Chapter 21

# FORENSIC TRACKING AND MOBILITY PREDICTION IN VEHICULAR NETWORKS

Saif Al-Kuwari and Stephen Wolthusen

**Abstract** Most contemporary tracking applications consider an online approach where the target is tracked in real time. In criminal investigations, however, it is often the case that only offline tracking is possible, i.e., tracking takes place after the fact. In offline tracking, given an incomplete trace of a target, the task is to reconstruct the missing parts and obtain the full trace. The proliferation of modern transportation systems means that a targeted entity is likely to use multiple modes of transportation. This paper introduces a class of mobility models tailored for forensic analysis. The mobility models are used to construct a multi-modal forensic tracking system that can reconstruct a complete trace of a target. Theoretical analysis of the reconstruction algorithm demonstrates that it is both complete and optimal.

**Keywords:** Forensic tracking, mobility models, trace reconstruction

## 1. Introduction

Traditional digital forensics is primarily concerned with extracting evidence from electronic devices that may have been associated with or used in criminal activities. In most criminal cases, however, it is also desirable to uncover additional information about suspects, including details about their physical activities. Investigating the locations of a suspect before, during and after a crime, may constitute key evidence, especially if it helps prove that the suspect was in a specific location at a specific time that he previously denied. This type of investigation is called "forensic tracking" [2], where the tracking is conducted for forensic purposes.

Forensic tracking investigations are usually carried out in an offline manner. A location trace of a suspect is obtained, which undergoes a probabilistic analysis designed to reconstruct the missing parts. A prime example is when a target is randomly captured by CCTV cameras scattered over a particular area. Previous work has focused on conducting offline forensic investigations in a vehicular setting [1]. This paper extends the approach to deal with scenarios where a suspect uses multiple modes of transportation.

The trace reconstruction algorithm described in this paper involves two main phases: (i) scene representation; and (ii) trace reconstruction. Scene representation uses information about when and where a target was observed to create scattered points over an area, that are subsequently connected to determine the routes that the target could have taken. Trace reconstruction attempts to fill the gaps of missing data between the points where the target was observed. In a multi-modal scenario, a targeted entity is expected to use multiple modes of transportation; thus, all possible routes through the gaps involving pedestrian routes, public routes and a combination of both must be considered to obtain a complete trace of the target. Theoretical analysis of the reconstruction algorithm demonstrates that it is both complete and optimal.

## 2. Scene Representation

In order to systematically reconstruct the trace of a target, a graphical representation (map) of the crime scene and the surrounding area must be generated. This is accomplished in five steps as described in this section. To simplify the notation, unnecessary labels and tags are dropped when referring to certain edges and vertices in the map.

**Step 1: Map Preparation.** In this initial step, a schematic map $G_M$ (based on a geographical area $M$) of the reconstruction scene is obtained. The reconstruction scene corresponds to the area over which the target trace is to be reconstructed. No restrictions are imposed on the size of $G_M$ other than it must cover: (i) all the points at which the target was observed (available traces of the target); and (ii) crime location(s).

Let $G_M = (\mathcal{V}^{G_M}, \mathcal{E}^{G_M})$ be a scene graph with vertices $\mathcal{V}^{G_M}$ and edges $\mathcal{E}^{G_M}$. We assume that $\{X_s^{G_M} \cup C^{G_M}\} \in \mathcal{V}^{G_M}$ such that:

- $X_s^{G_M} = \{x_1^{\kappa_p}, \ldots, x_n^{\kappa_q}\}$ is the set of locations where the target $s$ was observed; $\kappa_p < \kappa_q$ are the first and last times that $s$ was observed in $G_M$.
- $C^{G_M} = \{c_1^{\kappa_k}, \ldots, c_m^{\kappa_l}\}$ is the set of crime locations visited between times $k$ and $l$. To simplify the discussion, we describe the specifics

of the reconstruction algorithm using a single crime. Of course, the algorithm is applicable to multiple crimes.

**Step 2: Route Marking.** In this step, relevant public transport networks (e.g., buses and trains) $B_1, B_1, \ldots, B_n \in \mathcal{B}$ are marked on $G_M$. A transport network $B_j \in \mathcal{B}$ consists of a set of routes $B_j = \{R_1^{B_j}, R_2^{B_j}, \ldots, R_r^{B_j}\}$ that constitute most of the vertices and edges in $G_M$. Since only public transport routes are marked, vertices in a route $R_i$ correspond to a stop (e.g., train station) denoted as an $S$-vertex, or a road turn denoted as a $U$-vertex. Similarly, edges can be routed (i.e., part of a route) denoted as a $B$-edge, or unrouted denoted as a $W$-edge (mostly added in Step 4 below).

Let $e^\rho$ be an edge of type $\rho$, then:

$$\rho = \begin{cases} B & \text{if } e \in \bigcup_{B_j \in \mathcal{B}} \bigcup_{i=1}^{|B_j|} R_i^{B_j} \\ W & \text{if } e \in \mathcal{E}^{G_M} \setminus \bigcup_{B_j \in \mathcal{B}} \bigcup_{i=1}^{|B_j|} R_i^{B_j} \end{cases}$$

where $|x|$ is the number of elements in set $x$ assuming that $x$ does not have repeated elements (i.e., corresponding to loop-free routes).

Thus, a route $R_i$ is defined by the set of vertices $V_{R_i} = \{v_1, v_2, \ldots, v_k\}$ it comprises, and the edges $E_{R_i} = \{e_1, e_2, \ldots, e_{k-1}\}$ that link the vertices.

After all the routes are marked, the available traces of the target $X_s^M = \{x_1, x_2, \ldots, x_n\}$ are plotted. The traces specify the times and locations where the target has been observed in $G_M$ (these form the gaps that must be reconstructed). Each $x_i$ is either located on top of a vertex or over an edge (corresponding to a location at an intersection or on a road), i.e., $x_1, x_2, \ldots, x_n \in \mathcal{V}^{G_M} \cup \mathcal{E}^{G_M}$.

Elements in $X_s^M$ should naturally be represented as vertices. Thus, if any $x_i$ is located on $e_i \in \mathcal{E}^{G_M}$, then $e_i$ is split at the location of $x_i$ such that $e_i = e_i^1 + e_i^2$. Following this, $x_i$ is added to $\mathcal{V}^{G_M}$ (as a $U$-vertex) and $e_i$ is replaced by $e_i^1, e_i^2$ in $\mathcal{E}^{G_M}$, while updating the $V^{R_i}$ and $E^{R_i}$ of any route $R_i$ passing through $e_i$. Next, the locations of the crimes $C^{G_M} = \{c_1, c_2, \ldots, c_m\}$ are marked on $G_M$, but this time a $c_i$ may not be on top of a vertex or an edge, in which case, a $W$-edge is created between $c_i$ and the closest $v_i \in \mathcal{V}^{G_M}$. Note that it is acceptable for $c_i$ to be on top of an edge $e_i \in \mathcal{E}^{G_M}$ because $C^{G_M}$ is not involved in the reconstruction process. Finally, the directions of all the edges $e_i \in \mathcal{E}^{G_M}$ are specified. The directions of the $B$-edges $e_i^B$ can easily be determined by referring to their corresponding routes, while the $W$-edges $e_i^W$ are undirected.

Generally, a single edge $e_i$ or vertex $v_i$ cannot have two different types at the same time. If a particular vertex $v_i$ is part of $n$ routes $R_i, R_j, \ldots, R_n$, then it is an $S$-vertex as long as $v_i$ is an $S$-vertex in at least one of $R_i, R_j, \ldots, R_n$; otherwise, it is a $U$-vertex. In contrast, edges are not allowed to be part of more than one route because different routes may assign different weights to their edges. Thus, if more than one route traverses an edge, then as many edges as there are routes are created.

Let $e_i : v_p \rightarrow v_q$ be an edge between vertices $v_p$ and $v_q$, and suppose that $n$ routes pass through $e_i$, then $e_i$ is relabeled to $e_{i,1}$ and $n-1$ extra edges are created and labeled as $e_{i,2}, \ldots, e_{i,n}$. Thus, $G_M$ is a mixed multi-graph (i.e., it contains directed and undirected edges) and multiple directed edges can have the same head and tail vertices.

**Step 3: Vertex/Edge Labeling.** All the vertices and edges (except $W$-edges) are assigned unique labels to specify the routes of which they are a part. A vertex $v_i^{\ell_i}$ with label $\ell_i = R_j^k, \ldots, R_m^n$ indicates that the $i^{th}$ vertex in $\mathcal{V}^{G_M}$ is simultaneously the $k^{th}$, $\ldots$, $n^{th}$ vertex of routes $R_j, \ldots, R_m$, respectively. Since all the vertices are parts of routes, a label $\ell$ should contain information about at least one route. Edges are characterized by the vertices they link. Thus, $e_i^{\ell_i} : v_p^{\ell_p} \rightarrow v_q^{\ell_q}$ means that the $i^{th}$ edge in $\mathcal{E}^{G_M}$ has its head and tail at $v_p, v_q \in \mathcal{V}^{G_M}$ where $p, q \in \{1, 2, \ldots, |\mathcal{V}^{G_M}|\}$. The head $v_p$ and tail $v_q$ belong to at least one common route, and they are ordered in succession according to the direction of the edge. If more than one route passes by $e_i$, extra parallel edges are created and labeled (Step 2).

**Step 4: End Vertices.** After all the vertices and edges are labeled, a special set $\rho^{G_M}$ is created that contains all the "end" vertices – these are the first and last vertices of every route $R_i \in B_j$ (head and tail of $R_i$). To simplify the discussion, we consider the routes of a single transportation network $B_j$; this can easily be extended to multiple networks $B_m, \ldots, B_n \in \mathcal{B}$.

Vertices belonging to $\rho^{G_M}$ are found by computing the adjacency matrices $A^{R_1}, A^{R_2}, \ldots, A^{R_n}$ of all the routes $R_1, R_2, \ldots, R_n \in B_j$ where $|B_j| = n$ ($B_j$ contains $n$ routes). A particular vertex in $R_i$ belongs to $\rho^{G_M}$ if its corresponding row in $A^{R_i}$ sums to one. Note that $A_{i,j}^{R_y}$ represents the element in the $i^{th}$ row and $j^{th}$ column of $A^{R_i}$. An entire row is denoted as $A_{i,*}^{R_z}$ and an entire column is denoted as $A_{*,j}^{R_z}$ (i.e.,

$A^{R_z} = A^{R_z}_{*,*}$). Thus, formally:

$$\rho^{G_M} = \bigcup_{B_j \in \mathcal{B}} \rho^{G_M}_{B_j} = \bigcup_{B_j \in \mathcal{B}} \bigcup_{R_i \in B_j} \left( v_k : \sum_{k \in R_i} A^{R_i}_{k,*} = 1 \right)$$

**Proposition 1.** *A vertex $v_j \in V_i$ in the adjacency matrix $A^{R_i}$ of a finite loop-free route (i.e., simple path) $R_i = (V_i, E_i)$, where $V_i$ and $E_i$ are the sets of vertices and edges forming $R_i$, is an end vertex if its corresponding row $A^{R_i}_{j,*}$ in $A^{R_i}$ sums to one.*

*Proof.* Let the route $R_i$ be represented by the ordered sequence of vertices $v_1, v_2, \ldots, v_n$ where $v_1$ and $v_n$ are the first and last vertices of $R_i$ (end vertices). Clearly, $v_1$ and $v_n$ are each adjacent to a single vertex belonging to $R_i$, namely $v_2$ and $v_{n-1}$, respectively. All the other vertices $v_2, \ldots, v_{n-1}$ are adjacent to two vertices belonging to $R_i$, i.e., $v_i$ is adjacent to $v_{i-1}$ and $v_{i+1}$ for $i \in \{2, \ldots, n-1\}$. Therefore, $A^{R_i}_{1,*} = A^{R_i}_{n,*} = 1$, while $A^{R_i}_{i,*} = 2$ for $i \in \{2, 3, \ldots, n-1\}$. □

Since the routes in $G_M$ are directed, $\rho^{G_M}_{B_j} = \overrightarrow{\rho}^{G_M} \cup \overleftarrow{\rho}^{G_M}$ where $\overrightarrow{\rho}^{G_M}$ and $\overleftarrow{\rho}^{G_M}$ are the sets of head and tail (end) vertices of the routes in $G_M$.

**Step 5: Additional Edges.** This final step creates additional $W$-edges between several vertices. A new $W$-edge is created between any two $S$-vertices if: (i) they belong to different routes; and (ii) the distance between them is no greater than a threshold $W_{max}$. Formally, the set $\eta$ of the new $W$-edges is:

$$\eta = \left\{ e^{W,\ell_k}_k = v^{S,\ell_m}_m \leftrightarrow v^{S,\ell_n}_n : \ell_m \neq \ell_n \wedge d(v^{S,\ell_m}_m, v^{S,\ell_n}_n) \leq W_{max} \right\}$$

where $d(x, y)$ is the distance between $x$ and $y$. Note that the effect of the infrastructure on the $W$-edges is disregarded. In other words, we assume that there are no major obstacles between the $S$-edges that prevent $W$-edges from being created. However, integrating infrastructure information is easy because most modern maps contain such information. Next, $d(x, y)$ is found by rerouting around the infrastructure and checking that $d(x, y) \leq W_{max}$. Finally, the graph $G_M = (\mathcal{V}^{G_M}, \mathcal{E}^{G_M})$ is defined in terms of its edges and vertices, where $\mathcal{E}^{G_M} = E^{G_M}_R \cup E^{G_M}_W$ and $\mathcal{V}^{G_M} = X^{G_M}_s \cup C^{G_M} \cup V^{G_M}_S \cup V^{G_M}_U$.

## 3. Mobility Modeling

Mobility models have traditionally been used in computer simulation, where running an experiment (e.g., evaluating a protocol) on real systems is both costly and inconvenient. Mobility models generate artificial mobility traces that resemble mobility patterns of real entities. However, these traces cannot be directly used to reconstruct real traces that have been already made by real-life entities. This is mainly because real mobility patterns are based on human judgment, which is usually stochastic in nature. However, we show that, although mobility traces generated by these models are sufficiently artificial, they can still be used effectively in the reconstruction process.

Mobility models are usually developed at a microscopic level, modeling the mobility of each object in relation to its surrounding environment and neighboring objects, and thus generating realistically-looking traces. Most models, therefore, carefully parameterize the velocity and acceleration of the objects and repeatedly adjust them throughout the simulation. However, in our case, mobility models are used to estimate the time a target entity may have spent while moving from one point to another, completely disregarding the microscopic details. We refer to this class of mobility models as "delay mobility models." In addition, since we are considering a multi-modal scenario where an entity occasionally changes its mode of transportation, it is necessary to model each mobility class as well as the "gluing" of the different models to obtain smooth transitions.

The following sections introduce several mobility delay models, and proceed to model the transition between them.

### 3.1 Pedestrian Mobility Delay Model

Popular pedestrian mobility models, such as the social force model [6], cannot be directly used in our scenario because they require detailed microscopic information that may not be available. Also, the models focus on inter-pedestrian behavior, which is not important in our scenario. Therefore, we introduce the pedestrian mobility delay model (PMDM) to calculate the time an entity $x$ (pedestrian) takes to move from point $a$ to point $b$ (we are only concerned about the time). The mobility of the entity is mainly influenced by the static and moving obstacle objects that force the entity to perform a suitable "maneuver" in order to avoid them. Each obstacle object to be avoided by the entity is represented as a circle with known centre and radius.

The extra distance that the entity has to travel when maneuvering around an obstacle is approximately the length of the arc formed by

the chord cutting through the obstacle's circle based on the entity's direction. The time spent by the entity to traverse from point $a$ to point $b$ is given by:

$$t^s_{a,b} = \frac{d_{a,b} + \sum_{i \in \mathcal{S}} r_i \cos^{-1}\left(\frac{2r_i^2 - c^2}{4r_i}\right) + \omega}{v_s - \epsilon} + \sum_{j \in \mathcal{M}} g((r_s + r_j) - d_{x,j}) \quad (1)$$

where $\mathcal{S}$ and $\mathcal{M}$ are the spaces of the static and moving objects, respectively; $r_i$ is the radius of the circle surrounding an object (representing its range); $c$ is the length of the chord cutting through the object's circle (obtained via secant line geometry and the direction $\overrightarrow{e_s}$ of the target); $r_s$ is the radius of the circle surrounding the target $s$ moving with speed up to $v_s$; $d_{s,j}$ is the distance between the center of $s$ and the center of $j$; $\omega$ is a slight delay due to random factors imposed on the entity (e.g., crossing a road); $\epsilon$ is a random negative value (modeling the deceleration of the entity as it avoids obstacles); and $g()$ is a function given by:

$$g((r_s + r_j) - d_{s,j}) = \begin{cases} 1 & \text{if } r_s + r_j \geq d_{s,j} \\ 0 & \text{otherwise} \end{cases}$$

which models the time the entity pauses when it comes across a moving object (i.e., waiting for the object to move away).

At first glance, Equation (1) may appear to incorporate microscopic details because it models the interactions between objects. However, these details can be modeled without necessarily simulating the scenario at a microscopic level and only by assuming knowledge of the movement directions of the entity. The maneuvering behavior of the entity around static objects (e.g., buildings) is easily modeled by referring to the scene map $G_M$. The number of interactions between an entity and moving objects (e.g., other pedestrians) can be estimated subjectively based on the popularity of the area and the time.

## 3.2 Transport Mobility Delay Models

Another class of common mobility models describes the mobility of vehicular entities, modeling public transport modes such as buses, trains, and underground tubes. Note that we do not consider private vehicles because they are not relevant to our scenario; however, they can be considered to be a special type of the mobility model described in this section (TTMDM). In this model, the mobility of objects is more structured and less stochastic than those in pedestrian models because they are usually constrained by a fixed infrastructure (e.g., roadways and train tracks). However, based on the infrastructure, a distinction

can be made between two naturally different types of vehicular mobility patterns. We call the first type "traffic-based transport" and the second "non-traffic-based transport."

The traffic-based transport mobility delay model (TTMDM) is concerned with objects whose mobility is governed by uncertain parameters that, in some cases, could affect the mobility behavior significantly. The model describes the mobility of objects like buses, coaches and similar road-based public transport carriers whose mobility patterns highly depend on road traffic conditions that cannot be modeled precisely in most cases. The non-traffic-based transport mobility delay model (NTMDM), on the other hand, is easier to develop because random delay factors (such as those in TTMDM) have negligible, if any, effects on mobility behavior. NTMDM is used to model the mobility of infrastructure-based public transportation modes such as trains and underground tubes where, apart from rare occasional signal and other minor failures, have deterministic mobility patterns.

**Traffic-Based Transport Mobility Delay Model.** Most realistic traffic-based mobility models [5] adjust the velocity of objects to avoid collisions. However, this level of microscopic modeling is not required in our scenario because we are only concerned about the time taken for objects to move from one point to another, not their actual movements. Thus, we model the factors that affect the time value, which is estimated using the equation:

$$t_{a,b} = \frac{d_{a,b}}{\bar{v}_{a,b}} + \left[ D_{a,b}^{\text{traffic}} + \sum D_{a,b}^{\text{interest}} + \sum D_{a,b}^{\text{abnormal}} \right] \tag{2}$$

where $\bar{v}_{a,b}$ is the maximum allowable speed of the roadway between points $a$ and $b$; $D_{a,b}^{\text{traffic}}$ is the expected traffic delay of the roadway between points $a$ and $b$ that depends on the geographical and physical characteristics of the area and the time of day; $D_{a,b}^{\text{interest}}$ are delays incurred by points of interest located between $a$ and $b$; and $D_{a,b}^{\text{abnormal}}$ represents abnormal events on the road segment between $a$ and $b$ (e.g., accidents and breakdowns), both $D_{a,b}^{\text{interest}}$ and $D_{a,b}^{\text{abnormal}}$ can be obtained offline from public resources (e.g., maps) or from the police.

**Non-Traffic-Based Transport Mobility Delay Model.** Modeling non-traffic-based transport is clearly more straightforward because the uncertainty of the stochastic delays suffered by traffic-based transport is largely eliminated (or mitigated). This class describes the mobility of vehicular entities with fixed infrastructure such as trains and underground tubes. In this case, the time taken by an object to move from

point $a$ to point $b$ can be computed using the classical distance equation:

$$t_{a,b} = \frac{d_{a,b}}{v_{a,b}} + \sum D_{a,b}^{\text{abnormal}}$$

where $v_{a,b}$ is the fixed speed of the object on its journey from $a$ to $b$ spanning a distance $d_{a,b}$ (which can be obtained offline).

## 3.3 Multimodal Mobility Delay Model

Conventionally, when modeling the mobility of a particular object, it is implicitly assumed that the object's behavior is consistent throughout the simulation. However, in our scenario, we cannot rule out the possibility that the target could have used multiple different transportation modes, each with different mobility characteristics. Thus far, we have introduced three mobility delay models (PMDM, TTMDM and NTMDM). We now model the transition between them by constructing a multimodal mobility model (MMM) to assure a continuous flow of the target. Essentially, MMM only models the "transition behavior" between two different models (or two different carriers of the same model) because, once the transition is completed, the relevant mobility model is called to simulate the mobility of the next part of the journey until another transition is required. A "homogenous" transition occurs between two carriers of the same mobility model (e.g., changing a bus). On the other hand, a "heterogeneous" transition occurs between two carriers of different mobility models (e.g., changing from a bus to a train).

In a vehicular setting, we are interested in tracking the target who is transported by a carrier vehicle, not the vehicle itself; thus, the entity who makes the transition must be modeled. Clearly, in PMDM, both the entity and the carrier are a single component. When an entity shifts from any model to PMDM, the transition is smooth and incurs no delay (i.e., an individual does not have to wait before commencing a "walk" behavior). For any other situation, however, transition modeling is required to calculate the time an entity waits before shifting to the next model (or carrier). The main idea is to observe the timetables of the carriers at the transition location and calculate the transient wait time.

In a level-1 transition, the entity shifts from PMDM to TTMDM or NTMDM. In both cases, the entity most likely experiences a slight transient delay due to the time interval between its arrival at location $x$ and when the next carrier belonging to the intended model departs. Therefore, as soon as the entity arrives at $x$, it checks the intended carrier's timetable for the next departure time at its current location based on the current time and calculates the time difference. We will

discuss this process in detail when we describe a level-2 transition, which generalizes a level-1 transition.

Recall that in our scene graph $G_M$ roadways are represented by edges $\mathcal{E}^{G_M}$ and intersections by vertices $\mathcal{V}^{G_M}$. A transition can only occur at an intersection, so let $v_i \in \mathcal{V}^{G_M}$ be a transition vertex at which $n$ carriers from either TTMDM or NTMDM stop. Let these carriers be denoted by $R_1, R_2, \ldots, R_n$ (this information is included in the label of $v_i$; see Section 2). The first step is to obtain the timetables of the $n$ carriers $T^{R_1}, T^{R_2}, \ldots, T^{R_n}$ and convert them into matrices $M^{R_1}, M^{R_2}, \ldots, M^{R_n}$, where the rows represent stops and the columns represent journeys.

Note that the dimensions of the matrices depend on the timetables and may be different for different carriers. Next, we extract the rows corresponding to $v_i$ from $M^{R_1}, M^{R_2}, \ldots, M^{R_n}$ and create a 3D matrix $M^{v_i}$ by superposing the rows.

The dimensions of this new 3D matrix $M^{v_i}$ are $1 \times L \times n$, such that:

$$L = max\{w(M^{R_1}), w(M^{R_2}), \ldots, w(M^{R_n})\}$$

where $w(M)$ is the width (number of columns) of matrix $M$. In other words, $L$ is the number of journeys made by the carrier $R_i$ that makes the highest number of journeys where $i \in \{1, 2, \ldots, n\}$. Obviously, if $R_1, R_2, \ldots, R_n$ do not all make the same number of journeys, $M^{v_i}$ would contain some undefined values. We assume access to a global clock that returns the current time when the function $cTime()$ is invoked. After $M^{v_i}$ is created, $c = cTime()$ is used to build a $1 \times n$ matrix $\hat{M}^{v_i} = [m_{1,1}, m_{1,2}, \ldots, m_{1,n}]$ such that:

$$m_{1,j} = \begin{cases} |c - m_{1,j,z}| + \epsilon & \text{if } z \geq c \geq z + 1 \\ \epsilon & \text{if } c = z \text{ or } c = z + 1 \\ \infty & \text{otherwise} \end{cases}$$

where $\epsilon$ is a random delay representing the factors that may hold the carriers (e.g., traffic) plus the wait time at each stop. The matrix $\hat{M}^{v_i}$ now indicates how long an entity at the current location $x$ has to wait to pick a carrier $R_1, R_2, \ldots, R_n$ passing by $x$ (regardless of whether $R_1, R_2, \ldots, R_n$ belong to the same or different models). In particular, the matrix lists all the carriers that stop at $v_i$ along with their delay times.

## 4. Trace Reconstruction

Classical missing data algorithms, such as EM [3] and data augmentation [7], cannot be directly used in our scenario because the algorithms primarily make statistical inferences based on incomplete data, but do

not reconstruct traces. Additionally, it cannot be assumed that a sufficiently large number of available traces are available in order to use these algorithms. Iterative sampling algorithms, such as Markov chain Monte Carlo (MCMC) [4], even when adapted for missing data, cannot be used for the same reasons. Instead, we take a different algorithmic approach to fill the "gaps" formed by the missing traces. We develop an efficient reconstruction algorithm that, using mobility delay models, selects the route(s) that the target most likely has taken through a gap given the time it spent traversing the gap. In the worst case scenario, the algorithm eliminates several routes that the target could not possibly have taken, which may still constitute important evidence.

The reconstruction algorithm $\mathcal{A}_R$ first considers each gap individually and reconstructs the gap. The reconstructed gaps are connected to obtain the full trace of a target $s$.

Abstractly, $\mathcal{A}_R$ has two fundamental building blocks: (i) a multigraph traversing algorithm called weight-bound-search (WBS); and (ii) various mobility models. After $\mathcal{A}_R$ is executed, the reconstruction algorithm proceeds by running WBS over a gap. The WBS algorithm, in turn, repeatedly calls the mobility models (possibly via $\mathcal{A}_R$) and returns a route (or routes) that connect the gap. The $\mathcal{A}_P$ algorithm then reconstructs the other gaps in a similar fashion.

The WBS algorithm employs a branch-and-bound approach to optimize the reconstruction process, and uses a "crawler" to traverse gaps and find plausible routes. For a gap $G_p : v_m \rightarrow v_n$ between vertices $v_m$ and $v_n$ where $m, n \in \{1, 2, \ldots, |\mathcal{V}^{G_M}|\}$, a crawler $C^{G_p}$ is generated at $v_m$ and broadcasted toward $v_n$. The crawler $C^{G_p}$ maintains two data structures: (i) a LIFO list of vertices and edges traversed $\chi_{C^{G_p}}$; and (ii) a delay variable $\tau_{C^{G_p}}$. The variable $\chi_{C^{G_p}}$ is dynamically updated whenever $C^{G_p}$ traverses a vertex or edge to keep track of all the vertices and edges that the crawler $C^{G_p}$ has visited. The delay variable $\tau_{C^{G_p}}$, which is initially set to zero, is also dynamically updated whenever $C^{G_p}$ traverses an edge or an $S$-vertex (but not a $U$-vertex as described below).

When $C^{G_p}$ is first initiated at $v_m$, it checks the $v_m$ label $\ell_m = \{R_x^y, \ldots, R_k^l\}$, which contains information about the routes of which $v_m$ is a part and locates its $\hat{\ell_m}$ next-hop neighboring vertices using the equation:

$$\hat{\ell_m} = \begin{cases} |\ell_m| & \text{if } v_m \notin \overleftarrow{\rho}^G \\ |\ell_m| - k & \text{if } v_m \in \overleftarrow{\rho}^G \end{cases}$$

where $k$ is the number of times $v_m$ appears in $\overleftarrow{\rho}^G$ (number of routes in which $v_m$ is a tail-end vertex; these routes terminate at $v_m$ and thus do not have a next-hop).

Since we are considering a multi-graph, it is possible that some of the routes pass by the same next-hop neighbor (creating parallel edges between two vertices), so the $\hat{\ell_m}$ list may contain repeated vertices. If this is the case, each outgoing edge (even if all the edges are parallel) must be considered separately because the edge may have a different weight depending on the route to which it belongs. Thus, after all the next-hop neighbors are found, $C^{G_p}$ selects one of them, say $v_u$, finds the edges (routes) between $v_m$ and $v_u$, i.e., $\{e_i | e_i : v_m \rightarrow v_u\}$, and selects one $e_i$.

After an $e_i$ is selected, $C^{G_p}$ tags it as "visited," updates $\chi_{C^{G_p}}$ and proceeds to traverse it. It is important that $C^{G_p}$ tags every traversed edge as "visited" so that the edge is not revisited, potentially resulting in an infinite loop. Furthermore, if $C^{G_p}$ arrives at a vertex $v_k$ and finds that there is only one unvisited edge $e_i$, it tags $e_i$ as "visited," traverses it and then tags $v_k$ as "exhausted" so that $v_k$ is skipped if it happens to be a neighbor of some vertex that $C^{G_p}$ traverses in the future. Based on the type of the edges connecting $v_m$ with its next-hop neighboring vertices, $C^{G_p}$ calls the appropriate mobility model (PMDM, TTMDM or NTMDM) to calculate the delay of the edge. Next, it updates $\tau_{C^{G_p}}$ using the equation:

$$\tau_{C^{G_p}} = \tau_{C^{G_p}} + t_{v_x,v_y}$$

where $t_{v_x,v_y}$ is the delay returned for the edge $e_i : v_x \rightarrow v_y$ by the relevant mobility model (this applies to $R$- and $W$-edges). Similarly, when $C^{G_p}$ reaches an $S$-vertex $v_y$, it again updates $\tau_{C^{G_p}}$, but this time by calling MMM such that:

$$\tau_{C^{G_p}} = \tau_{C^{G_p}} + t_{v_y}$$

where $t_{v_y}$ is the delay assigned to $v_y$ by MMM. However, since there is no transition between mobility models in $U$-vertices, MMM is not called when a $U$-vertex is reached.

The crawler traverses the various routes by repeatedly backing-up whenever it finds a plausible or implausible route. The back-up procedure proceeds as follows: when the crawler finds an (im)plausible route, it checks $\chi_{C^{G_p}}$ and traverses backward through the edge in $\chi_{C^{G_p}}[1]$ toward the vertex $\chi_{C^{G_p}}[2]$ where $\chi[n]$ is the $n^{th}$ element of the list $\chi$. Next, it deletes the two elements from $\chi_{C^{G_p}}$ and repeats the entire traversal process, but this time it does not traverse the edge it just came from because it is now tagged as "visited" (or generally any edge tagged as "visited").

The crawler $C^{G_p}$ backs-up if: (i) $\tau_{C^{G_p}} + \epsilon > t_{v_n,v_m}$; or (ii) traverses a vertex/edge that already exists in its $\chi_{C^{G_p}}$; or (iii) $v_n$ (other end of the

gap) is reached where $\epsilon$ is a random value; or (iv) reaches a vertex $v_j$ such that $v_j$ is a tail-end vertex in all its routes (i.e., $v_j$ is childless).

In (i), the crawler backs-up when $\tau_{C^{G_p}}$ reaches a value greater than $t_{v_n,v_m}$ (time difference between when the target was observed at $v_m$ and later at $v_n$ – corresponding to the two ends of a gap $G_p$), and $\epsilon$ is a small constant. This means that the target would take much longer than $t_{v_m,v_n}$ if it had traversed that route. In (ii), only loop-free routes are accepted because these are what a rational target would opt to take (they also prevent infinite loops); so, if $C^{G_p}$ reaches a vertex $v_i$ such that $v_i \in \chi_{C^{G_p}}$, then it backs-up. In (iii), when $C^{G_p}$ reaches $v_n$, it checks $\tau_{C^{G_p}}$. If $\tau_{C^{G_p}} + \epsilon \leq t_{v_m,v_n} - \epsilon$, it backs-up (in other words, if a crawler returns a time much shorter than $t_{v_m,v_n}$, it is probably not the route that the target has taken). Otherwise, if $t_{v_m,v_n} - \epsilon \leq \tau_{C^{G_p}} \leq t_{v_m,v_n} + \epsilon$, it backs-up, returns the route in $\chi_{C^{G_p}}$ as a possible route the target may have taken and returns the corresponding $\tau_{C^{G_p}}$. Finally, in (iv) $C^{G_p}$ also backs-up when it reaches a childless vertex $v_j$; additionally, it tags $v_j$ as "exhausted."

The WBS algorithm terminates when its crawler terminates. This occurs when the crawler reaches a vertex in which all neighboring (next-hop) vertices are tagged as "exhausted," meaning that they have already been traversed extensively (i.e., all their outgoing edges are tagged as "visited").

**Proposition 2.** *Given a finite search graph, the WBS algorithm eventually terminates with or without returning valid routes.*

*Proof.* Since the WBS is a weight-based algorithm, it is guaranteed to stop traversing a particular route $R_i$ when its weight counter $\tau_{C^{G_p}}$ expires (i.e., $\tau_{C^{G_p}} \geq t_{v_m,v_n} + \epsilon$ where $t_{v_m,v_n}$ is the delay through gap $G_p : v_m \rightarrow v_n$ and $\epsilon$ is a small constant). Thus, the only way for the algorithm to run indefinitely is when it gets into an infinite loop and traverses the same route over and over again. However, a route $R_i$ cannot be traversed more than once because the algorithm tags every visited edge and does not traverse any tagged edge. Thus, the algorithm terminates as long as the graph has a finite number of edges. □

The WBS algorithm also terminates when processing an infinitely deep graph because it traverses the graph down to the point when its weight counter $\tau_{C^{G_p}}$ expires. However, the WBS algorithm may fail to terminate when processing an infinitely wide graph if none of the children of the current level has a weight higher than $\tau_{C^{G_p}}$. This, nevertheless, contributes to the completeness of the WBS algorithm.

**Proposition 3.** *Given a finite search graph, the WBS algorithm is complete. If multiple solutions exist, the algorithm returns all the solutions.*

*Proof.* For a gap $G_p : v_m \rightarrow v_n$, a valid solution means that there is a route $R_i : v_m \rightarrow v_n$ with a weight $\tau$ such that $t_{v_m,v_n} - \epsilon \leq \tau \leq t_{v_m,v_n} + \epsilon$. The crawler $C^{G_p}$ traverses all valid and invalid routes and terminates when there are no more edges to traverse. Therefore, if such a solution route $R_i$ exists, the crawler $C^{G_p}$ will find the route. □

If multiple routes are returned after the crawler terminates, then the algorithm selects the "best-fit" route such that:

$$|\chi_{C_f^{G_p}}| = min\{|\chi_{C_1^{G_p}}|, |\chi_{C_2^{G_p}}|, \ldots, |\chi_{C_n^{G_p}}|\}.$$

The route with less hops is selected because a rational target would probably choose a route that does not have many stops. Additionally, by observing the labels of the edges and vertices of the returned routes, a preferred route can be selected that minimizes the number of transitions between different mobility models and/or carriers with the same model.

## 5. Conclusions

The multi-modal trace reconstruction algorithm described in this paper engages mobility models tailored for forensic analysis to construct a complete trace of a target who may use multiple modes of transportation. Gaps of missing data between the points where the target was observed are filled by considering all possible routes through the gaps involving pedestrian routes, public routes and a combination of both. Theoretical analysis of the reconstruction algorithm demonstrates that it is both complete and optimal. Certain details have been omitted for reasons of space. Interested readers may contact the authors for a complete description of the algorithm and the accompanying theoretical analysis.

## References

[1] S. Al-Kuwari and S. Wolthusen, Probabilistic vehicular trace reconstruction based on RF-visual data fusion, *Proceedings of the Eleventh IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security*, pp. 16–27, 2010.

[2] S. Al-Kuwari and S. Wolthusen, Fuzzy trace validation: Toward an offline forensic tracking framework, *Proceedings of the Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2011.

[3] A. Dempster, N. Laird and D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 39(1), pp. 1–38, 1977.

[4] W. Gilks, S. Richardson and D. Spiegelhalter (Eds.), *Markov Chain Monte Carlo in Practice*, Chapman and Hall/CRC Press, Boca Raton, Florida, 1996.

[5] J. Harri, F. Filali and C. Bonnet, Mobility models for vehicular ad hoc networks: A survey and taxonomy, *IEEE Communications Surveys and Tutorials*, vol. 11(4), pp. 19–41, 2009.

[6] D. Helbing and P. Molnar, Social force model for pedestrian dynamics, *Physical Review E*, vol. 51(5), pp. 4282–4286, 1995.

[7] M. Tanner and W. Wong, The calculation of posterior distributions by data augmentation, *Journal of the American Statistical Association*, vol. 82(398), pp. 528–540, 1987.