`
    `set_pointer(`v_{root}`)`
    return v_{root}

Algorithm 3: Two-dimensional search for a rectangular range query using fractional cascading

Input : a query rectangle $[x, x'] \times [y, y']$; an FC tree v_{root}
Output: the set of all the points R which lie in $[x, x'] \times [y, y']$

$S \leftarrow \{v_{\text{root}}\}$
$R \leftarrow \emptyset$
if all points of v_{root} do not lie in $[y, y']$ **then**
    return R
$a_l(v_{\text{root}}) \leftarrow$ the smallest element of v_{root} lying in $[y, y']$
$a_r(v_{\text{root}}) \leftarrow$ the largest element of v_{root} lying in $[y, y']$
while $S \neq \emptyset$ **do**
    `// Pop out an element from a set` S
    $v \leftarrow S$.pop()
    if all points in v from $a_l(v)$ and $a_r(v)$ lies in $[x, x']$ **then**
        all points in v from $a_l(v)$ and $a_r(v)$ are added to R
        continue
    $C \leftarrow \emptyset$
    $sv \leftarrow$ the x coordinate of v_{split}
    if $x \leq sv$ **then**
        add v_{left} to C
    if $sv < x'$ **then**
        add v_{right} to C
    for $v_{child} \in C$ **do**
        $a_l(v_{\text{child}}) \leftarrow \text{minmax}(a_l(v), v_{\text{child}})$
        $a_r(v_{\text{child}}) \leftarrow \text{maxmin}(a_r(v), v_{\text{child}})$
        if $a_l(v_{child}) \neq$ nil and $a_r(v_{child}) \neq$ nil **then**
            `// The most important fact here is that a point of` v_{child} `lies in` $[y, y']$ `if and only if the point is between` $a_l(v_{\text{child}})$ `and` $a_r(v_{\text{child}})$
            add v_{child} to S
return R

Chapter 5

Numerical Experiment

This section first explains the dataset and introduces the parameters used in the experiment. We then compare our TEG-matching with two latest algorithms, namely, the STD-matching (Hsueh and Chen, 2018) and the AntMapper algorithm (Gong et al., 2018), using an open dataset.

5.1 Experiment settings

5.1.1 Dataset

In our experiments, we utilized worldwide vehicle *trajectories* in a public dataset (Kubička et al., 2015). This dataset includes 100 global vehicle *trajectories* (ID $= 0, 1, \ldots, 99$), each of which is associated with the *correct path* and the *road network* around the vehicle *trajectory*. The vehicle *trajectory* is a sequence of GPS points, each of which consists of a timestamp and a longitude–latitude pair. The longitude–latitude pair is converted to 2D coordinates using a UTM-WGS84 converter[1]. We excluded data (ID=2,19,33,41,42,69,75,76,80,86, and 89) that the corresponding *correct path* $(e_1, \ldots, e_m)$ is not a path; that is, there exist consecutive edges e_i, e_{i+1} such that $\mathrm{head}(e_i) \neq \mathrm{tail}(e_{i+1})$. The average distance between two consecutive GPS points is 11 m, which is not suitable for evaluating map matching for low-frequency data. Hence, we sample the GPS points from each GPS trajectory by utilizing bottom-up segmentation with $max\_error = 7$ m. The thinned-out vehicle *trajectories* are used for evaluating map-matching algorithms. Bottom-up segmentation instead of constant interval sampling avoids overlooking significant direction changes, which is a critical assumption of our TEG-matching. The $max\_error = 7$ provides an outline of the vehicle *trajectory* while removing unnecessary position fixes. The number of GPS points and distance between two consecutive GPS points applied and not applied to the bottom-up segmentation are summarized in Table 5.1.

5.1.2 Preprocess

This section explains the parameters and procedure of the map-matching preprocess explained in Section 1.5. We set the upper bound of the spatial measurement error as $r_{\mathrm{GPS}} = 200$ m and the maximum length of the *shape arc* $\ell_{max} = 2(1 + \sqrt{2})r_{\mathrm{GPS}}$ meter. *Shape nodes* and *junctions* (illustrated in Figure 1.1) are identified based on the *road network* topology because the original dataset does not distinguish them. We also split long *shape arcs* (add *shape nodes*) such that the length of each *shape arc* is less than or equal to ℓ_{max} to obtain all the *shape arcs* close to a position fix (see details in Section 1.5). For each area associated with a *vehicle trajectory*, we built a fractional cascading (FC) data structure and applied bottom-up segmentation.

[1]https://github.com/Turbo87/utm

max_error (m)	Original	Bottom-up segmentation 3	7	15	30
#GPS(%)	209,901 (100%)	17,723 (8.4%)	**10,041 (4.8%)**	6,351 (3.0%)	4,316 (2.1 %)
Distance interval (m)	11 ± 12	135 ± 257	$\mathbf{238 \pm 427}$	374 ± 572	549 ± 724

TABLE 5.1: The total number of GPS points (#GPS) and the distance (m) between two consecutive GPS points applied and not applied bottom-up segmentation over all vehicle *trajectories*. "Original" implies the vehicle *trajectories* where bottom-up segmentation is not applied. The percentage of #GPS is the ratio to the original *road network*. The distance interval is written as (mean)±(standard deviation). The *trajectories* used for evaluating map-matching algorithms are **bold**.

5.1.3 Experimental platform

We used only one core of the PC server (2.30 GHz Intel Core E5-2670 with 24 cores and 512 GB of memory). Python 3.8.1 and NetworkX are utilized to calculate the shortest and longest paths. The calculation of the longest path is used for STD-matching that is compared with our TEG-matching.

5.1.4 Evaluation index

We evaluated map-matching algorithms using *arcs*, rather than *shape arcs*, to verify the effectiveness of bottom-up segmentation, which removes redundant *shape nodes* from the *road network*. We compared our approach with existing models in terms of accuracy and speed (#GPS/sec). In our experiment, the accuracy is the intersection over the union of the two multisets of *matched arcs* and *correct arcs*. More precisely, given two multisets of *correct arcs* C_i and *matched arcs* D_i for each *vehicle trajectory* $i \in \{1, \dots, N\}$, the accuracy is defined as follows:

$$\text{Accuracy for i-th trajectory} = \frac{\#(C_i \cap D_i)}{\#(C_i \cup D_i)} \tag{5.1}$$

$$\text{Accuracy for dataset} = \frac{\sum_{i=1}^{N} \#(C_i \cap D_i)}{\sum_{i=1}^{N} \#(C_i \cup D_i)} \tag{5.2}$$

We do not use the average of $\frac{\#(C_i \cap D_i)}{\#(C_i \cup D_i)}$ because the number of position fixes varies per trajectory. For each *vehicle trajectory* $i \in \{1, \dots, N\}$, let T_i be the processing time and $\#\text{GPS}_i$ be the number of remaining GPS points after bottom-up segmentation. Then, the speed (#GPS/sec) is defined as follows:

$$\text{Speed for i-th trajectory} = \frac{\#\text{GPS}_i}{T_i} \tag{5.3}$$

$$\text{Speed for dataset} = \frac{\sum_{i=1}^{N} \#\text{GPS}_i}{\sum_{i=1}^{N} T_i} \tag{5.4}$$

The processing time is defined as the duration from the end of candidate *node* and *arc* search to the output of a *matched path*. We leave out the candidate *node* and *arc* search from the speed because this process is inevitable for all map-matching algorithms.

5.2 Experimental results of fractional cascading

For each GPS point, almost all the matching algorithms must search the *nodes* and *arcs* close to the GPS point; hence, a fast searching speed (#GPS/sec) is vital for map matching. Fractional cascading (FC) accelerates the search for a given rectangular query, as explained in Chapter 4. We calculated the searching speed of the FC, k-d tree, and brute force for a square query with a side of $2r_{\mathrm{GPS}} = 400$ m while changing the number of *shape nodes* in the data structure. Brute force checks individually whether the query square includes a *node*. Table 5.2 shows the speed of each algorithm, using all GPS points. The FC searches the *shape nodes* 2.5x and 195x faster than the k-d tree and brute force search, respectively. The speed and memory usage for different numbers of *shape nodes* are illustrated in Figure 5.1 (theoretical values are summarized in Table 4.1). The speed of FC is faster than that of the kd-tree and brute force for any number of *shape nodes*. However, the speed gradually decreases as the number of *shape nodes* increases, which is compatible with the theoretical query time. Besides, the memory usage of the FC is much larger than that of the kd-tree and brute force. Therefore, if we have enough memory, we should utilize FC; otherwise, the kd-tree is suitable for a candidate *node* search.

	Fractional Cascading	kd-tree	Brute Force
Speed (#GPS/sec)	10,143	3,990	52

TABLE 5.2: Speed (#GPS/sec) of fractional cascading, k-d tree, and brute force for square range query with a side of 400 m. This speed is calculated using all the GPS points.

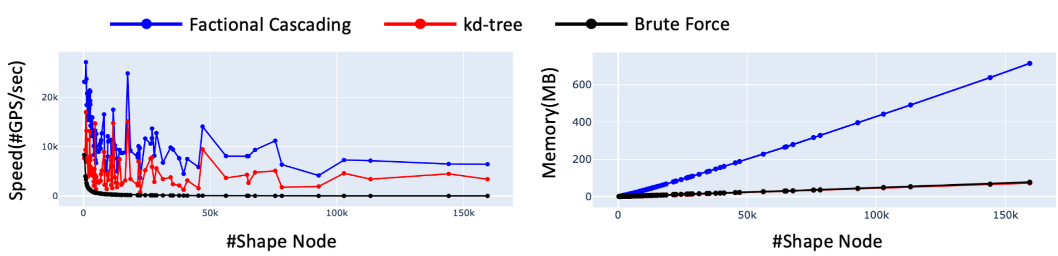

FIGURE 5.1: Speed (#GPS/sec) and memory of FC, k-d tree, and brute force for different numbers of *shape nodes*.

5.3 Map-matching models compared to our model

We compare our TEG-matching with STD-matching (Hsueh and Chen, 2018) and the AntMapper algorithm (Gong et al., 2018), and this section provides an overview of these models.

For each position fix, STD-matching lists the candidate *arcs* where a vehicle may be located, and the vehicle location is supposed to be the projection of the position fix onto the *arc*. To obtain a *matched path*, STD-matching constructs a graph whose nodes are projections and whose arcs are all two projections corresponding to two consecutive positions. The weight of each arc is determined based on two factors: (1) the ratio of the shortest path distance between the two projections to the distance of the corresponding two position fixes and (2) the distance between the position fix

	TEG	STD	AntMapper
Accuracy	**0.9645**	0.8672	0.3568
Speed (#GPS/sec)	**19.3**	3.4	0.2

TABLE 5.3: Accuracy and speed (#GPS/sec) of all algorithms for the dataset. The highest accuracy or speed is **bold**.

and the *arc*. STD-matching finally finds the most reasonable path from the projection of the first position fix to the projection of the last position fix.

In contrast, the AntMapper algorithm is non-deterministic; that is, the algorithm may produce a different *matched path* despite that the *trajectory* is the same. Similar to the case with STD-matching, the AntMapper algorithm matches each position fix to an *arc*. However, it computes both the global likelihood of the path and the local likelihood related to two consecutive projections. The AntMapper algorithm finally merges the local and global likelihoods, and the highest-value path is explored using the ant colony algorithm.

The parameters of STD-matching and AntMapper algorithms are the same as those used in the corresponding literature. Our dataset only includes timestamps and locations; thus, existing algorithms utilize only this information.

5.4 Comparison with all models

The accuracy, speed, and memory usage of our TEG-matching, STD-matching, and AntMapper algorithms are summarized in Table 5.3, Figure 5.2, 5.3 and 5.4. As shown in Table 5.3, our TEG-matching is 0.098 higher and 5.6 times faster than the existing models in terms of accuracy and speed, respectively. Significantly, our TEG-matching outperforms the existing models for almost all the data, as illustrated in Figure 5.2. TEG-matching achieves sufficient accuracy for almost all data compared to the existing methods, whose accuracy are extremely low for some data. The prolonged speed of the AntMapper algorithm results from the ant colony algorithm used for computing the global likelihoods. The ant colony algorithm, one of the population-based metaheuristics, also yields lower accuracy than the shortest path algorithm, one of the optimization algorithms used by TEG- or STD-matching. In general, an optimization algorithm produces a better solution than a heuristic algorithm, which is true for our experiments.

As illustrated in Figure 5.4, the peak memory usage of all algorithms has a weak correlation with the number of *shape nodes*, and the most memory-saved algorithm is difficult to determine. Our TEG-matching and the AntMapper algorithm use large amounts of memory for some trajectories.

We analyzed the *matched paths* and revealed the advantages of TEG-matching over existing algorithms. We have provided some trajectories for which existing algorithms are unsuccessful, but our TEG-matching is successful in its prediction. We have identified the situation in which our TEG-matching works well and why existing algorithms predict incorrect paths. Another trajectory in which our model fails to predict the *correct path* is also provided.

Area weight of TEG-matching contributes to a correct prediction in Figure 5.5. Both STD-matching and AntMapper algorithm match P_{i+2} to *arc* (v_1, v_2) because of a spatial measurement error. In contrast, TEG-matching matches P_{i+2} to *arc* (v_1, v_3)

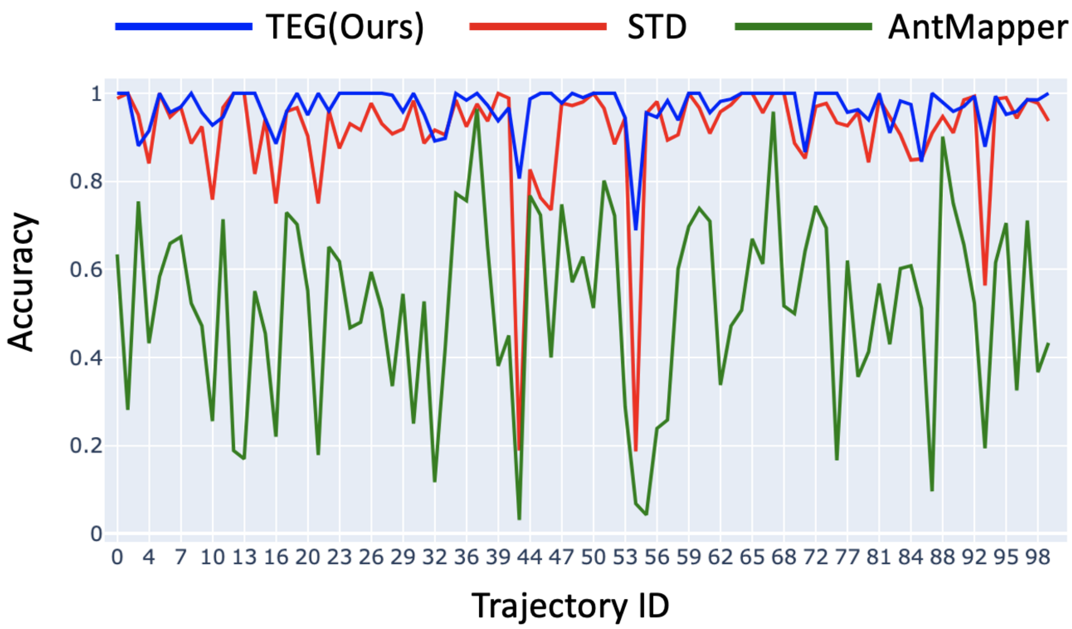

FIGURE 5.2: Accuracy of all algorithms for each *trajectory*.

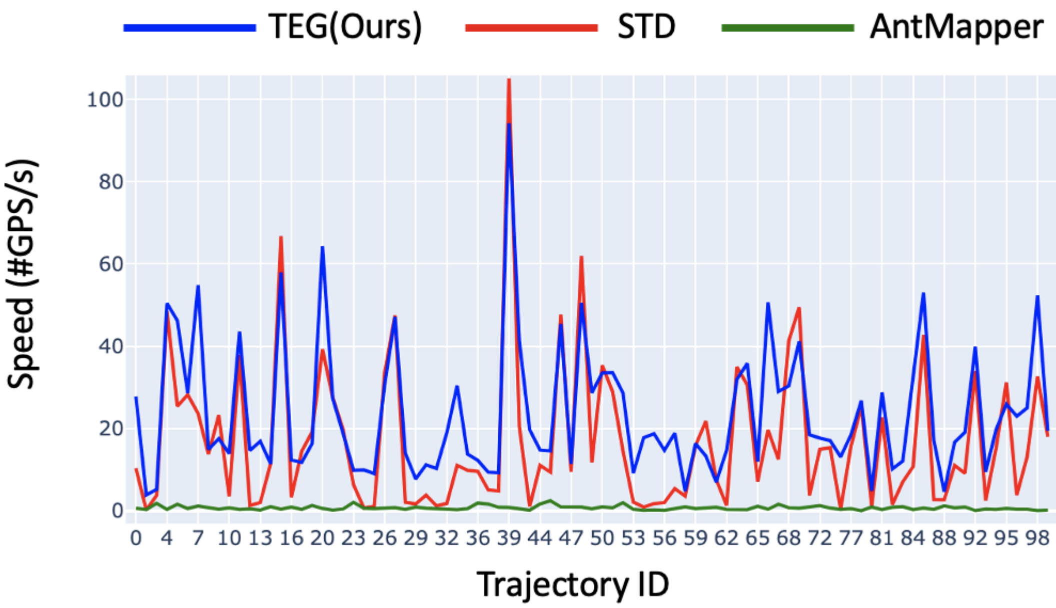

FIGURE 5.3: Speed (#GPS/sec) of all algorithms for each *trajectory*.

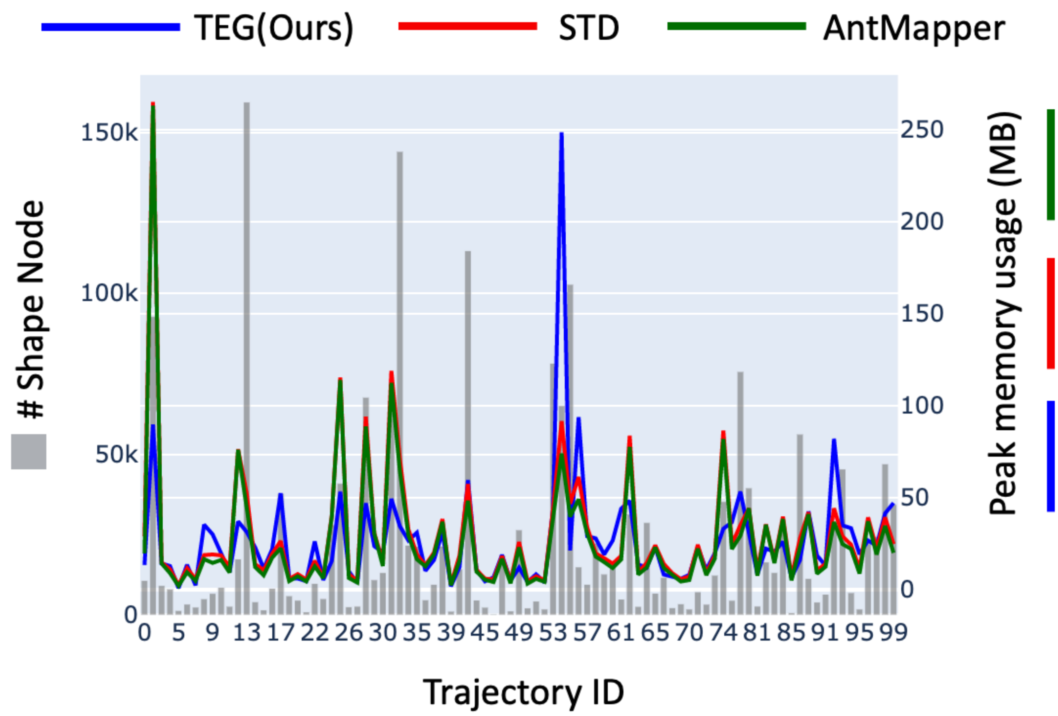

FIGURE 5.4: Peak memory usage of all algorithms and the number of *shape nodes* for each vehicle *trajectory*. The peak memory usage denotes the required memory except the data structure such as FC and kd-tree.

by considering the area between the *trajectory* and a *matched path*. The area becomes small if we match P_{i+2} to *arc* (v_1, v_3) rather than match P_{i+2} to *arcs* (v_1, v_2) and (v_2, v_3). Moreover, the angle difference between (P_{i+2}, P_{i+3}) and P_{i+2}'s matched *arc* becomes large if we match P_{i+2} to (v_1, v_2) and (v_2, v_3) compared to matching P_{i+2} to (v_1, v_3). Therefore, *area weight* of the *correct path* becomes smaller than that of an incorrect path (*matched path* of STD-matching or AntMapper), resulting in the accurate prediction of TEG-matching.

The trajectory (ID=21) shown in Figure 5.6 has a spatial measurement error at the starting position (P_1 and P_2). The STD-matching and AntMapper predict the wrong U-turn paths because the direction between the correct *arc* and the segment (P_1, P_2) is the opposite. The *direction change weight* of our TEG-matching avoids U-turns and helps obtain the *correct path*.

Figure 5.7 (ID=21) is the example where the STD-matching and AntMapper present *matched paths* that go back and forth when the vehicle pauses. STD-matching matches two consecutive position fixes with two points such that the distance between the two position fixes is close to the shortest path distance of the corresponding two *projections*. $P_{i+1} \sim P_{i+6}$ are slightly different from each other, which causes an otiose detour for STD-matching. The inconsistent directions of two consecutive position fixes provide AntMapper with a round-trip path. The *direction change weight* of our TEG-matching helps remove these unnecessary detours.

TEG-matching fails to predict the *matched path* for the trajectory (ID=95) shown in Figure 5.8. The vehicle makes a U-turn in the middle of the long *arc* (v_1, v_2), and the "long" *arc* causes a mistake in the prediction. If our model matches (P_{i+2}, P_{i+3}) and (P_{i+3}, P_{i+4}) with *arcs* (v_1, v_2) and (v_2, v_1) respectively, the *direction change weight* $w_d\left((v_1, v_2)^{i+3}, (v_2, v_1)^{i+3}\right)$ becomes very large because P_{i+2}, P_{i+3} and P_{i+4} are far

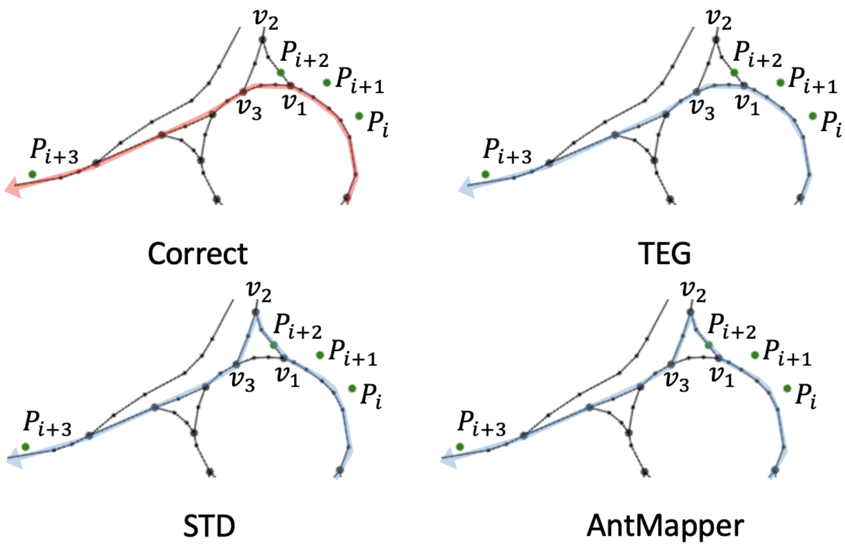

FIGURE 5.5: *Shape nodes* (small black dots), *junctions* (big black dots), a *vehicle trajectory* (green marks), and *matched paths* (sky blue) of map-matching algorithms, as well as the correct path (red).

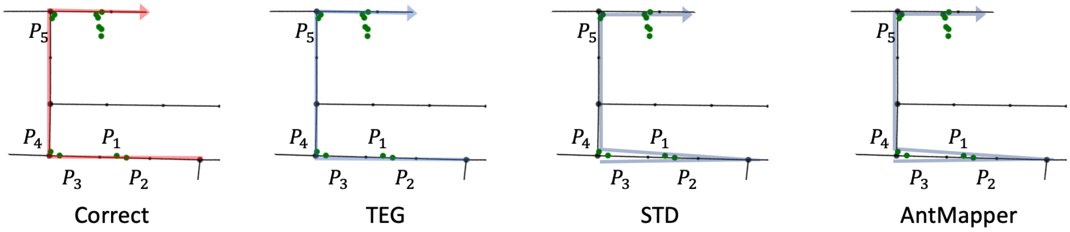

FIGURE 5.6: *Shape nodes* (small black dots), *junctions* (big black dots), a *vehicle trajectory* (green marks), and *matched paths* (sky blue) of map-matching algorithms, as well as the correct path (red).

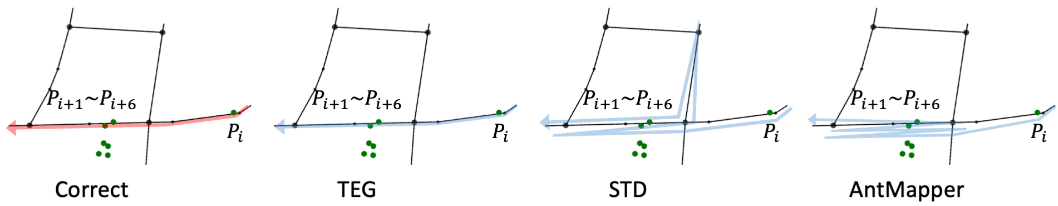

FIGURE 5.7: *Shape nodes* (small black dots), *junctions* (big black dots), a *vehicle trajectory* (green marks), and *matched paths* (sky blue) of map-matching algorithms, as well as the correct path (red).

from v_2. Therefore our models selects the shortcut path $(\dots, (v_1, v_3), (v_3, v_4), (v_4, v_3), \dots)$. To avoid this situation, we need to split long *arcs* at direction change *shape nodes*. AntMapper is a non-deterministic algorithm that offers an incorrect path containing two round-trips, and STD-matching yields the correct *matched path*.

In summary, our TEG-matching is robust against spatial measurement errors and pauses, such as traffic lights. However, if the vehicle makes a U-turn in the middle of a long *arc*, the TEG-matching outputs an incorrect shortcut path. We can solve this problem by splitting long *arcs* at direction change *shape nodes*.

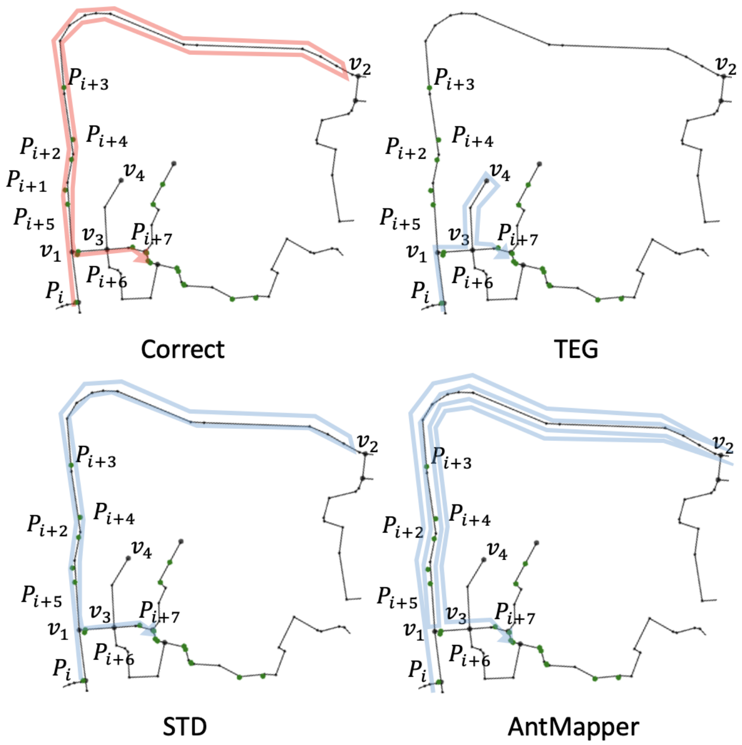

FIGURE 5.8: *Shape nodes* (small black dots), *junctions* (big black dots), a *vehicle trajectory* (green marks), and *matched paths* (sky blue) of map-matching algorithms, as well as the correct path (red).

5.5 Sensitivity analysis

5.5.1 Impact of bottom-up segmentation

The bottom-up segmentation explained in Chapter 3 reduces the graph size while preserving the topology of a *road network*. Table 5.4 shows the number of *shape nodes*, speed, and accuracy of our TEG-matching with or without bottom-up segmentation. In the table, "Original" implies not applying bottom-up segmentation to the *road network*. Surprisingly, a tiny parameter value (max_error $= 3$) achieves approximately 50% reduction in *shape nodes*, which implies that we succeed in significant node reduction while preserving the shape of the original *road network*. We achieved a 1.78x speed increase with only a 0.0074 accuracy drop at the max_error=10, where the accuracy 0.9571 is still higher than the existing models (STD:0.8672 and AntMapper:0.3568). The accuracy and speed for each *trajectory* are shown in Figure 5.9 and 5.10, respectively. Bottom-up segmentation achieves an effective map-matching speedup with only a small accuracy drop.

Figure 5.11 shows the vehicle *trajectory* whereby bottom-up segmentation has a negative influence on the accuracy. Bottom-up segmentation brings the *correct arc* away for the position fix P_i; hence, the *matched path* is wrong. The appropriate max_error is the key to balancing the accuracy and speed of map matching. In contrast, bottom-up segmentation occasionally improves the accuracy, as shown in Figure 5.12. Our TEG-matching produces a shortcut path with a U-turn (yellow *arc*) for the original *road network*. The long purple *arc* is unlikely to be chosen because the long *arc* increases the *area weight*. Besides, P_{i+1} and P_{i+2} achieve perpendicular

max_error (meter)	Original	Bottom-up segmentation 3	5	10	30
#Shape node(%)	1,996 K (100%)	1,045 K (52%)	890 K (45%)	725 K (36%)	553 K (28 %)
Accuracy	**0.9645**	0.9622	0.9619	0.9571	0.9469
Speed (#GPS/sec)	19.3	30.9	32.0	34.3	**44.20**

TABLE 5.4: Number of *shape nodes*, accuracy and speed of our TEG-matching with or without *bottom-up segmentation*. The *shape nodes* covers areas associated with all vehicle *trajectories*. "Original" implies that bottom-up segmentation is not applied to the *road network*. K represents $\times 10^3$, and a percentage is a ratio to the original *shape node*'s number. The highest accuracy and speed are **bold**.

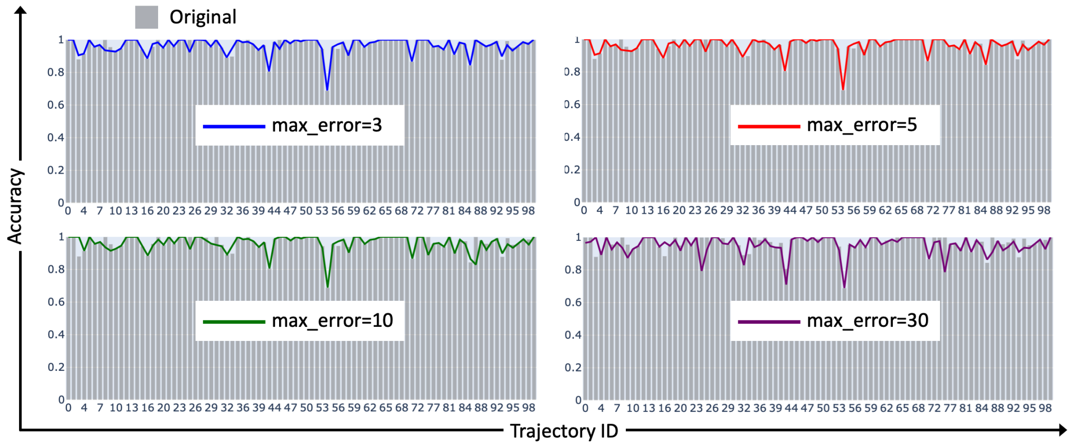

FIGURE 5.9: Accuracy of our TEG-matching for each *trajectory* while changing the max_error of the bottom-up segmentation. "Original" implies no use of bottom-up segmentation to the *road network*.

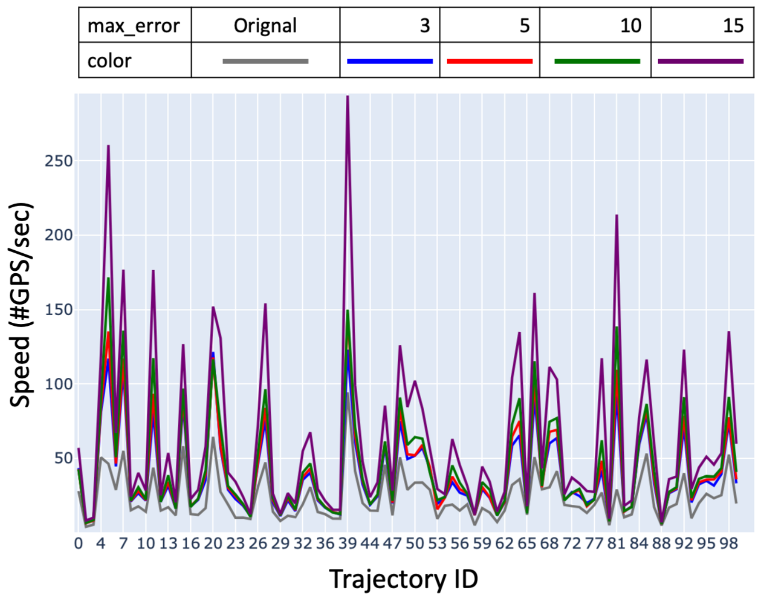

FIGURE 5.10: Speed (#GPS/sec) of our TEG-matching for each *trajectory* while changing the max_error of bottom-up segmentation. "Original" implies no use of bottom-up segmentation to the *road network*.

distances on the yellow *arc*, which reduces the *area weight* of the *arc*. After bottom-up segmentation, P_{i+2} does not achieve a perpendicular distance on the yellow *arc*; therefore, TEG-matching is successful in prediction.

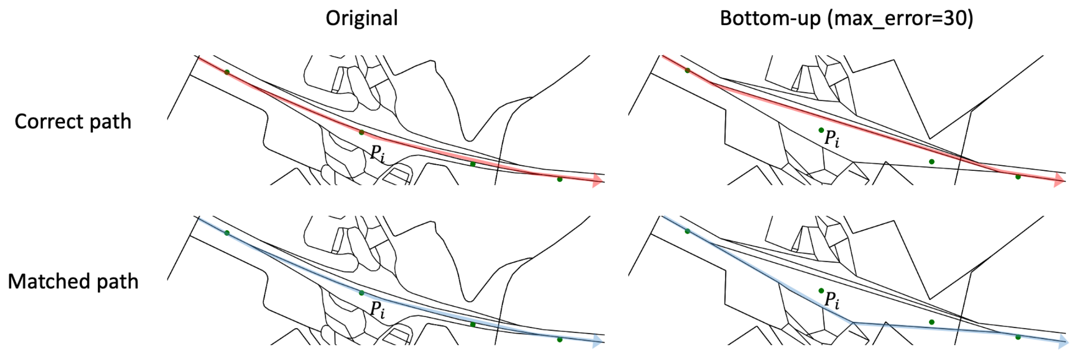

FIGURE 5.11: Vehicle trajectory (green marks), *correct path* (red), and *matched paths* (sky blue) of TEG-matching applied and not applied bottom-up segmentation, as well as the *correct path* (red).

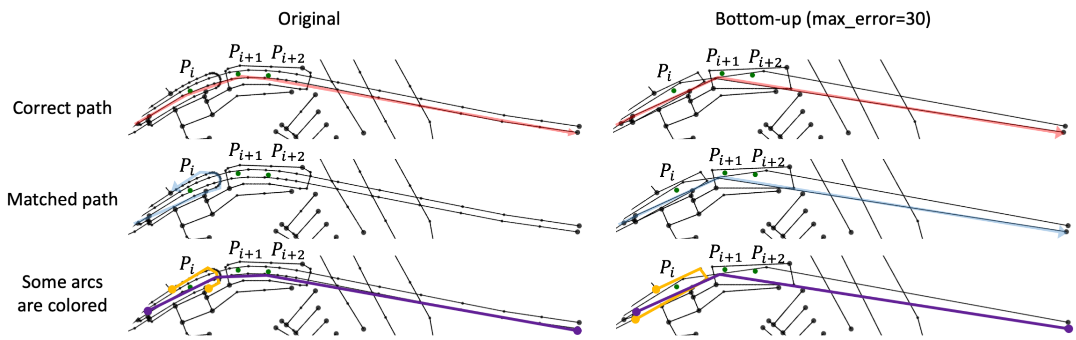

FIGURE 5.12: Vehicle trajectory (green marks), *correct path* (red), and *matched paths* (sky blue) of TEG-matching applied and not applied bottom-up segmentation, as well as some colored *arcs*.

Chapter 6

Conclusion

We propose a parameter-free map-matching algorithm called TEG-matching. TEG-matching achieves an accuracy improvement and 5.6x speedup than existing models. TEG-matching constructs a time-dependent graph and solves the shortest path problem to obtain the most plausible space-time path. Numerical experiments indicate that our TEG-matching is robust against spatial measurement errors and pauses such as at traffic lights. However, an appropriate *arc* split is required for further accuracy improvement. We also performed a theoretical analysis and determined how large a square is needed to obtain all the *arcs* within a radius r from a certain point. Moreover, we utilized the fractional cascading to achieve a high-speed neighborhood search. Bottom-up segmentation also achieves a 64% reduction in *shape nodes*, resulting in a 1.78x speed increase with only a small accuracy reduction for map matching.

The proposed algorithm is promising for offline usage in ITS, such as traffic dynamics analysis and urban planning to alleviate traffic congestion. The traffic dynamics analysis applies data mining methods to understand spatial and social behavior such as travelers' route choice, accessibility patterns, and commercial center attractiveness. Advantageous properties of our algorithm are (1) high speed and accuracy for low-frequency data, (2) being parameter-free, and (3) only requiring ordered locations for map matching, which are highly beneficial to a practical case that requires high performance and reduces the cost of data transmission and tuning hyperparameters.

Appendix A

Proof about Square Query

We utilize square query to find all *arcs* near a *vehicle trajectory*. To this end, we repeatedly obtain all *shape arcs* contained in a square. This section answers how large the square is required to obtain either of the endpoints of the *shape arc* located within a radius r meter from a point. Theorem 2 presents the side length of the square, and both Lemma 1 and Lemma 2 are used to prove the Theorem 2.

Lemma 1. *Let* $0 < r \leq s, \partial C := \left\{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\boldsymbol{x}\|_2 = r\right\}$*, and* $\partial L := \left\{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_\infty = s\right\}$*. Then we have*

$$\min_{(x,y)\in\partial C} l(x,y) = \min\left\{2\left(\sqrt{2}s - r\right), 2s\right\}$$

where $l(x,y)$ *is the distance between the two intersection points between* ∂L *and the tangent line at* (x,y) *of* ∂C*. If* $s = r$ *and the tangent line is either* $x = r$*,* $x = -r$*,* $y = r$*, or* $y = -r$*, the intersection points are infinite. In this case, we define* $l(x,y) = 2s$*, which is compatible with this lemma.*

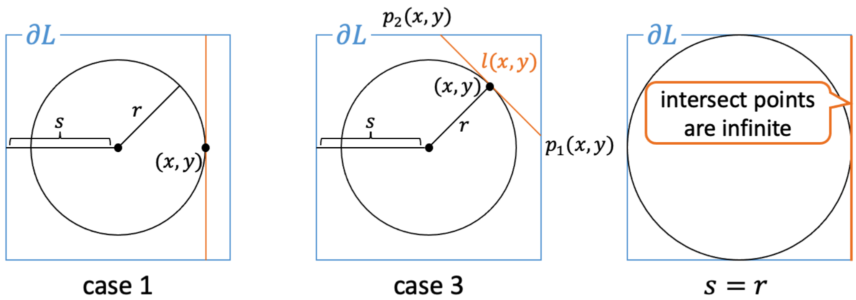

FIGURE A.1: Visualization of the symbols used in the lemma.

Proof. By symmetry, we only consider the point of tangency at $0 \leq x \leq r, 0 \leq y \leq r$. Given the circle with the center at the origin and radius of r, the tangent line at $(x_0, y_0)^T \in \mathbb{R}^2$ is given by the following equation:

$$x_0 x + y_0 y = r^2 \tag{A.1}$$

Hence, the intersection conforms to any of the following three cases:
(1) The tangent line intersects ∂L at $y = s$ and $y = -s$.
(2) The tangent line intersects ∂L at $x = s$ and $x = -s$.
(3) The tangent line intersects ∂L at $x = s$ and $y = s$.
In cases (1) and (2), $l(x,y)$ takes minimum value $l(x,y) = 2s$ if the tangent line is parallel to either the x-axis or y-axis; hence, we only consider case (3) in the following proof. Because the tangent point is neither $x = 0$ nor $y = 0$, the tangent line at

$(x, y)^T$ $(0 < x, y < r)$ intersects ∂L at the following two points:

$$p_1(x,y) = \left(s, \frac{r^2 - sx}{y}\right)^T, p_2(x,y) = \left(\frac{r^2 - sy}{x}, s\right)^T \tag{A.2}$$

By using polar coordinates, the distance between $p_1(x,y)$ and $p_2(x,y)$, denoted by $l(x,y)$, is expressed by

$$\begin{aligned}
(l(x,y))^2 &= \|p_1(x,y) - p_2(x,y)\|_2^2 &\text{(A.3)}\\
&= \left(\frac{r^2 - s(x+y)}{x}\right)^2 + \left(\frac{r^2 - s(x+y)}{y}\right)^2 &\text{(A.4)}\\
&= \frac{(r^2 - s(x+y))^2(x^2+y^2)}{x^2y^2} &\text{(A.5)}\\
&= \frac{r^2(r^2 - sr(\cos\theta + \sin\theta))^2}{r^4 \sin^2\theta\cos^2\theta} &\text{(A.6)}\\
&= \frac{(r - s(\cos\theta + \sin\theta))^2}{\sin^2\theta\cos^2\theta} &\text{(A.7)}\\
&\equiv f(\theta), &\text{(A.8)}
\end{aligned}$$

where

$$x = r\cos\theta, y = r\sin\theta \tag{A.9}$$

Because $l(x,y)$ is non-negative, the problem is equivalent to finding the minimum value of $f(\theta)$ on $\theta \in \left(0, \frac{\pi}{2}\right)$.

We perform the first derivative test and find the global minimum value. Because

$$(\sin^2\theta\cos^2\theta)' = 2\sin\theta\cos\theta(\cos^2\theta - \sin^2\theta), \tag{A.10}$$

the derivative of f, denoted by f', agrees with the following equation:

$$\begin{aligned}
f'(\theta) =& \frac{-2s\sin^2\theta\cos^2\theta(\cos\theta - \sin\theta)\left(r - s(\sin\theta + \cos\theta)\right)}{\sin^4\theta\cos^4\theta} &\text{(A.11)}\\
&- \frac{2\sin\theta\cos\theta\left(\cos^2\theta - \sin^2\theta\right)\left(r - s(\sin\theta + \cos\theta)\right)^2}{\sin^4\theta\cos^4\theta} &\text{(A.12)}\\
=& \frac{2(r - s(\sin\theta + \cos\theta))\left(s(1 + \sin\theta\cos\theta) - r(\sin\theta + \cos\theta)\right)(\cos\theta - \sin\theta)}{\sin^3\theta\cos^3\theta}. &\text{(A.13)}
\end{aligned}$$

We first evaluate $r - s(\sin\theta + \cos\theta)$. Because $\sin\theta + \cos\theta = \sqrt{2}\sin\left(\theta + \frac{\pi}{4}\right)$ and $0 < \theta < \frac{\pi}{2}$,

$$1 < \sin\theta + \cos\theta \leq \sqrt{2} \quad \left(\frac{\pi}{4} < \theta + \frac{\pi}{4} < \frac{3}{4}\pi\right) \tag{A.14}$$

Hence,

$$r - s(\sin\theta + \cos\theta) < r - s \leq 0 \tag{A.15}$$

Next, we handle $s(1+\sin\theta\cos\theta)-r(\sin\theta+\cos\theta)$ by transforming the equation as follows:

$$s(1+\sin\theta\cos\theta)-r(\sin\theta+\cos\theta) \geq s\left(1+\sin\theta\cos\theta-(\sin\theta+\cos\theta)\right) \tag{A.16}$$
$$= s(1-\sin\theta)(1-\cos\theta) > 0 \tag{A.17}$$

We finally consider $\cos\theta-\sin\theta$. Because $\cos\theta-\sin\theta = -\sqrt{2}\sin\left(\theta-\frac{\pi}{4}\right)$ and $-\frac{\pi}{4} < \theta-\frac{\pi}{4} < \frac{\pi}{4}$ when $0<\theta<\frac{\pi}{2}$, we have

$$-\frac{\pi}{4} < \theta-\frac{\pi}{4} < 0 \quad \left(0<\theta<\frac{\pi}{4}\right) \Rightarrow \cos\theta-\sin\theta > 0 \tag{A.18}$$
$$0 \leq \theta-\frac{\pi}{4} < \frac{\pi}{4} \quad \left(\frac{\pi}{4}\leq\theta<\frac{\pi}{2}\right) \Rightarrow \cos\theta-\sin\theta \leq 0 \tag{A.19}$$

From the above discussions, we conclude as follows:

$$0<\theta<\frac{\pi}{4} \Rightarrow f'(\theta) < 0 \tag{A.20}$$
$$\frac{\pi}{4}\leq\theta<\frac{\pi}{2} \Rightarrow f'(\theta) \geq 0 \tag{A.21}$$

Therefore, $f(\theta)$ has a minimum value $4\left(\sqrt{2}s-r\right)^2$ at $\theta=\frac{\pi}{4}$ on $(0,\frac{\pi}{2})$, implying that $l(x,y)$ finds the minimum value $\sqrt{4\left(\sqrt{2}s-r\right)^2} = 2\left(\sqrt{2}s-r\right)$ at $|x|=|y|=\frac{r}{\sqrt{2}}$. □

Lemma 2. *Let $0<r\leq s$ and $L := \{\boldsymbol{x}\in\mathbb{R}^2 \mid \|\boldsymbol{x}\|_\infty \leq s\}$. Any line intersecting $C := \{\boldsymbol{x}\in\mathbb{R}^2 \mid \|\boldsymbol{x}\|_2 \leq r\}$ has exactly two intersection points with ∂L, and the distance between the two intersection points has the minimum value $\min\{2(\sqrt{2}s-r), 2s\}$. If $s=r$ and the line is either $x=r$, $x=-r$, $y=r$, or $y=-r$, the intersection points are infinite. In this case, we define the distance as $2s$, which is compatible with this lemma.*

Proof. We divide this problem into three cases according to the conditions of the line. Here, we define r' as the distance between the line and the origin.
(1) $r'=0$
The distance has a minimum value of $2s$ when the line is parallel to the x-axis or y-axis.
(2) $0<r'\leq r$
Any line not going through the origin is regarded as the tangent line of the origin-centered circle with a radius of r'; hence, we denote the tangent point by $(x,y)^T$ ($\|(x,y)^T\|_2 = r'$). Let $l(x,y)$ be the distance between the two intersection points between the tangent line and ∂L. From Lemma 1, $l(x,y)$ has a minimum value of $\min\{2(\sqrt{2}s-r'), 2s\}$.
From (1) and (2), we can conclude that the distance has a minimum value of $\min\{2(\sqrt{2}s-r), 2s\}$. □

Theorem 2. *Consider the road network whose shape arc is represented as a straight segment. Let $0<r\leq s$, $\ell_{\max}>0$ be the maximum length of the shape arc that satisfies $\ell_{\max} \leq \min\{2\left(\sqrt{2}s-r\right), 2s\}$. Then, for the shape arc (u,v) and any point P, we have*

$$d\left((u,v),P\right) \leq r \Rightarrow \min\left\{\|u-P\|_\infty, \|v-P\|_\infty\right\} \leq s$$

. Moreover, if $\ell_{\max} \leq 2\left(\sqrt{2}+1\right)r$, then $\ell_{\max} \leq \min\left\{2\left(\sqrt{2}s-r\right), 2s\right\}$ is equivalent to $\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s$. Otherwise, $\frac{\ell_{\max}}{2} \leq s$.

Proof. Without losing generality, we may assume that P is located at the origin by shifting the road network in parallel. Except for the equivalent condition, the claim is directly induced from Lemma 2; hence, we prove the equivalent condition by considering the following two cases:
(1) $2\left(\sqrt{2}s-r\right) \leq 2s$
(2) $2\left(\sqrt{2}s-r\right) \geq 2s$
In case (1), as $\min\left\{2\left(\sqrt{2}s-r\right), 2s\right\} = 2\left(\sqrt{2}s-r\right)$,

$$\ell_{\max} \leq \min\left\{2\left(\sqrt{2}s-r\right), 2s\right\} \Leftrightarrow \frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \tag{A.22}$$

. In case (2), as $\min\left\{2\left(\sqrt{2}s-r\right), 2s\right\} = 2s$,

$$\ell_{\max} \leq \min\left\{2\left(\sqrt{2}s-r\right), 2s\right\} \Leftrightarrow \frac{\ell_{\max}}{2} \leq s \tag{A.23}$$

On the other hand, performing a simple transformation, we have

$$2\left(\sqrt{2}s-r\right) \leq 2s \Leftrightarrow s \leq \left(\sqrt{2}+1\right)r \tag{A.24}$$

Therefore

$$\ell_{\max} \leq \min\left\{2\left(\sqrt{2}s-r\right), 2s\right\} \tag{A.25}$$

$$\Leftrightarrow \left(\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \leq \left(\sqrt{2}+1\right)r\right) \vee \left(\max\left\{\frac{\ell_{\max}}{2}, \left(\sqrt{2}+1\right)r\right\} \leq s\right) \tag{A.26}$$

If $\ell_{\max} \leq 2\left(\sqrt{2}+1\right)r$, $\max\left\{\frac{\ell_{\max}}{2}, \left(\sqrt{2}+1\right)r\right\} = \left(\sqrt{2}+1\right)r$. Therefore, if we are concerned about

$$\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq \left(\sqrt{2}+1\right)r \Leftrightarrow \ell_{\max} \leq 2\left(\sqrt{2}+1\right)r \tag{A.27}$$

, we have

$$\left(\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \leq \left(\sqrt{2}+1\right)r\right) \vee \left(\max\left\{\frac{\ell_{\max}}{2}, \left(\sqrt{2}+1\right)r\right\} \leq s\right) \tag{A.28}$$

$$\Leftrightarrow \left(\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \leq \left(\sqrt{2}+1\right)r\right) \vee \left(\left(\sqrt{2}+1\right)r \leq s\right) \tag{A.29}$$

$$\Leftrightarrow \left(\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \vee \left(\sqrt{2}+1\right)r \leq s\right) \wedge \left(s \leq \left(\sqrt{2}+1\right)r \vee \left(\sqrt{2}+1\right)r \leq s\right) \tag{A.30}$$

$$\Leftrightarrow \min\left\{\frac{\ell_{\max}+2r}{2\sqrt{2}}, \left(\sqrt{2}+1\right)r\right\} \leq s \tag{A.31}$$

$$\Leftrightarrow \frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \tag{A.32}$$

. Otherwise, if $\ell_{\max} > 2\left(\sqrt{2}+1\right)r$, we have $\frac{\ell_{\max}+2r}{2\sqrt{2}} > \left(\sqrt{2}+1\right)r$ from A.27. Therefore, we have

$$\left(\frac{\ell_{\max}+2r}{2\sqrt{2}} \leq s \leq \left(\sqrt{2}+1\right)r\right) \vee \left(\max\left\{\frac{\ell_{\max}}{2}, \left(\sqrt{2}+1\right)r\right\} \leq s\right) \tag{A.33}$$

$$\Leftrightarrow \max\left\{\frac{\ell_{\max}}{2}, \left(\sqrt{2}+1\right)r\right\} \leq s \tag{A.34}$$

$$\Leftrightarrow \frac{\ell_{\max}}{2} \leq s \tag{A.35}$$

. □

Appendix B

NewTEG-matching

In the main body of our paper, we propose TEG-matching that obtains the *matched path* that has a small area between a *vehicle trajectory*. TEG-matching outperforms state-of-the-art algorithms in terms of accuracy and speed; however, it has the following problems:

- TEG-matching implicitly assumes that at most one GPS is obtained each *arc*; however, this situation is not actual for a long *arc* or high-frequency GPS data.
- We understand the motivation and exact definition of an area between a *vehicle trajectory* and *matched path*, but the calculation is complex and time expensive.

Therefore, Appendix B proposes a more straightforward, intuitive, and high-speed map matching algorithm, called NewTEG-matching, that solves these problems. We perform the theoretical analysis for NewTEG-matching, and comprehensive experiments will be conducted in our future research.

B.1 Abstract of NewTEG-matching

NewTEG-matching finds the vehicle route that has the smallest area between a *vehicle trajectory*. Because vehicle route candidates are innumerable and hence impossible to calculate an are for each route, we define the area as the sum of areas between the *vehicle trajectory* and the arc contained in the route. The area between the *vehicle trajectory* and an arc is named as *arc area*. We develop an interpretative model by defining the *arc area* simply and intuitively. Finally, we propose a new time-expanded graph and transform a map-matching problem into the shortest path problem on the new TEG (NewTEG).

The remainder of this Appendix is organized as follows: Section B.2 presents a problem setting and symbols used in Appendix B. We define *arc area* and an area between a route and *vehicle trajectory* in Section B.3. Section B.4 describes the algorithm that calculates the areas. NewTEG and its weight are defined in Section B.5. We finally prove that the shortest path on NewTEG corresponds to the route with the minimum area between a *vehicle trajectory* in Section B.6.

B.2 Problem setting and symbols

This section formulates a map matching problem and defines the symbols used in Appendix B. We are concerned that (1) the output of NewTEG-matching is different from TEG-matching; (2) symbols used in Appendix B have different meanings to the main body ones.

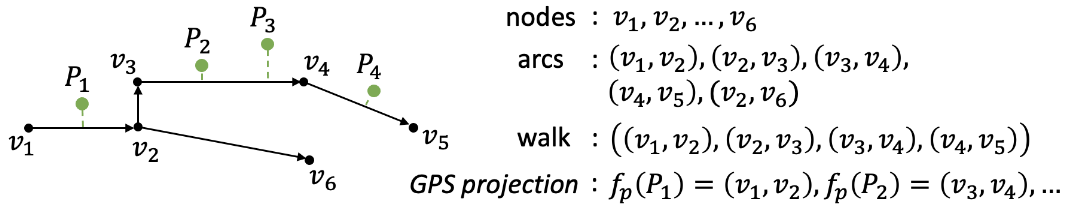

FIGURE B.1: Example of a *road network*, a *vehicle trajectory* $\boldsymbol{P} = (P_i)_{i=1}^{4}$, and a *detailed walk*.

Symbol	Definition
V	set of nodes
A	set of arcs
$G = (V, E)$	*road network*
$\ell_a \ (a \in A)$	length of an arc
$\mathrm{tail}(a) \ (a \in A)$	tail of the arc a
$\mathrm{head}(a) \ (a \in A)$	tail of the arc a
$\boldsymbol{P} = (P_i)_{i=1}^{n}$	*vehicle trajectory* from time stamp i to n
$W = (a_1, a_2, \ldots, a_m) \ \ (a_j \in A)$	walk on a *road network*
$f_p : \{P_1, P_2, \ldots, P_n\} \to W$	*GPS projection*
$\mathcal{W}_{\mathrm{DW}}$	set of *detailed walks*
$d(x, y)$	Euclidean distance between x and y
$d(x, y)_\infty$	Chebyshev distance between x and y
$f_{\mathrm{idx}} : W \to \{1, \ldots, m\}, a_j \mapsto j$	*index function* for an walk $W = (a_1, a_2, \ldots, a_m)$
$\mathcal{W}_{\mathrm{TEG}}$	set of s-t walks on a NewTEG

TABLE B.1: Symbols in Appendix B

We regard a 2D *road network* $G = (V, A)$ where V is the set of nodes and A is the set of arcs (directed edges). Unlike the main body, a node is one type and hence is not categorized into *junctions* and *shape nodes*. An arc is an ordered pair of adjacent nodes. For an arc $a = (v, v') \in A$, ℓ_a is the length of the arc, and $\mathrm{tail}(a) = v$ and $\mathrm{head}(a) = v'$ denotes the tail and head of the arc, respectively. A *vehicle trajectory* is a chronologically ordered position fixes $\boldsymbol{P} = (P_i)_{i=1}^{n} = (P_1, P_2, \ldots, P_n)$ produced by a GPS device mounted on a vehicle. For each time step i, P_i includes only the east and north coordinates. NewTEG-matching aims to restore the most likely walk $W = (a_1, a_2, \ldots, a_m) \quad (a_j \in A, \mathrm{head}(a_j) = \mathrm{tail}(a_{j+1}))$ of the vehicle under a given vehicle trajectory $\boldsymbol{P} = (P_i)_{i=1}^{n}$. NewTEG-matching also specifies the arc where the vehicle is located at each time step, and the function is named *GPS projection* and denoted by $f_p : \{P_1, P_2, \ldots, P_n\} \to W$. A pair of walk W and *GPS projection* f_p is named an *detailed walk* $W_{\mathrm{DW}} = (W, f_p)$, and its set is denoted by $\mathcal{W}_{\mathrm{DW}}$. In summary, NewTEG-matching outputs a *detailed walk* given a *vehicle trajectory*. Figure B.1 shows a *road network*, a *GPS trajectory*, and a *detailed walk*. Symbols used in this Appendix are arranged in Table B.1.

B.3 Detailed walk area

NewTEG-matching finds the *detailed walk* whose walk has the smallest area between a *vehicle trajectory*. The area is named as a *detailed walk area* and is defined as the sum of *arc areas* of the arcs contained in a walk, where an *arc area* is the area bounded between the arc and a *vehicle trajectory*. This section provides the definition of *arc area* and *detailed walk area*.

Arc area is defined as the product of the arc length and the average distance between the GPS and actual vehicle position when the vehicle passes through the arc. We estimate the average distance by dividing into two cases whether or not at least one GPS point is obtained during the transition on the arc $a = (v, v')$. Figure B.2 illustrates an *arc area* for each case, and the formal definition is as follows. Let $\boldsymbol{P} = (P_i)_{i=1}^{n}$ be a *vehicle trajectory*, $W_{\mathrm{DW}} = (W, f_p)$ be a *detailed walk*, and $a_j = (v, v')$ be the arc contained in the walk W. Then, the *arc area* $w\left((v, v'), \boldsymbol{P}, f_p\right)$ is defined as follows:

- case1 (at least one GPS point is obtained when the vehicle passes the arc a)
  Suppose that $P_{i_s}, P_{i_s+1}, \ldots, P_{i_e}$ are all the GPS points obtained during the transition on the arc $a_j = (v, v')$, i.e.,

  $$\begin{aligned} & f_p\left(P_i\right) = a_j \ (i_s \leq i \leq i_e) \\ & \wedge i_s - 1 \geq 1 \Rightarrow f_p\left(P_{i_s-1}\right) \neq a_j \\ & \wedge i_e + 1 \leq n \Rightarrow f_p\left(P_{i_e+1}\right) \neq a_j \end{aligned} \tag{B.1}$$

  . Then, the average distance between the vehicle and GPS position is estimated as $\frac{1}{i_e - i_s + 1} \sum_{i=i_s}^{i_e} d\left(P_i, (v, v')\right)$ where $d(X, Y)$ is the Euclidean distance between X and Y. Therefore, we define

  $$w\left((v, v'), \boldsymbol{P}, f_p\right) := \ell_{(v,v')} \cdot \frac{1}{i_e - i_s + 1} \sum_{i=i_s}^{i_e} d\left(P_i, (v, v')\right)$$

  .

- case2 (GPS point P_{i_s} and P_{i_s+1} are observed before and after the transition on the arc a_j, respectively)
  This case is formally expressed as:

  $$f_{\mathrm{idx}}\left(f_p(P_{i_s})\right) < j < f_{\mathrm{idx}}\left(f_p(P_{i_s+1})\right) \tag{B.2}$$

  , where $f_{\mathrm{idx}} : W \to \{1, \ldots, m\}, a_j \mapsto j$ returns the index of the walk W and is names as an *index function* for an walk W. Then, we estimate the average distance between the vehicle and GPS position as the mean distance between an endpoint of the arc and a segment (P_{i_s}, P_{i_s+1}). Therefore, we define

  $$w\left((v, v'), \boldsymbol{P}, f_p\right) := \ell_{(v,v')} \cdot \frac{1}{2}\left(d\left(v, (P_{i_s}, P_{i_s+1})\right) + d\left(v', (P_{i_s}, P_{i_s+1})\right)\right)$$

  .

Using *arc area* $w\left((v, v'), \boldsymbol{P}, f_p\right)$, a *detailed walk area* $w(W, \boldsymbol{P}, f_p)$ is defined as:

$$w(W, \boldsymbol{P}, f_p) := \sum_{j=1}^{m} w\left(a_j, \boldsymbol{P}, f_p\right) \tag{B.3}$$

, where $W = (a_1, a_2, \ldots, a_m)$. Because NewTEG-matching finds the *detailed walk* that has the minimum *detailed walk area*, the output of NewTEG-matching is the optimal solution of the following optimization problem:

$$\min_{W_{\mathrm{DW}}=(W,f_p)\in\mathcal{W}_{\mathrm{DW}}} w\left(W,\boldsymbol{P},f_p\right) \tag{B.4}$$

We transform the problem into a shortest path problem on a NewTEG that is explained later.

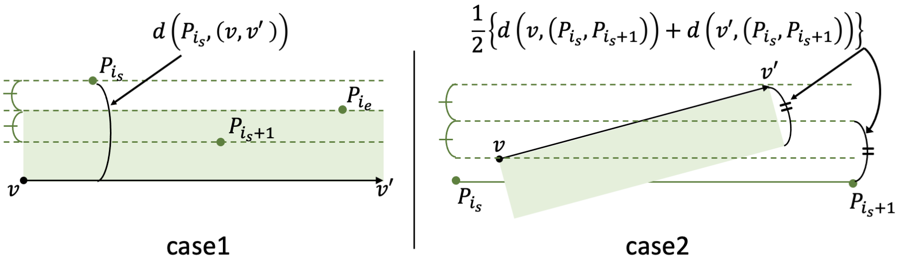

FIGURE B.2: An *arc* are is drawn as a green area for each case.

B.4 Algorithm calculating a detailed walk area

Algorithm 4 describes the way to calculate a *detailed walk area*, and this section provides an overview and the correctness of the algorithm.

Algorithm 4 calculates an *arc area* for the first to last arc in a walk W (line 4 or 9) and finally summing up these *arc areas* (line 20). We calculate an *arc area* based on the relationship between an *arc* and a *GPS projection* that is classified into case1 and case2 (see Section B.3). In Algorithm 4, the functions "calculate_nonskip_area" and "calculate_skip_area" correspond to case1 and case2, respectively.

Now we prove the correctness of Algorithm 4 by focusing loop invariant and termination. Loop invariants, conditions that satisfy at any iteration, are listed below.

- calculate_nonskip_area
  - At the start of the function (line 3), the following proposition is always true.

$$f_p(P_{i_s}) = a_j \wedge \left(i_s > 1 \Rightarrow f_p\left(P_{i_s-1}\right) \neq a_j\right) \tag{B.5}$$

  - At the calculation of an *arc area* (line 4), Equation B.1 always satisfies.
- calculate_skip_area
  - At the right after line 7, the following proposition is always true.

$$\text{Equation B.2 is true for } j \leq \forall j' < \min\left\{k \in \{j,\ldots,m\} \mid f_p(P_{i_s+1}) = a_k\right\} \tag{B.6}$$

    , which implies that Equation B.2 is always true at the calculation of an *arc area* (line 9).

We show that the above invariants are true at any iteration. Consider the first iteration of calculate_nonskip_area. Then, $i_s = 1$ and $j = 1$. Because $f_p(P_1) = a_1$, Equation B.5 satisfies. From the definition of i_e at line 3, Equation B.1 is true for any iteration if Equation B.5 satisfies at line 3. Next, we prove Equation B.6 for any iteration. Because "calculate_skip_area" is called after "calculate_nonskip_area", $f_p\left(P_{i_e}\right) = a_j \wedge f_p\left(P_{i_e+1}\right) \neq a_j$ is always true at the right before line 6; hence

$$f_p\left(P_{i_s}\right) = a_{j-1} \wedge f_p\left(P_{i_s+1}\right) \neq a_{j-1} \tag{B.7}$$

is true at the right after line 7, which implies Equation B.6. Finally, we show that Equation B.5 satisfies at the 2nd or more than iteration. From Equation B.7 and j is nondecreasing from line 8 to 11, $f_p(P_{i_s-1}) \neq a_j$ at the end of "calculate_skip_area". Obviously, $f_p(P_{i_s}) = a_j$ is true at the end of "calculate_skip_area". Because "calculate_nonskip_area" is called after "calculate_skip_area" except for the first iteration, Equation B.5 is true at the 2nd or more than iteration.

Because we have proved that (1) Equation B.1 is always true at line 4; (2) Equation B.2 is always true at line 9, any *arc area* is correctly calculated. Moreover, the right after line 19, we have calculated $w\left(a_j, \boldsymbol{P}, f_p\right)$ for $j = 1, \ldots, m$, the correctness of Algorithm 4 is verified.

Algorithm 4: *calculate detailed walk area*$(\boldsymbol{P}, W_{\mathrm{DW}})$

Input : a *vehicle trajectory* $\boldsymbol{P} = (P_1, \ldots, P_n)$; a *detailed walk* $W_{\mathrm{DW}} = (W, f_p)$ where $W = (a_1, \ldots, a_m)$ and $f_p : \boldsymbol{P} \rightarrow A$.

Output: the *detailed walk area* $w(W, \boldsymbol{P}, f_p)$

1

2 **Function** `calculate_nonskip_area`$(\boldsymbol{P}, W_{\mathrm{DW}}, i_s, j)$:

3 $\quad i_e \leftarrow \max\{i \mid i_s \leq i \leq n, f_p\left(P_{i'}\right) = a_j \ (i_s \leq \forall i' \leq i)\}$

4 $\quad w\left(a_j, \boldsymbol{P}, f_p\right) \leftarrow \ell_{a_j} \cdot \frac{1}{i_e - i_s + 1} \sum_{i=i_s}^{i_e} d\left(P_i, a_j\right)$

5 **Function** `calculate_skip_area`$(\boldsymbol{P}, W_{\mathrm{DW}}, i_e, j)$:

6 $\quad i_s \leftarrow i_e$

7 $\quad j \leftarrow j + 1$

8 $\quad$ **while** $f_p(P_{i_{s+1}}) \neq a_j$ **do**

9 $\quad\quad w\left(a_j, \boldsymbol{P}, f_p\right) \leftarrow \ell_{a_j} \cdot \frac{1}{2}\left\{d\left(\mathrm{tail}\left(a_j\right), \left(P_{i_s}, P_{i_s+1}\right)\right) + d\left(\mathrm{head}\left(a_j\right), \left(P_{i_s}, P_{i_s+1}\right)\right)\right\}$

10 $\quad\quad j \leftarrow j + 1$

11 $\quad i_s \leftarrow i_s + 1$

12 **Function** `calculate_detailedwalk_area`$(\boldsymbol{P}, W_{\mathrm{DW}})$:

13 $\quad i_s \leftarrow 1$

14 $\quad j \leftarrow 1$

15 $\quad$ **while** ture **do**

16 $\quad\quad$ `calculate_nonskip_area` $(\boldsymbol{P}, W_{\mathrm{DW}}, i_s, j)$

17 $\quad\quad$ **if** $j = m$ **then**

18 $\quad\quad\quad$ **break**

19 $\quad\quad$ `calculate_skip_area` $(\boldsymbol{P}, W_{\mathrm{DW}}, i_e, j)$

20 $\quad S \leftarrow \sum_{j=1}^{m} w\left(a_j, \boldsymbol{P}, f_p\right)$

21 $\quad$ **return** S

B.5 Definition of NewTEG

This section defines a NewTEG that is a directed graph. Any *detailed walk* is paired with exactly one walk from s to t on a NewTEG, and the wight of the walk on the NewTEG equals to the *detailed walk area*, where s and t are special nodes on the NewTEG. The bijection is defined in Section B.6.

Let $G = (V, A)$ be a *road network*, $\boldsymbol{P} = (P_i)_{i=1}^{n}$ be a *vehicle trajectory*. The parameters of NewTEG are as follows.

- $h \geq 1$: maximum number of GPS points to be acquired when passing through one *arc*.
- r_{GPS} : maximum GPS error.

Then, a NewTEG $G(\boldsymbol{P}) = (U(\boldsymbol{P}), E(\boldsymbol{P}))$ is defines as follows. For time stamp $i \in \{1, \ldots, n-1\}$, U^i is defined as the set of *arcs* where a vehicle may exist from time stamp i to $i+1$ (*arcs* near P_i or P_{i+1}):

$$U^i := \left\{ a^i \mid a \in A_{r_{\text{GPS}}, \ell_{\max}}(P_i, P_{i+1}) \right\} \tag{B.8}$$

where $A_{r_{\text{GPS}}, \ell_{\max}}(P_i, P_{i+1}) \subset A$ is the *neighborhood arcs* of P_i and P_{i+1} defined in the last paragraph of this section. For any notation x, x^i is the copy of x related to the time stamp i, and the superscript i is used for distinguishing copies related to different time stamps. E_i $(i \in \{1, 2, \ldots, n-1\})$ denotes a transition from one *arc* to another from time stamp i to $i+1$ and is defined as:

$$E^i := \{(a^i, b^i) \in U_i \times U_i \mid \text{head}(a) = \text{tail}(b)\} \tag{B.9}$$

. $(a^i, b^i) \in E^i$ means that a vehicle pass through the *arc* a from time stamp i to $i+1$, that is, there exists j such that

$$\left(f_{\text{idx}}\left(f_P(P_i)\right) < j < f_{\text{idx}}\left(f_P(P_{i+1})\right)\right) \wedge \left(a_j = a\right) \wedge \left(a_j \text{ is in the walk on a \textit{detailed walk area}}\right) \tag{B.10}$$

. Based on the definition of an *arc area*, the the weight of $e^i = \left(a^i, b^i\right) \in E^i$ is defined as follows:

$$w\left(e^i\right) := \ell_a \cdot \frac{1}{2}\left\{d\left(\text{tail}(a), (P_i, P_{i+1})\right) + d\left(\text{head}(a), (P_i, P_{i+1})\right)\right\} \tag{B.11}$$

$$\tag{B.12}$$

. For a convenience, we define $\widetilde{U^i}$ as the copy of U^i $(i \in \{1, \ldots, n-1\})$;

$$\widetilde{U^i} := \left\{ \widetilde{a^i} \mid a^i \in U^i \right\} \tag{B.13}$$

. For $\forall i \in \{1, \ldots, n-2\}$ and $i+1 \leq \forall i' \leq \min\{i+h, n-1\}$, $E_{LtL}^{i,i'}$ denotes an *arc* where a vehicle exists from time stamp $i+1$ to i':

$$E_{LtL}^{i,i'} := \left\{ (a^i, \widetilde{a^{i'}}) \in U^i \times \widetilde{U^{i'}} \right\} \tag{B.14}$$

. $\left(a^i, \widetilde{a^{i'}}\right) \in E_{LtL}^{i,i'}$ represents that

$$f_p(P_i) \neq a \wedge f_p(P_k) = a \ (i+1 \leq \forall k \leq i') \wedge f_p(P_{i'+1}) \neq a \tag{B.15}$$

; hence the the weight of $\left(a^i, \widetilde{a^{i'}}\right) \in E_{LtL}^{i,i'}$ is defined as follows:

$$w\left(\left(a^i, \widetilde{a^{i'}}\right)\right) := \ell_a \cdot \frac{1}{i'-i} \sum_{k=i+1}^{i'} d\left(P_k, a\right) \tag{B.16}$$

. For $\forall i \in \{1, \ldots, n-1\}$, we define $\widetilde{E^i}$ to ensure the connectivity of $\widetilde{U^i}$:

$$\widetilde{E^i} := \left\{(\widetilde{a^i}, b^i) \in \widetilde{U^i} \times U^i \mid \text{head}(a) = \text{tail}(b)\right\} \tag{B.17}$$

. Then, the weight of $\widetilde{e^i} \in \widetilde{E^i}$ is defined as $w\left(\widetilde{e^i}\right) = 0$. We add s, the source node of any walk, and define E_{source}^i $(1 \leq \forall i \leq h)$ that represents *arcs* where a vehicle exists from a time stamp 1 to i:

$$E_{\text{source}}^i := \left\{\left(s, \widetilde{a^i}\right) \mid \widetilde{a^i} \in \widetilde{U^i}, a \in A_{r_{\text{GPS}}, \ell_{\max}}(P_k) \ (1 \leq \forall k \leq i)\right\} \tag{B.18}$$

$$E_{\text{source}} := \bigcup_{1 \leq i \leq h} E_{\text{source}}^i \tag{B.19}$$

, where $A_{r_{\text{GPS}}, \ell_{\max}}(P_k) \subset A$ is the *neighborhood arcs* of P_k defined in the last paragraph of this section. $\left(s, \widetilde{a^i}\right) \in E_{\text{source}}^i$ represents that

$$f_p(P_k) = a \ (1 \leq \forall k \leq i) \wedge f_p(P_{i+1}) \neq a \tag{B.20}$$

. Hence, the wight of $\left(s, \widetilde{a^i}\right) \in E_{\text{source}}^i$ is defined as:

$$w\left(\left(s, \widetilde{a^i}\right)\right) := \ell_a \cdot \frac{1}{i} \sum_{i'=1}^{i} d\left(P_{i'}, a\right) \tag{B.21}$$

. Similarly, we add the sink node t and E_{sink}^i $(n-h \leq \forall i \leq n-1)$ that is the set of *arcs* where a vehicle exists from $i+1$ to n:

$$E_{\text{sink}}^i := \left\{\left(a^i, t\right) \mid a^i \in U^i, a \in A_{r_{\text{GPS}}, \ell_{\max}}(P_k) \ (i+1 \leq \forall k \leq n)\right\} \tag{B.22}$$

$$E_{\text{sink}} := \bigcup_{n-h \leq i \leq n-1} E_{\text{sink}}^i \tag{B.23}$$

. $\left(a^i, t\right) \in E_{\text{sink}}^i$ represents that

$$f_p(P_i) \neq a \wedge f_p(P_k) = a \ (i+1 \leq \forall k \leq n) \tag{B.24}$$

. Therefore, the wight of $\left(a^i, t\right) \in E_{\text{sink}}^i$ is defined as:

$$w\left(\left(a^i, t\right)\right) := \ell_a \cdot \frac{1}{n-i} \sum_{i'=i+1}^{n} d\left(P_{i'}, a\right) \tag{B.25}$$

. Finally, we define U^* and E^* to represent the case that a vehicle always exists an *arc* $a \in A$:

$$U^* := \{a^* \mid a \in A_{r_{\mathrm{GPS}},\ell_{\max}}(P_i)\ (1 \le \forall i \le n)\} \tag{B.26}$$

$$E^* := \bigcup_{a^* \in U^*} \{(s, a^*), (a^*, t)\} \tag{B.27}$$

. The weight of $(s, a^*), (a^*, t) \in E^*$ is defines as

$$w\left((s, a^*)\right) := \ell_a \cdot \frac{1}{n} \sum_{i=1}^{n} d\left(P_i, a\right) \tag{B.28}$$

$$w\left((a^*, t)\right) := 0 \tag{B.29}$$

. Utilizing these symbols, NewTEG $G(\boldsymbol{P}) = (U(\boldsymbol{P}), E(\boldsymbol{P}))$ is defines as

$$U(\boldsymbol{P}) := \left(\bigcup_{i=1}^{n-1} U^i \cup \tilde{U}^i \right) \cup \{s, t\} \cup U^* \tag{B.30}$$

$$E(\boldsymbol{P}) := \left(\bigcup_{i=1}^{n-1} E^i \cup \tilde{E}^i \right) \cup \left(\bigcup_{i=1}^{n-2} \bigcup_{i'=i+1}^{\min\{i+h, n-1\}} E_{LtL}^{i,i'} \right) \cup E_{\mathrm{source}} \cup E_{\mathrm{sink}} \cup E^* \tag{B.31}$$

. A walk from s to t is named as a s-t walk, and $\mathcal{W}_{\mathrm{TEG}}$ denotes the set of s-t walks on a NewTEG. For $W_{\mathrm{TEG}} \in \mathcal{W}_{\mathrm{TEG}}$, the weight of W_{TEG} is defined as the total weight of arcs contained in W_{TEG} and is denoted by $w\left(W_{\mathrm{TEG}}\right)$. Then, there exists a bijection $f_{\mathrm{DW}} : \mathcal{W}_{\mathrm{TEG}} \to \mathcal{W}_{\mathrm{DW}}$ such that $w\left(W, \boldsymbol{P}, f_p\right) = w(W_{\mathrm{TEG}})$ $(\forall W_{\mathrm{TEG}} \in \mathcal{W}_{\mathrm{TEG}})$, where $(W, f_p) = f_{\mathrm{DW}}(W_{\mathrm{TEG}})$, which implies that the shortest path on the NewTEG corresponds to the *detailed walk* with the minimum *detailed walk area*. Therefore, the output of NewTEG-matching is the *detailed walk* corresponding to the shortest path on NewTEG. Section B.6 proves the existence of the bijection. Figure B.3 illustrates a *detailed walk* W_{DW} and the corresponding s-t walk W_{TEG} on a NewTEG. If a *detailed walk* $W_{\mathrm{DW}} = (W, f_p)$ is

$$W = (a_{12}, a_{23}, a_{34}, a_{45}) \tag{B.32}$$

$$f_p = \begin{cases} f_p(P_1) & = a_{12} \\ f_p(P_i) & = a_{34} \quad (2 \le i \le 3) \\ f_p(P_4) & = a_{45} \end{cases} \tag{B.33}$$

, the corresponding s-t walk on the NewTEG is

$$W_{\mathrm{TEG}} = \left(\left(s, \widetilde{a_{12}^1}\right), \left(\widetilde{a_{12}^1}, a_{23}^1\right), \left(a_{23}^1, a_{34}^1\right), \left(a_{34}^1, \widetilde{a_{34}^3}\right), \left(\widetilde{a_{34}^3}, a_{45}^3\right), \left(a_{45}^3, t\right) \right) \tag{B.34}$$

, where we define $a_{ij} := (v_i, v_j) \in A$.

We define the *neighborhood arcs* $A_{r_{\mathrm{GPS}},\ell_{\max}}(P)$ and $A_{r_{\mathrm{GPS}},\ell_{\max}}(P, P')$ that appear in this section. $\ell_{\max}$ is the maximum length of an *arc*, and r_{GPS} is the maximum GPS error. We define $A_{r_{\mathrm{GPS}},\ell_{\max}}(P)$ as the set of *arcs* that intersects the minimum square required to obtain all the *arcs* within r_{GPS} from P:

$$A_{r_{\mathrm{GPS}},\ell_{\max}}(P) := \{a = (u, v) \in A \mid d_\infty(u, P) \le c \vee d_\infty(v, P) \le c\} \tag{B.35}$$

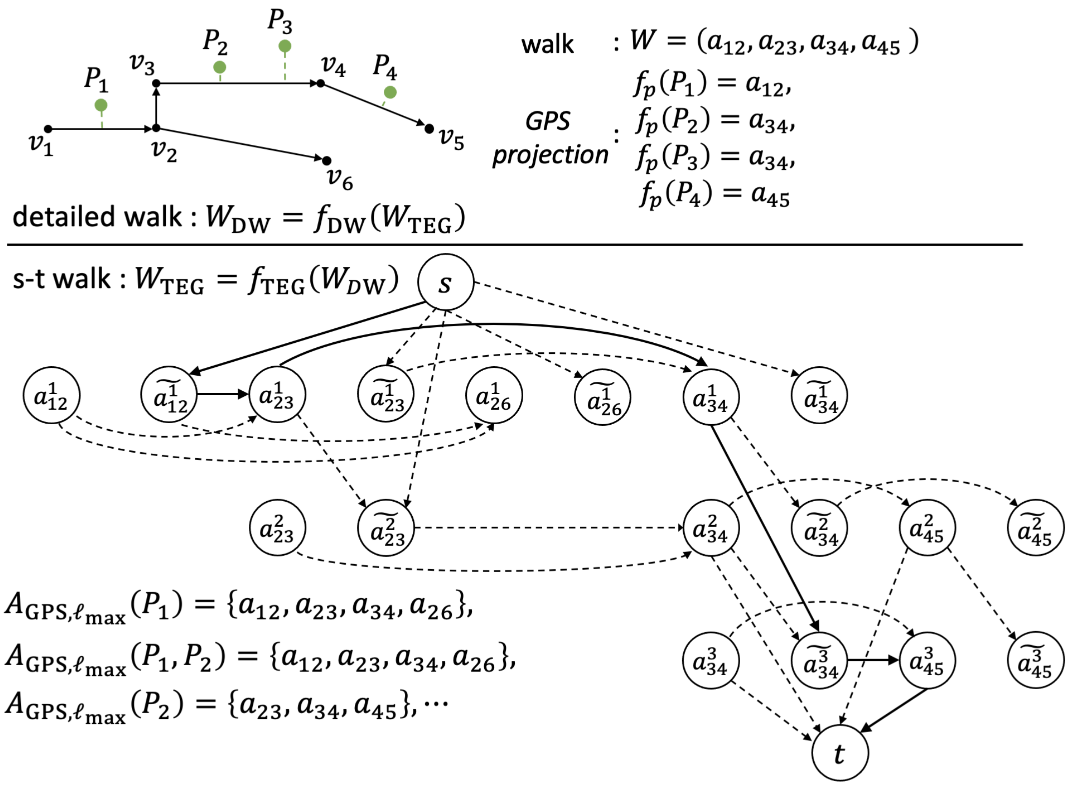

FIGURE B.3: Example of a *detailed walk* W_{DW} and the corresponding s-t walk W_{TEG} on a NewTEG. s-t walk on the NewTEG is drawn as bold arrows, and the other edges are drawn as dotted arrows. The parameter h of the NewTEG is two, and $a_{ij} \in A$ is defined as $a_{ij} := (v_i, v_j)$.

, where $c = \frac{\ell_{max}+2r_{GPS}}{2\sqrt{2}}$ if $\ell_{max} \leq 2\left(1+\sqrt{2}\right) r_{GPS}$ $\left(\text{otherwise } c = \frac{\ell_{max}}{2}\right)$. We note that

$$d\left((u,v),P\right) \leq r_{GPS} \Rightarrow d_\infty(u,P) \leq c \vee d_\infty(v,P) \leq c \Rightarrow (u,v) \in A_{r_{GPS},\ell_{max}}(P) \quad \text{(B.36)}$$

satisfies because of Theorem 2. $A_{r_{GPS},\ell_{max}}(P,P')$ denotes the set of *arcs* intersects the minimum square required to obtain all the *arcs* within $r_{GPS} + \frac{d(P,P')}{2}$ from $\frac{P+P'}{2}$:

$$A_{r_{GPS},\ell_{max}}\left(P,P'\right) := \left\{a = (u,v) \in A \mid d_\infty\left(u, \frac{P+P'}{2}\right) \leq c \vee d_\infty\left(v, \frac{P+P'}{2}\right) \leq c\right\} \quad \text{(B.37)}$$

, where $c = \frac{\ell_{max}+2r'}{2\sqrt{2}}$ if $\ell_{max} \leq 2\left(1+\sqrt{2}\right) r'$ $\left(\text{otherwise } c = \frac{\ell_{max}}{2}\right)$ and $r' = r_{GPS} + \frac{d(P,P')}{2}$.

B.6 Bijection between detailed walks and walks on NewTEG

This section creates a bijection $f_{DW} : \mathcal{W}_{TEG} \to \mathcal{W}_{DW}$ such that

$$w\left(W, \boldsymbol{P}, f_p\right) = w(W_{TEG}) \ (\forall W_{TEG} \in \mathcal{W}_{TEG}) \quad \text{(B.38)}$$

, where $(W, f_p) = f_{\mathrm{DW}}(W_{\mathrm{TEG}})$ and $\boldsymbol{P} = (P_i)_{i=1}^n$ is a *vehicle trajectory*. We first present some lemmas needed to prove Equation B.38 and finally obtain the output of NewTEG-matching by utilizing f_{DW}.

Lemma 3. *For any s-t walk on a NewTEG, there exists the corresponding* detailed walk *on a* road network*. The mapping is denoted by* $f_{\mathrm{DW}} : \mathcal{W}_{\mathrm{TEG}} \to \mathcal{W}_{\mathrm{DW}}$.

Proof. Let $\boldsymbol{P} = (P_i)_{i=1}^n$ be a *vehicle trajectory* and W_{TEG} be a s-t walk on the NewTEG. If $W_{\mathrm{TEG}} = ((s, a^*), (a^*, t))$ for an *arc* a, a vehicle exists the *arc* a from time stamp 1 to n; hence the corresponding *detailed walk* $\mathrm{W}_{\mathrm{DW}} = (W, f_p)$ is $W = (a)$ and $f_p(P_i) = a$ $(1 \le \forall i \le n)$. In the other case, W_{TEG} is expressed by

$$
\begin{aligned}
W_{\mathrm{TEG}} = &\left(e_s, \widetilde{e^{i_1}}, e_0^{i_1}, \ldots, e_{n_{i_1}-1}^{i_1}, e_{n_{i_1}}^{i_1}\right) \\
&+ \left(e_{\mathrm{LtL}}^{i_1,i_2}, \widetilde{e^{i_2}}, e_0^{i_2}, \ldots, e_{n_{i_2}}^{i_2}\right) \\
&+ \ldots \\
&+ \left(e_{\mathrm{LtL}}^{i_{K-1},i_K}, \widetilde{e^{i_K}}, \ldots, e_{n_{i_K}}^{i_K}, e_t\right) \\
= &\left(\left(s, \widetilde{a_{i_1}^{i_1}}\right), \left(\widetilde{a_{i_1}^{i_1}}, a_{i_1,0}^{i_1}\right), \left(a_{i_1,0}^{i_1}, a_{i_1,1}^{i_1}\right), \ldots, \left(a_{i_1,n_{i_1}-1}^{i_1}, a_{i_1,n_{i_1}}^{i_1}\right), \left(a_{i_1,n_{i_1}}^{i_1}, a_{i_2}^{i_1}\right)\right) \\
&+ \left(\left(a_{i_2}^{i_1}, \widetilde{a_{i_2}^{i_2}}\right), \left(\widetilde{a_{i_2}^{i_2}}, a_{i_2,0}^{i_2}\right), \left(a_{i_2,0}^{i_2}, a_{i_2,1}^{i_2}\right), \ldots, \left(a_{i_2,n_{i_2}}^{i_2}, a_{i_3}^{i_2}\right)\right) \\
&+ \ldots \\
&+ \left(\left(a_{i_K}^{i_{K-1}}, \widetilde{a_{i_K}^{i_K}}\right), \left(\widetilde{a_{i_K}^{i_K}}, a_{i_K,0}^{i_K}\right), \ldots, \left(a_{i_K,n_{i_K}}^{i_K}, a_n^{i_K}\right), \left(a_n^{i_K}, t\right)\right)
\end{aligned}
$$

, where $1 \le K \le n-1$, $1 \le i_k < i_{k+1} \le n-1$ $(1 \le \forall k \le K-1)$, $e_s = \left(s, \widetilde{a_{i_1}^{i_1}}\right) \in E_{\mathrm{source}}$, $\widetilde{e^{i_k}} = \left(\widetilde{a_{i_k}^{i_k}}, a_{i_k,0}^{i_k}\right) \in \widetilde{E^{i_k}}$ $(1 \le \forall k \le K)$, $e_j^{i_k} = \left(a_{i_k,j}^{i_k}, a_{i_k,j+1}^{i_k}\right) \in E^{i_k}$ $(1 \le \forall k \le K, 0 \le \forall j \le n_{i_k} - 1)$, $e_{n_{i_k}}^{i_k} = \left(a_{i_k,n_{i_k}}^{i_k}, a_{i_{k+1}}^{i_k}\right) \in E^{i_k}$ $(1 \le \forall k \le K-1)$, $e_{n_{i_K}}^{i_K} = \left(a_{i_K,n_{i_K}}^{i_K}, a_n^{i_K}\right) \in E^{i_K}$, $e_{\mathrm{LtL}}^{i_k,i_{k+1}} = \left(a_{i_{k+1}}^{i_k}, \widetilde{a_{i_{k+1}}^{i_{k+1}}}\right) \in E_{\mathrm{LtL}}^{i_k,i_{k+1}}$ $(1 \le k \le K-1)$, and $e_t = \left(a_n^{i_K}, t\right) \in E_{\mathrm{sink}}$. For $1 \le \forall k \le K$, n_{i_k} is either -1 or a non-negative integer. If $n_{i_k} = -1$, we define

$$
\ldots, e_{\mathrm{LtL}}^{i_{k-1},i_k}, \widetilde{e^{i_k}}, e_{\mathrm{LtL}}^{i_k,i_{k+1}}, \cdots = \ldots, \left(a_{i_k}^{i_{k-1}}, \widetilde{a_{i_k}^{i_k}}\right), \left(\widetilde{a_{i_k}^{i_k}}, a_{i_{k+1}}^{i_k}\right), \left(a_{i_{k+1}}^{i_k}, \widetilde{a_{i_{k+1}}^{i_{k+1}}}\right), \ldots \tag{B.39}
$$

as a special case. From the definition of $e_s = \left(s, \widetilde{a_{i_1}^{i_1}}\right)$ (Equation B.20), we have

$$
f_p\left(P_i\right) = a_{i_1} \ (0 \le i \le i_1) \wedge f_p\left(P_{i_1+1}\right) \ne a_{i_1} \tag{B.40}
$$

. The definition of $e_j^{i_1} = \left(a_{i_1,j}^{i_1}, a_{i_1,j+1}^{i_1}\right) \in E^{i_1}$ (Equation B.10) indicates that all the *arcs* where a vehicle pass through from the right after i_1 to the right before $i_1 + 1$ are $a_{i_1,0}, a_{i_1,1}, \ldots, a_{i_1,n_{i_1}}$ in chronological order. Besides, $e_{\mathrm{LtL}}^{i_1,i_2} = \left(a_{i_2}^{i_1}, \widetilde{a_{i_2}^{i_2}}\right) \in E_{\mathrm{LtL}}^{i_1,i_2}$ means that

$$f_p\left(P_i\right) = a_{i_2}\ \left(i_1+1 \le i \le i_2\right) \wedge f_p(P_{i_2+1}) \neq a_{i_2} \tag{B.41}$$

, as mentioned Equation B.15. We apply a similar discussion to i_k $(2 \le k \le K)$ and finally have

$$f_p\left(P_i\right) = a_n \quad \left(i_K+1 \le i \le n\right) \tag{B.42}$$

from the definition of $e_t = \left(a_n^{i_K}, t\right)$ (see Equation B.24). In summary, the corresponding *detailed walk* $W_{\mathrm{DW}} = (W, f_p)$ is as follows:

$$W := \left(a_{i_1}, a_{i_1,0}, a_{i_1,1}, \ldots, a_{i_1,n_{i_1}}, a_{i_2}, a_{i_2,0}, \ldots, a_{i_K,n_{i_K}}, a_n\right) \tag{B.43}$$

$$f_p\left(P_i\right) := \begin{cases} a_{i_1} & \left(0 \le i \le i_1\right) \\ a_{i_{k+1}} & \left(1 \le k \le K-1, i_k+1 \le i \le i_{k+1}\right) \\ a_n & \left(i_K+1 \le i \le n\right) \end{cases} \tag{B.44}$$

. □

Lemma 4. *For any* detailed walk *on a* road network, *there exits the corresponding s-t walk on a NewTEG. The mapping is denoted by $f_{\mathrm{TEG}} : \mathcal{W}_{\mathrm{DW}} \to \mathcal{W}_{\mathrm{TEG}}$.*

Proof. Let $\boldsymbol{P} = (P_i)_{i=1}^n$ be a *vehicle trajectory* and $W_{\mathrm{DW}} = (W, f_p)$ be a *detailed walk* on a *road network*. If $W = (a)$ for an *arc* a, $f_p(P_i) = a$ $(1 \le \forall i \le n)$; hence, the corresponding s-t walk on the NewTEG is $W_{\mathrm{TEG}} = ((s, a^*), (a^*, t))$. In the other case, W_{DW} is expressed by

$$W = \left(a_{i_1}, a_{i_1,0}, a_{i_1,1}, \ldots, a_{i_1,n_{i_1}}, a_{i_2}, a_{i_2,0}, \ldots, a_{i_2,n_{i_2}}, \ldots, a_{i_K,n_{i_K}}, a_n\right) \tag{B.45}$$

$$P_r\left(P_i\right) = \begin{cases} a_{i_1} & \left(0 \le i \le i_1\right) \\ a_{i_{k+1}} & \left(1 \le k \le K-1, i_k+1 \le i \le i_{k+1}\right) \\ a_n & \left(i_K+1 \le i \le n\right) \end{cases} \tag{B.46}$$

, where $1 \le K \le n-1$ and $1 \le i_k < i_{k+1} \le n-1$ $(1 \le \forall k \le K-1)$. For $1 \le \forall k \le K$, n_{i_k} is either -1 or a non-negative integer. If $n_{i_k} = -1$, we define

$$W = \left(\ldots, a_{i_{k-1},n_{i_{k-1}}}, a_{i_k}, a_{i_{k+1}}, \ldots\right) \tag{B.47}$$

as a special case. Then, the corresponding s-t walk W_{TEG} is

$$W_{\mathrm{TEG}} = \left(\left(s, \widetilde{a_{i_1}^{i_1}}\right), \left(\widetilde{a_{i_1}^{i_1}}, a_{i_1,0}^{i_1}\right), \left(a_{i_1,0}^{i_1}, a_{i_1,1}^{i_1}\right), \ldots, \left(a_{i_1,n_{i_1}-1}^{i_1}, a_{i_1,n_{i_1}}^{i_1}\right), \left(a_{i_1,n_{i_1}}^{i_1}, a_{i_2}^{i_1}\right)\right) \tag{B.48}$$

$$+ \left(\left(a_{i_2}^{i_1}, \widetilde{a_{i_2}^{i_2}}\right), \left(\widetilde{a_{i_2}^{i_2}}, a_{i_2,0}^{i_2}\right), \left(a_{i_2,0}^{i_2}, a_{i_2,1}^{i_2}\right), \ldots, \left(a_{i_2,n_{i_2}}^{i_2}, a_{i_3}^{i_2}\right)\right) \tag{B.49}$$

$$+ \ldots \tag{B.50}$$

$$+ \left(\left(a_{i_K}^{i_{K-1}}, \widetilde{a_{i_K}^{i_K}}\right), \left(\widetilde{a_{i_K}^{i_K}}, a_{i_K,0}^{i_K}\right), \ldots, \left(a_{i_K,n_{i_K}}^{i_K}, a_n^{i_K}\right), \left(a_n^{i_K}, t\right)\right) \tag{B.51}$$

. We note that $\left(s, \widetilde{a_{i_1}^{i_1}}\right) \in E_{\text{source}}$, $\left(\widetilde{a_{i_k}^{i_k}}, a_{i_k,0}^{i_k}\right) \in \widetilde{E^{i_k}}$ $(1 \leq \forall k \leq K)$, $\left(a_{i_k,j}^{i_k}, a_{i_k,j+1}^{i_k}\right) \in E^{i_k}$ $(1 \leq \forall k \leq K, 0 \leq \forall j \leq n_{i_k} - 1)$, $\left(a_{i_k,n_{i_k}}^{i_k}, a_{i_{k+1}}^{i_k}\right) \in E^{i_k}$ $(1 \leq \forall k \leq K-1)$, $\left(a_{i_{k+1}}^{i_k}, \widetilde{a_{i_{k+1}}^{i_{k+1}}}\right) \in E_{LtL}^{i_k,i_{k+1}}$ $(1 \leq \forall k \leq K-1)$, $\left(a_{i_K,n_{i_K}}^{i_K}, a_n^{i_K}\right) \in E^{i_K}$, and $\left(a_n^{i_K}, t\right) \in E_{\text{sink}}$. $\square$

Lemma 5. *Let* $W_{DW} = (W, f_p)$ *be a* detailed walk *on a* road network *and* $f_{TEG}(W_{DW}) = W_{TEG}$ *be the corresponding s-t walk on a NewTEG. Then,* $w\left(W, \boldsymbol{P}, f_p\right) = w(W_{TEG})$*, where* $\boldsymbol{P} = (P_i)_{i=1}^n$ *is a* vehicle trajectory.

Proof. If $W = (a)$ for an *arc* a, $f_p(P_i) = a$ $(1 \leq \forall i \leq n)$; hence $W_{\text{TEG}} = ((s, a^*), (a^*, t))$ from Lemma 4. Based on the definition of the weight on the NewTEG,

$$w(W_{\text{TEG}}) = w((s, a^*)) + w((a^*, t)) = \ell_a \cdot \frac{1}{n} \sum_{i=1}^{n} d(P_i, a) \tag{B.52}$$

. On the other hand, if we calculate the *detailed walk area* using Algorithm 4, we have

$$w\left(W, \boldsymbol{P}, f_p\right) = \ell_a \cdot \frac{1}{n-1+1} \sum_{i=1}^{n} d(P_i, a) \tag{B.53}$$

, which implies $w\left(W, \boldsymbol{P}, f_p\right) = w(W_{\text{TEG}})$. If $W \neq (a)$, $W_{\text{DW}} = (W, f_p)$ is expressed as

$$W = \left(a_{i_1}, a_{i_1,0}, a_{i_1,1}, \ldots, a_{i_1,n_{i_1}}, a_{i_2}, a_{i_2,0}, \ldots, a_{i_2,n_{i_2}}, \ldots, a_{i_K,n_{i_K}}, a_n\right) \tag{B.54}$$

$$f_p(P_i) = \begin{cases} a_{i_1} & (1 \leq i \leq i_1) \\ a_{i_{k+1}} & (1 \leq k \leq K-1, i_k + 1 \leq i \leq i_{k+1}) \\ a_n & (i_K + 1 \leq i \leq n) \end{cases} \tag{B.55}$$

, where $1 \leq K \leq n-1$ and $1 \leq i_k < i_{k+1} \leq n-1$ $(1 \leq \forall k \leq K-1)$. For $1 \leq \forall k \leq K$, n_{i_k} is either -1 or a non-negative integer. If $n_{i_k} = -1$, we define

$$W = \left(\ldots, a_{i_{k-1},n_{i_{k-1}}}, a_{i_k}, a_{i_{k+1}}, \ldots\right) \tag{B.56}$$

as a special case. Then, utilizing Algorithm 4, we have

$$
\begin{aligned}
w\left(W,\boldsymbol{P},P_r\right) =& \ell_{a_{i_1}} \cdot \frac{1}{i_1}\sum_{i=1}^{i_1} d(P_i,a_{i_1}) \\
&+ \sum_{j=0}^{n_{i_1}} \ell_{a_{i_1,j}} \cdot \frac{1}{2}\left\{d\left(\text{tail}\left(a_{i_1,j}\right),\left(P_{i_1},P_{i_1+1}\right)\right) + d\left(\text{head}\left(a_{i_1,j}\right),\left(P_{i_1},P_{i_1+1}\right)\right)\right\} \\
&+ \ell_{a_{i_2}} \cdot \frac{1}{i_2-i_1}\sum_{i=i_1+1}^{i_2} d(P_i,a_{i_2}) \\
&+ \cdots \\
&+ \sum_{j=0}^{n_{i_K}} \ell_{a_{i_K,j}} \cdot \frac{1}{2}\left\{d\left(\text{tail}\left(a_{i_K,j}\right),\left(P_{i_K},P_{i_K+1}\right)\right) + d\left(\text{head}\left(a_{i_K,j}\right),\left(P_{i_K},P_{i_K+1}\right)\right)\right\} \\
&+ \ell_{a_n} \cdot \frac{1}{n-i_K}\sum_{i=i_K+1}^{n} d(P_i,a_n) \qquad \text{(B.57)}\\
=& \sum_{k=0}^{K}\left\{\ell_{a_{i_{k+1}}} \cdot \frac{1}{i_{k+1}-i_k}\sum_{i=i_k+1}^{i_{k+1}} d\left(P_i,a_{i_{k+1}}\right)\right\} \\
&+ \sum_{k=1}^{K}\sum_{j=0}^{n_{i_k}}\left\{\ell_{a_{i_k,j}} \cdot \frac{1}{2}\left\{d\left(\text{tail}\left(a_{i_k,j}\right),\left(P_{i_k},P_{i_k+1}\right)\right) + d\left(\text{head}\left(a_{i_k,j}\right),\left(P_{i_k},P_{i_k+1}\right)\right)\right\}\right\} \qquad \text{(B.58)}
\end{aligned}
$$

, where $i_0 := 0$ and $i_{K+1} := n$ for a convenience. In contrast, based on Lemma 4, the corresponding s-w walk on the NewTEG is denoted by

$$
\begin{aligned}
W_{\text{TEG}} =& \left(\left(s,\widetilde{a^{i_1}_{i_1}}\right),\left(\widetilde{a^{i_1}_{i_1}},a^{i_1}_{i_1,0}\right),\left(a^{i_1}_{i_1,0},a^{i_1}_{i_1,1}\right),\ldots,\left(a^{i_1}_{i_1,n_{i_1}-1},a^{i_1}_{i_1,n_{i_1}}\right),\left(a^{i_1}_{i_1,n_{i_1}},a^{i_1}_{i_2}\right)\right) \\
&+ \left(\left(a^{i_1}_{i_2},\widetilde{a^{i_2}_{i_2}}\right),\left(\widetilde{a^{i_2}_{i_2}},a^{i_2}_{i_2,0}\right),\left(a^{i_2}_{i_2,0},a^{i_2}_{i_2,1}\right),\ldots,\left(a^{i_2}_{i_2,n_{i_2}},a^{i_2}_{i_3}\right)\right) \\
&+ \ldots \\
&+ \left(\left(a^{i_{K-1}}_{i_K},\widetilde{a^{i_K}_{i_K}}\right),\left(\widetilde{a^{i_K}_{i_K}},a^{i_K}_{i_K,0}\right),\ldots,\left(a^{i_K}_{i_K,n_{i_K}},a^{i_K}_{n}\right),\left(a^{i_K}_{n},t\right)\right) \qquad \text{(B.59)}
\end{aligned}
$$

. We note that $\left(s,\widetilde{a^{i_1}_{i_1}}\right) \in E_{\text{source}}$, $\left(\widetilde{a^{i_k}_{i_k}},a^{i_k}_{i_k,0}\right) \in \widetilde{E^{i_k}}$ $(1 \le \forall k \le K)$, $\left(a^{i_k}_{i_k,j},a^{i_k}_{i_k,j+1}\right) \in E^{i_k}$ $(1 \le \forall k \le K, 0 \le \forall j \le n_{i_k}-1)$, $\left(a^{i_k}_{i_k,n_{i_k}},a^{i_k}_{i_{k+1}}\right) \in E^{i_k}$ $(1 \le \forall k \le K-1)$, $\left(a^{i_k}_{i_{k+1}},\widetilde{a^{i_{k+1}}_{i_{k+1}}}\right) \in E^{i_k,i_{k+1}}_{LtL}$ $(1 \le \forall k \le K-1)$, $\left(a^{i_K}_{i_K,n_{i_K}},a^{i_K}_{n}\right) \in E^{i_K}$, and $\left(a^{i_K}_{n},t\right) \in E_{\text{sink}}$.

According to the definition of weight on the NewTEG, we have

$$
\begin{aligned}
w\left(W_{\mathrm{TEG}}\right) =& w\left(s, \widetilde{a_{i_1}^{i_1}}\right) \\
&+ \sum_{k=1}^{K} \sum_{j=0}^{n_{i_k}-1} w\left(\left(a_{i_k,j}^{i_k}, a_{i_k,j+1}^{i_k}\right)\right) \\
&+ \sum_{k=1}^{K-1} w\left(a_{i_k,n_{i_k}}^{i_k}, a_{i_{k+1}}^{i_k}\right) \\
&+ \sum_{k=1}^{K-1} w\left(\left(a_{i_{k+1}}^{i_k}, \widetilde{a_{i_{k+1}}^{i_{k+1}}}\right)\right) \\
&+ w\left(\left(a_{i_K,n_{i_K}}^{i_K}, a_n^{i_K}\right)\right) \\
&+ w\left(\left(a_n^{i_K}, t\right)\right) \qquad \text{(B.60)} \\
=& \ell_{a_{i_1}} \cdot \frac{1}{i_1} \sum_{i=1}^{i_1} d(P_i, a_{i_1}) \\
&+ \sum_{k=1}^{K} \sum_{j=0}^{n_{i_k}-1} \ell_{a_{i_k,j}} \cdot \frac{1}{2} \left\{ d\left(\mathrm{tail}\left(a_{i_k,j}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) + d\left(\mathrm{head}\left(a_{i_k,j}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) \right\} \\
&+ \sum_{k=1}^{K-1} \ell_{a_{i_k,n_{i_k}}} \cdot \frac{1}{2} \left\{ d\left(\mathrm{tail}\left(a_{i_k,n_{i_k}}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) + d\left(\mathrm{head}\left(a_{i_k,n_{i_k}}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) \right\} \\
&+ \sum_{k=1}^{K-1} \left\{ \ell_{a_{i_{k+1}}} \cdot \frac{1}{i_{k+1} - i_k} \sum_{i=i_k+1}^{i_{k+1}} d\left(P_i, a_{i_{k+1}}\right) \right\} \\
&+ \ell_{a_{i_K,n_{i_K}}} \cdot \frac{1}{2} \left\{ d\left(\mathrm{tail}\left(a_{i_K,n_{i_K}}\right), \left(P_{i_K}, P_{i_K+1}\right)\right) + d\left(\mathrm{head}\left(a_{i_K,n_{i_K}}\right), \left(P_{i_K}, P_{i_K+1}\right)\right) \right\} \\
&+ \ell_{a_n} \cdot \frac{1}{n - i_K} \sum_{i=i_K+1}^{n} d(P_i, a_n) \qquad \text{(B.61)} \\
=& \ell_{a_{i_1}} \cdot \frac{1}{i_1} \sum_{i=1}^{i_1} d(P_i, a_{i_1}) \\
&+ \sum_{k=1}^{K-1} \left\{ \ell_{a_{i_{k+1}}} \cdot \frac{1}{i_{k+1} - i_k} \sum_{i=i_k+1}^{i_{k+1}} d\left(P_i, a_{i_{k+1}}\right) \right\} \\
&+ \ell_{a_n} \cdot \frac{1}{n - i_K} \sum_{i=i_K+1}^{n} d(P_i, a_n) \\
&+ \sum_{k=1}^{K} \sum_{j=0}^{n_{i_k}} \left\{ \ell_{a_{i_k,j}} \cdot \frac{1}{2} \left\{ d\left(\mathrm{tail}\left(a_{i_k,j}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) + d\left(\mathrm{head}\left(a_{i_k,j}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) \right\} \right\} \\
& \qquad \text{(B.62)} \\
=& \sum_{k=0}^{K} \left\{ \ell_{a_{i_{k+1}}} \cdot \frac{1}{i_{k+1} - i_k} \sum_{i=i_k+1}^{i_{k+1}} d\left(P_i, a_{i_{k+1}}\right) \right\} \\
&+ \sum_{k=1}^{K} \sum_{j=0}^{n_{i_k}} \left\{ \ell_{a_{i_k,j}} \cdot \frac{1}{2} \left\{ d\left(\mathrm{tail}\left(a_{i_k,j}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) + d\left(\mathrm{head}\left(a_{i_k,j}\right), \left(P_{i_k}, P_{i_k+1}\right)\right) \right\} \right\} \\
& \qquad \text{(B.63)} \\
=& w\left(W, \boldsymbol{P}, P_r\right) \qquad \text{(B.64)}
\end{aligned}
$$

, where we define $i_0 := 0$ and $i_{K+1} := n$ for a convenience. □

Theorem 3. *There exists a bijection from the set of* detailed walks *on a* road network *to the set of s-t walks on a NewTEG, and the corresponding* detailed walk area *equals to the weight of the corresponding walk on the NewTEG. Formally,* $f_{DW} : \mathcal{W}_{TEG} \to \mathcal{W}_{DW}$ *is bijective, and* $w\left(W, \boldsymbol{P}, f_p\right) = w(W_{TEG})$ $(\forall W_{TEG} \in \mathcal{W}_{TEG})$*, where* $(W, f_p) = f_{DW}(W_{TEG})$ *and* $\boldsymbol{P} = (P_i)_{i=1}^n$ *is a* vehicle trajectory.

Proof. The proof is complete if we show the following things:

1. $f_{\mathrm{DW}}\left(f_{\mathrm{TEG}}\left(W_{\mathrm{DW}}\right)\right) = W_{\mathrm{DW}}$ $\left(\forall W_{\mathrm{DW}} \in \mathcal{W}_{\mathrm{DW}}\right)$.
2. $f_{\mathrm{TEG}}\left(f_{\mathrm{DW}}\left(W_{\mathrm{TEG}}\right)\right) = W_{\mathrm{TEG}}$ $\left(\forall W_{\mathrm{TEG}} \in \mathcal{W}_{\mathrm{TEG}}\right)$
3. $w\left(W, \boldsymbol{P}, f_p\right) = w(W_{\mathrm{TEG}})$ $\left(\forall W_{\mathrm{TEG}} \in \mathcal{W}_{\mathrm{TEG}}\right)$, where $\left(W, f_p\right) = f_{\mathrm{DW}}(W_{\mathrm{TEG}})$

1 and 2 are directly induced from the definitions of f_{DW} and f_{TEG} (see Lemma 3 and Lemma 4). We arbitrarily take $W_{\mathrm{TEG}} \in \mathcal{W}_{\mathrm{TEG}}$. Then, utilizing Lemma 5, we have

$$w\left(W, \boldsymbol{P}, f_p\right) = w\left(f_{\mathrm{TEG}}\left(f_{\mathrm{DW}}(W_{\mathrm{TEG}})\right)\right) = w(W_{\mathrm{TEG}}) \tag{B.65}$$

, where $\left(W, f_p\right) = f_{\mathrm{DW}}(W_{\mathrm{TEG}})$. □

We obtain the output of NewTEG-matching by utilizing $f_{\mathrm{DW}} : \mathcal{W}_{\mathrm{TEG}} \to \mathcal{W}_{\mathrm{DW}}$. Given a *vehicle trajectory* $\boldsymbol{P} = (P_i)_{i=1}^n$, we have

$$\min_{W_{\mathrm{DW}}=(W,f_p)\in\mathcal{W}_{\mathrm{DW}}} w\left(W, \boldsymbol{P}, f_p\right) \tag{B.66}$$

$$= \min_{W_{\mathrm{TEG}}\in\mathcal{W}_{\mathrm{TEG}}} w\left(W_{\mathrm{TEG}}\right) \tag{B.67}$$

from Theorem 3. Because the optimal solution of Equation B.67 is the shortest path from s to t on the NewTEG, denoted by W^*_{TEG}, and if we define $\left(W^*, f^*_p\right) = f_{\mathrm{DW}}\left(W^*_{\mathrm{TEG}}\right)$, we have

$$w\left(W^*, \boldsymbol{P}, f^*_p\right) = w(W^*_{\mathrm{TEG}}) = \min_{W_{\mathrm{TEG}}\in\mathcal{W}_{\mathrm{TEG}}} w\left(W_{\mathrm{TEG}}\right) = \min_{W_{\mathrm{DW}}=(W,f_p)\in\mathcal{W}_{\mathrm{DW}}} w\left(W, \boldsymbol{P}, f_p\right) \tag{B.68}$$

. Because the output of NewTEG-matching is defined as the optimal solution of Equation B.66, $f_{\mathrm{DW}}\left(W^*_{\mathrm{TEG}}\right)$ is the output of NewTEG-matching.

Bibliography

Alt, Helmut et al. (2003). “Matching planar maps”. In: *J. Algorithms* 49.2, pp. 262–283. ISSN: 01966774. DOI: 10.1016/S0196-6774(03)00085-3.

Bentley, Jon Louis (1975). “Multidimensional Binary Search Trees Used for Associative Searching”. In: *Commun. ACM* 18.9, pp. 509–517. ISSN: 15577317. DOI: 10.1145/361002.361007.

Bonnifait, Philippe and Jean Laneurit (2009). “Multi-hypothesis Map-Matching using Particle Filtering To cite this version :” in: *16th World Congr. ITS Syst. Serv. HAL*, pp. 1–8.

Chao, Pingfu et al. (2020). “A Survey on Map-Matching Algorithms”. In: *Databases Theory Appl.* Ed. by Renata Borovica-Gajic, Jianzhong Qi, and Weiqing Wang. Cham: Springer International Publishing, pp. 121–133. ISBN: 978-3-030-39469-1.

Chazelle, Bernard and Leonidas J Guibas (1986). “Fractional cascading: I. A data structuring technique”. In: *Algorithmica* 1.1, pp. 133–162. ISSN: 1432-0541. DOI: 10.1007/BF01840440. URL: https://doi.org/10.1007/BF01840440.

Chen, Bi Yu et al. (2014). “Map-matching algorithm for large-scale low-frequency floating car data”. In: *Int. J. Geogr. Inf. Sci.* 28.1, pp. 22–38. ISSN: 13658816. DOI: 10.1080/13658816.2013.816427. URL: http://dx.doi.org/10.1080/13658816.2013.816427.

DOUGLAS, DAVID H and THOMAS K PEUCKER (1973). “ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE”. In: *Cartogr. Int. J. Geogr. Inf. Geovisualization* 10.2, pp. 112–122. ISSN: 0317-7173. DOI: 10.3138/FM57-6770-U75U-7727. URL: https://utpjournals.press/doi/10.3138/FM57-6770-U75U-7727.

Goh, C. Y. et al. (2012). “Online map-matching based on hidden Markov model for real-time traffic sensing applications”. In: *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC* 117543, pp. 776–781. DOI: 10.1109/ITSC.2012.6338627.

Gong, Yue Jiao et al. (2018). “AntMapper: An Ant Colony-Based Map Matching Approach for Trajectory-Based Applications”. In: *IEEE Trans. Intell. Transp. Syst.* 19.2, pp. 390–401. ISSN: 15249050. DOI: 10.1109/TITS.2017.2697439.

Hsueh, Yu Ling and Ho Chian Chen (2018). “Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions”. In: *Inf. Sci. (Ny).* 433-434, pp. 55–69. ISSN: 00200255. DOI: 10.1016/j.ins.2017.12.031. URL: https://doi.org/10.1016/j.ins.2017.12.031.

Hu, Gang et al. (2017). “IF-Matching: Towards Accurate Map-Matching with Information Fusion”. In: *IEEE Trans. Knowl. Data Eng.* 29.1, pp. 114–127. ISSN: 10414347. DOI: 10.1109/TKDE.2016.2617326.

Huang, Zhenfeng et al. (2021). “Survey on vehicle map matching techniques”. In: *CAAI Trans. Intell. Technol.* 6.1, pp. 55–71. ISSN: 24682322. DOI: 10.1049/cit2.12030.

Hunter, Timothy, Pieter Abbeel, and Alexandre Bayen (2014). “The path inference filter: Model-based low-latency map matching of probe vehicle data”. In: *IEEE Trans. Intell. Transp. Syst.* 15.2, pp. 507–529. ISSN: 15249050. DOI: 10.1109/TITS.2013.2282352. arXiv: 1109.1966.

Keogh, E. et al. (2002). “An online algorithm for segmenting time series”. In: pp. 289–296. DOI: 10.1109/icdm.2001.989531.

Keogh, Ej and Mj Pazzani (1998). “An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback”. In: *Kdd* 98, pp. 239–243. ISSN: <null>. DOI: 10.1.1.42.1358. URL: http://www.aaai.org/Papers/KDD/1998/KDD98-041.pdf.

Knapen, Luk et al. (2018). “Likelihood-based offline map matching of GPS recordings using global trace information”. In: *Transp. Res. Part C Emerg. Technol.* 93.June, pp. 13–35. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.05.014. URL: https://doi.org/10.1016/j.trc.2018.05.014.

Koski, Antti, Martti Juhola, and Merik Meriste (1995). “Syntactic recognition of ECG signals by attributed finite automata”. In: *Pattern Recognit.* 28.12, pp. 1927–1940. ISSN: 00313203. DOI: 10.1016/0031-3203(95)00052-6.

Kubička, Matěj et al. (2015). “Dataset for testing and training of map-matching algorithms”. In: *2015 IEEE Intell. Veh. Symp.* IEEE, pp. 1088–1093.

Luebke, D P (2001). “A Survey of Polygonal Simplification Algorithms”. In: *Comput. Graph. Appl. IEEE* 21, pp. 24–35. DOI: 10.1109/38.920624.

Newson, Paul and John Krumm (2009). “Hidden Markov map matching through noise and sparseness”. In: *GIS Proc. ACM Int. Symp. Adv. Geogr. Inf. Syst.*, pp. 336–343. DOI: 10.1145/1653771.1653818.

Park, S, D Lee, and W W Chu (1999). “Fast retrieval of similar subsequences in long sequence databases”. In: *Proc. 1999 Work. Knowl. Data Eng. Exch. (Cat. No.PR00453)*, pp. 60–67. DOI: 10.1109/KDEX.1999.836610.

Park, Sanghyun, Sang-Wook Kim, and Wesley W Chu (2001). “Segment-based Approach for Subsequence Searches in Sequence Databases”. In: *Proc. 2001 ACM Symp. Appl. Comput.* SAC ’01. New York, NY, USA: ACM, pp. 248–252. ISBN: 1-58113-287-5. DOI: 10.1145/372202.372334. URL: http://doi.acm.org/10.1145/372202.372334.

Quddus, Mohammed and Simon Washington (2015). “Shortest path and vehicle trajectory aided map-matching for low frequency GPS data”. In: *Transp. Res. Part C Emerg. Technol.* 55, pp. 328–339. ISSN: 0968090X. DOI: 10.1016/j.trc.2015.02.017. URL: http://dx.doi.org/10.1016/j.trc.2015.02.017.

Ray-Chaudhuri, D. K. (1967). “Characterization of line graphs”. In: *J. Comb. Theory* 3.3, pp. 201–214. ISSN: 00219800. DOI: 10.1016/S0021-9800(67)80068-1.

Sathiaseelan, Arjuna (2011). “The Role of Location Based Technologies in Intelligent Transportation Systems”. In: *Asian J. Inf. Technol.* 10.6, pp. 227–233. ISSN: 1682-3915. DOI: 10.3923/ajit.2011.227.233.

Taguchi, Shun, Satoshi Koide, and Takayoshi Yoshimura (2019). “Online Map Matching with Route Prediction”. In: *IEEE Trans. Intell. Transp. Syst.* 20.1, pp. 338–347. ISSN: 15249050. DOI: 10.1109/TITS.2018.2812147.

Tang, Jinjin et al. (2016). “Estimating the most likely space–time paths, dwell times and path uncertainties from vehicle trajectory data: A time geographic method”. In: *Transp. Res. Part C Emerg. Technol.* 66, pp. 176–194. ISSN: 0968-090X. DOI: 10.1016/J.TRC.2015.08.014. URL: https://www.sciencedirect.com/science/article/pii/S0968090X15003150.

Toledo-Moreo, R, D Betaille, and F Peyret (2010). “Lane-Level Integrity Provision for Navigation and Map Matching With GNSS, Dead Reckoning, and Enhanced Maps”. In: *IEEE Trans. Intell. Transp. Syst.* 11.1, pp. 100–112. ISSN: 1524-9050. DOI: 10.1109/TITS.2009.2031625.

Velaga, Nagendra R. and Kate Pangbourne (2014). “Achieving genuinely dynamic road user charging: Issues with a GNSS-based approach”. In: *J. Transp. Geogr.* 34, pp. 243–253. ISSN: 09666923. DOI: 10.1016/j.jtrangeo.2013.09.013. URL: http://dx.doi.org/10.1016/j.jtrangeo.2013.09.013.

Wang, Xuemei and Wenbo Ni (2016). "An improved particle filter and its application to an INS/GPS integrated navigation system in a serious noisy scenario". In: *Meas. Sci. Technol.* 27.9. ISSN: 13616501. DOI: 10.1088/0957-0233/27/9/095005.

Wei, Hong et al. (2013). "Map matching by Fréchet distance and global weight optimization". In: *Dep. Comput. Sci. ...*

White, Christopher E., David Bernstein, and Alain L. Kornhauser (2000). "Some map matching algorithms for personal navigation assistants". In: *Transp. Res. Part C Emerg. Technol.* 8.1-6, pp. 91–108. ISSN: 0968090X. DOI: 10.1016/S0968-090X(00)00026-7.

Yin, Yifang et al. (2018). "Feature-based Map Matching for Low-Sampling-Rate GPS Trajectories". In: *ACM Trans. Spat. Algorithms Syst.* 4.2, pp. 1–24. ISSN: 2374-0353. DOI: 10.1145/3266430.

Yuan, Jing et al. (2010). "An Interactive-Voting based Map Matching algorithm". In: *Proc. - IEEE Int. Conf. Mob. Data Manag.*, pp. 43–52. ISSN: 15516245. DOI: 10.1109/MDM.2010.14.

Zheng, Kai et al. (2012). "Reducing uncertainty of low-sampling-rate trajectories". In: *Proc. - Int. Conf. Data Eng.*, pp. 1144–1155. ISSN: 10844627. DOI: 10.1109/ICDE.2012.42.

Zhu, Lei, Jacob R Holden, and Jeffrey D Gonder (2017). "Trajectory segmentation map-matching approach for large-scale, high-resolution GPS data". In: *Transp. Res. Rec.* 2645.1, pp. 67–75.