Volume 35 (1), pp. 1–31
http://orion.journals.ac.za

ORiON
ISSN 0259–191X (print)
ISSN 2224–0004 (online)
©2019

Development of a map-matching algorithm for dynamic-sampling-rate GPS signals to determine vehicle routes on a MATSim network

Jaco-Ben Vosloo[*] Johan W. Joubert[†]

Received: 23 January 2019; Revised: 26 April 2019; Accepted: 27 April 2019

Abstract

The rapid development and proliferation of global positioning system (GPS)-enabled systems and devices have led to a significant increase in the availability of transport data, more specifically GPS trajectories, that can be used in researching vehicle activities. In order to save data storage- and handling costs many vehicle tracking systems only store low-frequency trajectories for vehicles. A number of existing methods used to map GPS trajectories to a digital road network were analysed and such an algorithm was implemented in Multi-Agent Transport Simulation (MATSim), an open source collaborative simulation package for Java. The map-matching algorithm was tested on a simple grid network and a real and extensive network of the City of Cape Town, South Africa. Experimentation showed the network size has the biggest influence on algorithm execution time and that a network must be reduced to include only the links that the vehicle most likely traversed. The algorithm is not suited for trajectories with sampling rates less than 5 seconds as it can result in unrealistic paths chosen, but it manages to obtain accuracies of around 80% up until sampling sizes of around 50 seconds whereafter the accuracy decreases. Further experimentation also revealed optimal algorithm parameters for matching trajectories on the Cape Town network. The use case for the implementation was to infer basic vehicle travel information, such as route travelled and speed of travel, for municipal waste collection vehicles in the City of Cape Town, South Africa.

Key words: MATSim, Map-Matching, GPS data processing, GPS trajectory.

[*] Centre for Transport Development, Industrial & Systems Engineering, University of Pretoria, South Africa, email: jacoben.vosloo@gmail.com

[†] Centre for Transport Development, Industrial & Systems Engineering, University of Pretoria, South Africa, email: johan.joubert@up.ac.za

http://dx.doi.org/10.5784/35-1-636

1 Introduction

In recent years, the availability of GPS data has increased dramatically, not only in the transport industry but also for human behaviour in general. The cause for this can be linked directly to the rapid development of GPS systems and their integration into mobile devices and fleet monitoring devices, as well as the increase in availability of dashboard-mounted travel guidance- and monitoring systems [5]. This spread of GPS-enabled systems and devices has led to an increase in accessibility of transport GPS trajectories; a sequence of GPS points recorded in chronological order. These trajectories are beneficial for many applications such as location-based service, urban planning and traffic management, by giving insight into understanding users' moving behaviour. Ghiani *et al.* [3] focus on the application of GPS to improve the performance of route planning algorithms on solid waste collection vehicles and state that by analysing vehicle trajectory data companies can significantly improve the services they provide. Although the benefits of GPS trajectories, especially in the transport industry, are evident through the use of intelligent transport systems (ITS) as described in Quddus *et al.* [8], the administration of the data can impose a considerable financial burden due to data warehousing. For applications where detailed location accuracy is not required, the data polling frequency is reduced to decrease operating costs by reducing data storage requirements. However, the resulting traffic information, albeit smaller in storage requirements, will be less accurate. The outcome of this is that if the polling frequency is low, for example, 1 GPS point every minute or 1 GPS point every 500 m, one will not be able to identify the route of the vehicle with a high level of certainty [6, 7]. Fortunately, there are many successful attempts documented in the literature of the development of map-matching algorithms that could accurately process the low polling frequency within a required level of accuracy.

In this study the specific use case for a map-matching algorithm was to analyse the routes of waste collection vehicles. Not only to identify the route a vehicle took, but also to infer from the results which parts of the route the vehicle was servicing and which parts were merely used for travelling from one service area to another, called *deadheading*. This will allow researchers to allocate the waste collected by the vehicle to a very specific area. To achieve this, the speed of the vehicle was used as a proxy to determine whether it is deadheading or collecting waste, since the travelling speed between the two activities differs significantly.

Due to the limitations of GPS devices caused by inherent error in accuracy, as well as the error caused by a low-sampling rate, map-matching algorithms are required to evaluate alternatives and give the most probable route the object travelled. The accuracy of map-matching could differ significantly depending on GPS accuracy and the sampling rate [10]. Without a proper map-matching algorithm it is unclear how to find the most probable route that the vehicle could have taken.

The objective for this study was to develop a map-matching algorithm that could be used to identify the most probable routes for vehicles from a dynamic sampling rate GPS data set. The primary focus of this study was to create the algorithm in Java so that it could be used to map a GPS trajectory to a Multi-Agent Transport Simulation (MATSim) network. Map-matching is the process of aligning a sequence of observed user positions with the road network on a digital map [5]. The choice to use a MATSim network is because this research forms part of a larger effort in which the agent-based, open source platform is used to model disaggregate people and vehicle movement on a large scale.

The proposed algorithm is robust and able to match GPS trajectories to an acceptable level of accuracy compared to other map-matching algorithms using similar data sets. However it is not suited for situations where GPS points are recorded in close proximity. This study not only delivered an algorithm available to use in the Java-based open source software, MATSim, but also experimental data sets and the algorithms that created them. The study also provided a novel way of using the probabilities from the map-matching algorithm to analyse the results in the absence of a true path (TP). The use of the probabilities together with the inferred speed versus free speed could provide substantial insight into the confidence level of the inferred path (IP).

2 Literature review

2.1 Map-matching methods

A number of studies exist on GPS point matching to a digital map but most researchers agree on the two most basic categorisations of map-matching algorithms, namely *local*, also called *incremental*, and *global.* Miwa *et al.* [7] concur with this categorical breakdown but refers to the two methods as *offline* and *online* due to the nature of their processing frame.

Ying *et al.* [11] describe a third approach, *multi-track map-matching*, while Lou *et al.* [5], although sharing research references with Ying *et al.* [11], describe it as a *statistical method.* Li *et al.* [4] also describe a multi-track map-matching algorithm but argue that all approaches can be categorised as either local or global and that multi-track map-matching is a global method.

Although the definitions of multi-track map-matching of both Ying *et al.* [11] and Li *et al.* [4], and the statistical method described by Lou *et al.* [5], align, perhaps the data-driven nature of the approach might be enough reason for a unique categorisation. Based on the more statistical approach of this method, reference will be made to multi-track map-matching as a separate category in this paper.

2.1.1 Local methods

Local methods are also known as online map-matching since they can be used to generate vehicle paths as new samples become available, typically at a rate of 1–30 s per sample. Quddus *et al.* [8] state that these local map-matching algorithms are generally appropriate for high-frequency positioning data from GPS at least 1 Hz or higher. They state that the use of these algorithms on low-frequency data, such as the data available for this study, especially in urban and sub-urban road networks are not suitable for local methods [9]. For this reason, this article considers local methods only briefly, mainly discussing their applicability to low polling data sets, while the global methods are discussed in detail.

Due to the online nature of local map matching methods they are often used in applications such as navigation, where a trade-off has to be made between accuracy, robustness and the speed of computation [5]. Local methods try to find a local match of geometries, and try to combine all partial matches into one. These methods work well on high-sampling rate GPS trajectories and are often used for analysis in a range of ITS and can provide services such as route guidance, fleet management, road user charging, accident and emergency response, bus arrival information, and other location-based services that require near real-time location information [8]. Local methods involve the processing of small amounts of data to identify vehicle route and current position on the road. Although GPS errors may affect the accuracy of the map-matching results, the procedure is concise because the process is repeated each time new data are collected [7]. Local methods can deal with simple road network structures if the polling frequency is low; however, due to the increase in complexity in road network structures in urban areas, the candidate road segments identified might be crowded, leading to a higher possibility of incorrect links being identified and causing a significant decrease in accuracy [11].

Local methods might yield some interesting algorithmic principles that can be applied in sparse GPS data sets, such as the one used in this study, but that it will not be as effective as some of the other methods available. The suggested way to improve the accuracy is to not analyse the points continuously, as per the local methods, but rather evaluate all the points of the complete trace at the end of travel using a global method.

It is worth noting that other researchers, such as Ghiani *et al.* [3], who have very specifically researched the use of GPS data from waste collection vehicles have not made reference to any of the aforementioned methods or classifications. They also refer to their method as the *reverse geocoding algorithm.* The principle of their algorithm matches more a local method, since it continuously matches the next data point based on previous points and not the whole set of points at once. Their data polling was also at a much higher frequency, 2 s, which makes their method less applicable to this study, even though their area of application is the same.

2.1.2 Global methods

While local methods assign portions of a trajectory onto a path based on the local geometry, global algorithms determine the globally optimal path after reading in complete trajectories. Global methods aim to match the entire trajectory with the road network after the object has completed its entire trajectory and therefore is also referred to as *offline map-matching.* Global methods focus on map-matching accuracy and robustness rather than on speed of execution [4].

Previous studies tried to either search for possible matches between the trajectory and the road segments using the minimum Fréchet distance, a measure of similarity, or to formulate map-matching problems as optimisation problems, trying to find the shortest travel distance between two points matched in the network [5]. However, as the complexity of the road network structure increases in urban areas, and the low-sampling rate of GPS data points becomes more of an issue, a significant decrease in accuracy occurs [11]. These shortcomings are addressed through the development of more intricate and complex methods.

Alt *et al.* [1] and Brakatsoulas *et al.* [2] present methods based on Fréchet distance or its variants and are suitable for comparing whole trajectories as they take the continuity of curves into account. In Alt *et al.* [1] the minimum Fréchet distance is determined by finding a monotone path in the free space created by two GPS points, from the lower left corner to the upper right corner. The algorithm runs in $O(mn \log^2 mn)$ time, where m and n are the number of edges and number of nodes in the road network respectively. Brakatsoulas *et al.* [2] continued on that work by proposing the use of the average Fréchet distance to reduce the effect of outliers. They also propose the use of the weak Fréchet distance as it reduces the runtime to $O(mn \log mn)$.

Yin & Wolfson [10] introduce a weight-based map-matching method that is accurate up to 94% depending on the GPS sampling rate. They make use of the edit distance to measure the similarity between trajectory and matched road segments, where the edit distance is the smallest number of insertions, deletions, and substitutions required to change IP to TP. The accuracy of their method decreases to below 60% when the sampling rates of GPS data points increased above 120 seconds. They consequently proposed that their method be used as an online method with high-frequency sampling data.

Lou *et al.* [5] proposed a novel method called spatial-temporal (ST) map-matching for low sampling rate trajectories. ST matching not only considers the spatial geometric and topological structures of the road network, but also the speed constraint of the road network. They managed to achieve 70% accuracies up to 5 min interval sampling GPS data, which is unattainable for local methods at such low data polling frequencies. Based on Lou *et al.* [5]'s work, Yuan *et al.* [12] proposed an interactive voting-based map-matching algorithm to improve accuracy of the results. They were able to improve the matching accuracy using the same data as Lou *et al.* [5] with an average of 10% for the same data set and road network over different data polling frequencies. Even when sampling at 10 min intervals they were able to achieve 67% accuracy.

Yuan *et al.* [12] and Ying *et al.* [11], further developed the ST algorithm of Lou *et al.* [5] breaking the model down into two modules; an offline mining model and an online matching module. The second model is called an online model due to the nature of the analysis being done on the data and not pre-analysing the trajectories compared to the mining model. This should not be confused with the local methods, which matches points continuously as they become available, and is also referred to as an online map-matching method.

The experimentation results of Ying *et al.* [11] show that their method is 100 times faster than the ST method of Lou *et al.* [5], and also shows significant improvement in accuracy, from 47% to 52% based on their specific comparative data set.

Global methods appear to be a suitable method for this project given their accuracy on sparse GPS trajectories and the effectiveness on big data sets.

2.1.3 Multi-track methods

Multi-track map-matching is a statistical approach where one matches a large number of possibly sparse trajectories simultaneously to the map by trying to recover regularity among input trajectories [4, 11]. Multi-track methods are also especially used for matching GPS observations when there is significant uncertainty in the data.

Multi-track map-matching and other statistical methods can be considered a subset of the general max-weight algorithms [4]. The focus of multi-track map-matching is to try to match a large number of possibly sparse trajectories simultaneously to the map. Usually, multi-track map-matching adopts data-driven techniques to improve the matching result using historical trajectories. Li *et al.* [4] further state that the the advantage of this approach comes from the observation that human generated trajectories, either from human operated vehicles or humans themselves, show a high degree of temporal and spatial regularity.

In most cases single track map-matching prefers cutting the corner while data-driven map-matching tends to follow larger, popular roads, although the distances travelled along the two paths are very similar and thus even the speed analysis would show little deviation from the segment free speed.

Multi-track map-matching methods might provide more accurate results in sparse GPS data sets, but because most methods use historical trajectories to find the most likely path for a trajectory, they were not suitable for the data set available for the current project due to the trajectories not grouped based on service area.

In conclusion, the best option for this study was to implement a global method similar to Lou *et al.* [5]'s ST method. Future work on this study can further enhance the algorithm based on the work of Yuan *et al.* [12] and Ying *et al.* [11].

2.2 Evaluation criteria

The two main measures for success of any map-matching algorithm are in terms of efficiency and accuracy, or in other words, run time and inference quality. Many of the aforementioned research papers and articles list the actual run time of the algorithm, as a function of the number of points and network segments, as the main criteria for the success of an algorithm. The runtime of an algorithm is calculated by means of parameter variation and by fitting a linear model on the output, thus getting the relationship between runtime and network complexity or GPS points to match.

To determine the accuracy of an algorithm there needs to be a true path, *i.e.* the correct path of the object, to compare the output of the algorithm to [5]. But often the true path is only available when an experimental high-frequency data set is available from which the sparse data set is created, or if the actual route has been logged through manual recording.

Miwa *et al.* [7] used two high-frequency data sets with intervals of 5 s and 50 m respectively. They created test data sets from these data sets by increasing the polling intervals to 20 s, 45 s and 90 s for the time-based data set and 100 m, 200 m and 450 m for the distance-based data set. This also allowed them to test the robustness of their algorithm on the sampling frequency, based on time as well as distance. Miwa *et al.* [7] still had to determine the correct route by hand, but this is believed to contain very few to no errors due to the high frequency of the GPS data. This enabled them to use the following formulas to calculate the accuracy of their algorithm:

$$\text{ARR} = \frac{\text{Length of correctly matched route}}{\text{Total length of correct route}}, \tag{1}$$

and

$$\text{IARR} = \frac{\text{Length of incorrectly matched route}}{\text{Total length of matched route}}. \tag{2}$$

If the matched route includes all of the links of the correct route accuracy ratio of route by length, total matched route (ARR) will be 1.0 even if incorrect links are included. On the other hand, inaccuracy ratio of route by length (IARR) will indicate how much of the matched route was incorrectly identified. Therefore, if the correct route is perfectly matched, ARR is 1.0 and IARR is 0.0. Miwa *et al.* [7] states that this provides better insight into the performance of the algorithm as opposed to the commonly used accuracy ratio of plot matched to correct links (ARP) index, as used by Lou *et al.* [5].

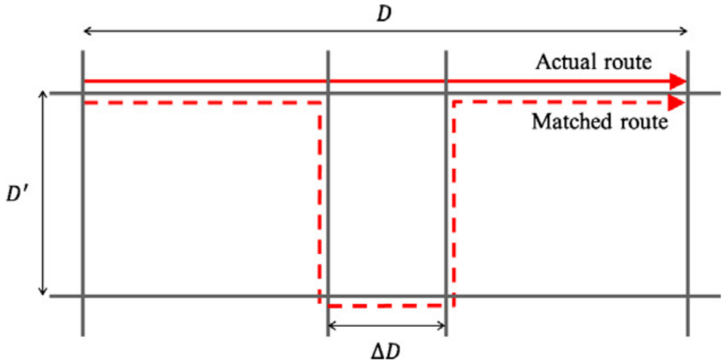


Figure 1: *Example of ARR and IARR.*

Figure 1 shows an example of the relationship between ARR and IARR. If ΔD decreases to 0 so will ARR increase to 1.0, which indicates a possible perfect match. However, the IARR will only approach 0 if D' reduces to 0.

Lou *et al.* [5] used a similar data set as Miwa *et al.* [7], and were able to synthesise experimental data sets from the human-labelled high-frequency data sets. Lou *et al.* [5] provided very similar metrics to Miwa *et al.* [7], but when comparing the length of the matched route to the actual route, they include the total length of the matched route and not only the correctly matched sections. Their definition is formulated as

$$\text{ARR2} = \frac{\sum \text{The length matched road segments}}{\sum \text{The length of identified road segments in correct route}}. \tag{3}$$

The drawback with the measurement in (3) is that if the algorithm identifies incorrect segments of similar length to the correct segments it missed, then the measurement can still provide a ratio of 1. Lou *et al.* [5] created a second measurement *Accuracy by Number* (4) that would have to be relied on to indicate the number of incorrect segments identified.

$$\text{ARRn} = \frac{\#\text{ of correctly matched road segments}}{\#\text{ All road segments in correct route}}. \tag{4}$$

However, (4) will still not provide an indication about the length of the incorrectly matched segments. For this reason this study will not be measuring *Accuracy by Length*, but rely on ARR and IARR. Lou *et al.* [5]'s measurement of *Accuracy by Number* (4), originally referred to them by the acronym *An*, is incorporated as it provides a useful additional insight to indicate how accurately individual segments were identified. To keep consistency in acronyms this measurement is renamed to accuracy ratio of route by number of links (ARRn).

Zheng *et al.* [13] calculated the inference quality of the algorithm by measuring the similarity between the inferred path IP and the true path TP using

$$\mathrm{Al} = \frac{\mathrm{LCR(TP,IP)}^{length}}{\mathrm{Max}(\mathrm{TP}^{length}, \mathrm{IP}^{length})}, \tag{5}$$

where *LCR* is the longest common road segment of IP and TP, and *length* indicates the overall length of the applicable path noted.

Although Zheng *et al.* [13] referred to their inferred route metric as *RI*, and true path as *Ground Truth* or *RG*, this study used the more intuitive acronyms, and appropriate to previously defined terminologies *inferred path*, IP and *true path*, TP respectively. In conclusion, all of the above measurements were used for this study, excluding ARR2, as it is replaced by ARR and IARR.

Lou *et al.* [5] argued that many current global methods have not tried using true paths from real world generated data to evaluate the actual matching accuracy and thus their true effectiveness is unknown. To some degree the same caveat did exist in this study, as there was no true path available for the waste collection vehicles' trajectories.

Li *et al.* [4] addresses this concern by constructing a benchmark data set. They selected 100 random sparse trajectories from their Beijing taxi data set and recruited four volunteers with more than five years of driving experience in Beijing to manually indicate the path they believed the taxis travelled. The purpose was to obtain a true path map for these trajectories based on how a real driver would choose a path that followed these trajectories based on his or her knowledge of the roads in and around the city.

3 Algorithm development

In this section the algorithm and problem statement are defined, adapted from Lou *et al.* [5], and describe how they were adjusted for this specific study. The first set of definitions explicitly define the common terms used in describing map-matching and how they relate to the MATSim-specific objects used for this study. MATSim is a collaborative open source project that can be used as a package in Java. This study use version 0.7.0 of MATSim and Java SE 1.7, for more info visit `https://github.com/matsim-org` and `https://www.oracle.com/technetwork/java/javase/overview/index.html` respectively.

To aid in explaining the algorithm, the following fictitious trip that was recorded from a waste collection vehicle that operated in the area shown in Figure 2 is used. The *true path* is the actual route that the fictitious driver travelled.

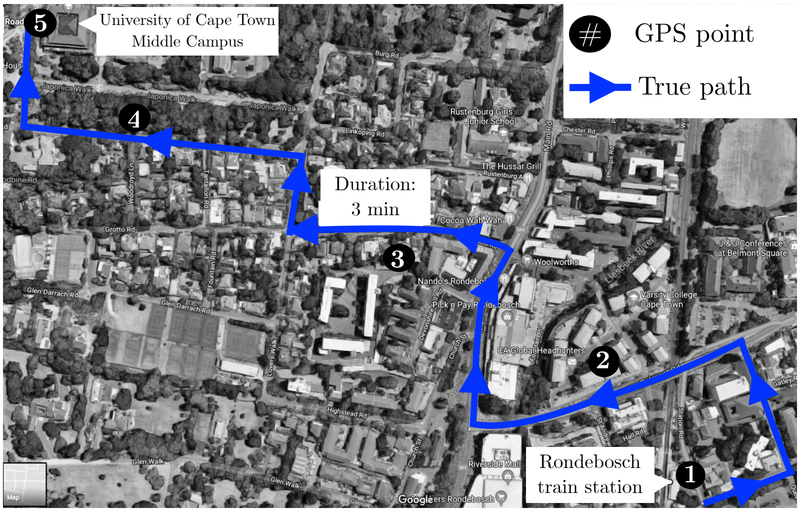


Figure 2: *Example of a low-sampling GPS trajectory between Rondebosch train station and University of Cape Town Middle Campus.*

3.1 Definitions

GPS point: To work in metres, the coordinate system used in MATSim is adapted from the Albers equal-area conic projection. Table 3.1 shows the GPS trajectory after the conversion to the SA-Albers projection. A trajectory using the SA-Albers projection is defined as $\boldsymbol{T}_{\text{SAA}}$.

Point	x	y	Time s
p_1	-513327.5134	-3716061.097	0
p_2	-513486.9662	-3715934.682	25
p_3	-513722.076	-3715759.626	50
p_4	-514033.4896	-3715661.143	100
p_5	-514115.2881	-3715473.827	150

Table 1: *GPS trajectory of example data with SA-Albers projection.*

Link: Similar to the definitions of a road segment, a link L in MATSim represents an edge with only one direction of travel. In order to represent bidirectional traffic between two points, two links need to be defined, one for each direction of travel. A link has, inter alia, the following fields:

- link identifier, L^{id} - the name of the link;
- a from and to node L^{fn}, L^{tn}, the starting and ending points of a link respectively;
- free speed L^{fs} in m/s. This is the speed an object is allowed to travel on the link, and can vary depending on the time of day; and
- the length of the link L^{length} in metres.

Since a link in MATSim does not contain any intermediate points, to represent a curved road segment, multiple links and nodes can be used to accurately represent the road shape in topological format. If the visual aspect of a link is not important, a curved road segment can, for example, be represented by just one straight link with the correct length defined programmatically and not graphically. This presents a potential issue in using the spatial and topological analysis part of the map-matching algorithm as it relies on the network to be as topologically and spatially accurate as possible, and MATSim network can only contain straight lines. As can be seen in Figure 3, the MATSim network available for this study has relatively accurate topological structures. This is achieved by representing curved segments on the road with a number of small, straight links.

Node: A node in MATSim, N, is defined as a topological point where one or more links can be joined and an agent can move from one link to another. A node has, inter alia, the following fields that are used during calculations: node ID N^{id}, list of In links N^{inLinks} (links that end inside this node), list of Out links N^{outLinks} (links that start inside of this node). In Figure 3, the nodes are represented by small grey diamonds that connect the links.

Network: A network is a set of links connected to each other via a set of nodes to form and represent a transport network along which agents can travel and interact with one another. A network is defined as $\boldsymbol{N_{MATSim}}(\boldsymbol{L}, \boldsymbol{N})$, where $\boldsymbol{N}$ is a set of nodes representing the intersections and terminal points of the road segments, and $\boldsymbol{L}$ is a set of links representing road segments.

The MATSim representation of the entire example network is illustrated in Figure 3 and includes the true path travelled by the vehicle as well as the GPS trajectory in the applicable coordinate system.

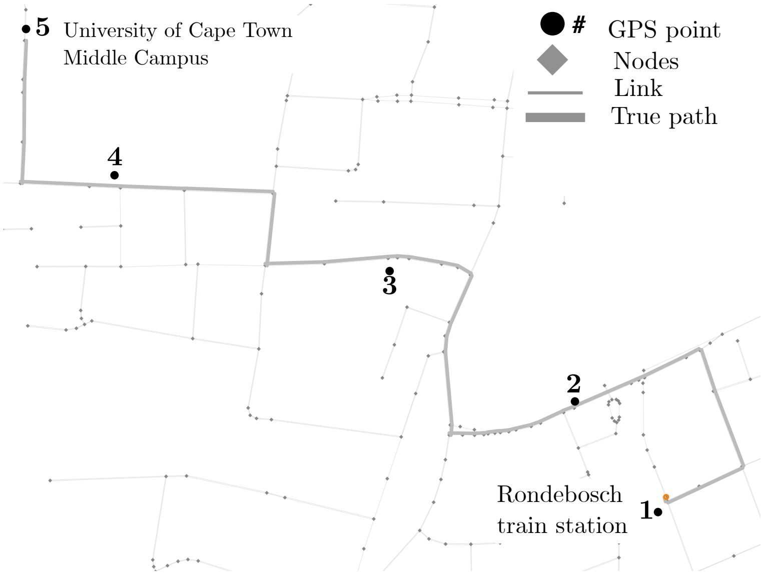


Figure 3: *MATSim network of sample data.*

Path: In MATSim, a path is a list of connected links along which agent can travel from one node to another, and is denoted by $\boldsymbol{P}_l$.

The map-matching problem can now be formulated as follows:

Problem statement: Given a raw GPS trajectory $\boldsymbol{T}_{\text{SAA}}$ generated by an agent travelling on a path on a road network $\boldsymbol{N_{MATSim}}(\boldsymbol{L}, \boldsymbol{N})$, determine the most likely path $\boldsymbol{P}_l$ that the agent has travelled.

3.2 Candidate preparation

For each GPS point in a trajectory, there is a potential list of road segments or links along which the agent could have travelled when generating the GPS point. These potential links are called *candidate links*. Given trajectory $\boldsymbol{T} = \{p_1, p_2, \ldots, p_n\}$, there exist a number of candidate links for each point $p_i \in \boldsymbol{T}$.

Lou *et al.* [5] made use of a search radius to identify a number of road segments for each point in the GPS trajectory where the road segment is within the radius from the GPS point.

There is no native functionality in MATSim to efficiently collect all the links that are within a radius of a certain point. However, all the nodes within a radius can be transposed into a specific data structure referred to as a *quadtree*, that allows for efficiently identifying nodes with a radius from a specific point. When all the nodes within a radius have been identified, one can check whether or not the links to and from the nodes are within the specified radius from the GPS point by determining the projected distance from the GPS point to the nearest point on the link. However, there exists an issue with explicitly defining the radius for collecting all the applicable nodes without taking into consideration the characteristics of the links within the network. The issue is that some nodes might be incident with the link right next to the GPS point but the link's endpoint nodes are outside of the search radius and thus the link will not be evaluated, leading to an incorrect path being inferred for the GPS trace. Figure 4 illustrates such an example, where GPS point 2 is associated with the highlighted link to its left, but the distance to the start and end nodes of this link is 1.7km and 1.5km away respectively. If the search radius for nodes in the quadtree does not include these outlying nodes, the GPS point will only have the short residential links to its right evaluated. Thus, the search radius needs to be a function of the longest link within the network to ensure that the algorithm does not miss potential candidate links that might possibly be the correct links. Since the search for nodes within a distance is efficiently performed using the quadtree, the algorithm uses the longest link length within the network as the search radius, which is not an input into the algorithm as per Lou *et al.* [5]'s implementation.

Once all the possible nodes have been identified and all the incoming and outgoing links from the nodes have been stored as potential candidate links for the GPS point being evaluated, a straight-line distance from the GPS point to each potential candidate link is calculated and saved. The list of potential candidate links identified is sorted according to proximity to the GPS point. The next step is to use only a certain number of links from the sorted list of potential candidate links for further analysis and result matching. The number of links used greatly affects the efficiency and effectiveness of the algorithm, and

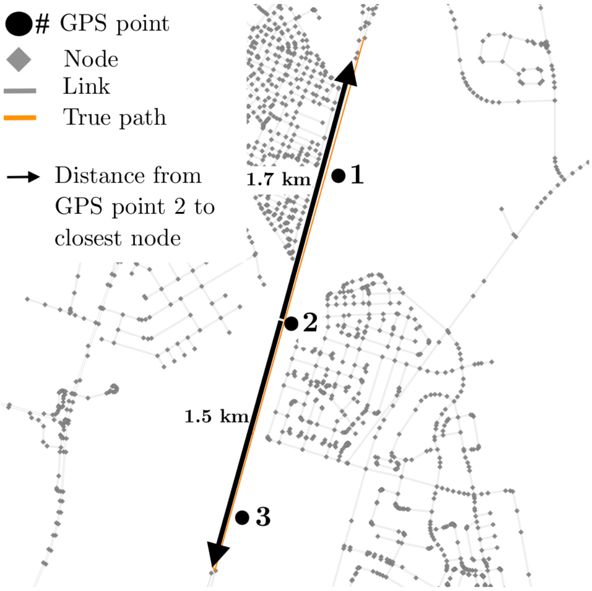


Figure 4: *Search radius required when searching for candidate links.*

a number of experiments were conducted to determine the influence of these parameters on results.

The next step was to determine a *candidate point* on each candidate link using segment projection, and was defined as follows:

Candidate point: This is the projection of a point p to a link l and is defined as point c on l such that $c = \arg\min_{c_i \in l} dist(c_i, p)$, where $dist(c_i, p)$ returns the distance between p and any point c_i on l. The jth candidate link and candidate point of p_i will then respectively be denoted by l_i^j and c_i^j. As shown in Figure 5, p_i's candidate points are c_i^1 c_i^2 and c_i^3. The candidate point represents the most likely point on the specific link at which the agent was when recording the GPS point being evaluated.

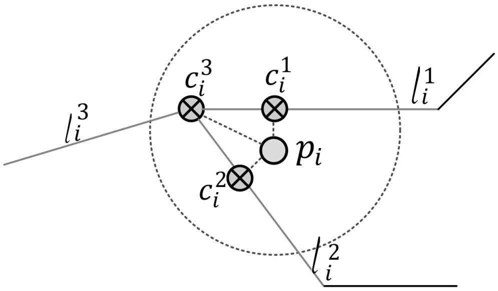


Figure 5: *Point projection on candidate links.*

Updated problem statement Once the candidate link sets and their respective candidate points are retrieved for all the sampling points on the trajectory $\boldsymbol{T}_{\text{SAA}}$, the problem statement becomes how to choose one candidate from each set so that $\boldsymbol{P}_l = \{c_i^{j_1}, c_i^{j_2}, \ldots c_n^{j_n}\}$ best matches $\boldsymbol{T}_{\mathrm{SAA}} = \{p_1, p_2, \ldots, p_n\}$.

3.3 Spatial analysis

In the spatial analysis both the geometric and topological information of the road network is used to evaluate the candidate points found in the previous step. The geometric information is used in the *observation probability*, and the topological information in the *transmission probability*.

Observation probability: The observation probability is defined as the likelihood that a GPS sampling point p_i matches a candidate point c_i^j and is computed based on the distance between the two points, defined as $dist(c_i^j, p_i)$. The inherent error in a GPS measurement causes p_i to be a certain distance from c_i^j and can be approximated using a normal distribution $N(\mu, \sigma^2)$. This distribution can be used to indicate how likely a GPS observation p_i can be matched to a candidate point c_i^j on the real road without considering its neighbouring points. Formally, the observation probability is defined as: $N(c_i^j)$, of c_i^j with regard to p_i as:

$$N(c_i^j) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{(x_i^j - \mu)^2}{2\sigma^2}} \tag{6}$$

where $x_i^j = dist(c_i^j, p_i)$, the distance between p_i and c_i^j. Similar to Lou *et al.* [5] this study makes use of a zero-mean normal distribution with a standard deviation of 20 m for GPS error. Using these parameters, the observation probability will always output very low probability values for any GPS point even if the point is within 0 m of the candidate point. During informal discussions with the original authors they indicated that the final ST probability value should rather be viewed as a *normalised score* instead of a probability, per se, due to the low values. In this study the output of the ST function was still referred to as a probability.

For the map-matching example used in this section, the 8 closest links were used to identify candidate links. The candidate links identified for GPS point 2 are shown in Table 2 and the candidate link names can be referenced from Figure 6.

Candidate link	Distance to link	Observation probability
l_2^1	7.08	0.0187
l_2^2	7.08	0.0187
l_2^3	8.15	0.0184
l_2^4	8.15	0.0184
l_2^5	19.08	0.127
l_2^6	19.08	0.127
l_2^7	19.08	0.127
l_2^8	19.08	0.127

Table 2: *GPS point candidate links for GPS point 2.*

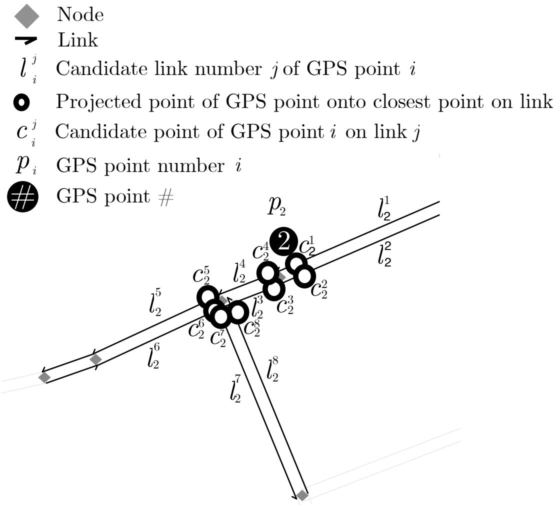

Node
Link
l_i^j Candidate link number j of GPS point i
Projected point of GPS point onto closest point on link
c_i^j Candidate point of GPS point i on link j
p_i GPS point number i
GPS point #
p_2
2
l_2^1
l_2^2
l_2^3
l_2^4
l_2^5
l_2^6
l_2^7
l_2^8
c_2^1
c_2^2
c_2^3
c_2^4
c_2^5
c_2^6
c_2^7
c_2^8

Figure 6: *Example of candidate links.*

Since the observation probability calculation does not take into account the position context of a GPS point, it can sometimes lead to wrong matching results. Figure 7 shows such an example. The thick lines represent a highway, and the thin vertical line represents a local road. Although p_i is closer to c_i^1 on the local road, it should match p_i to c_i^2 on the highway if it is already known that p_i's neighbours p_{i-1} and p_{i+1} are on the highway. This is based on the assumption that a vehicle is unlikely to take a roundabout path [5]. To prevent this potential error the algorithm makes use of the *transmission probability*.

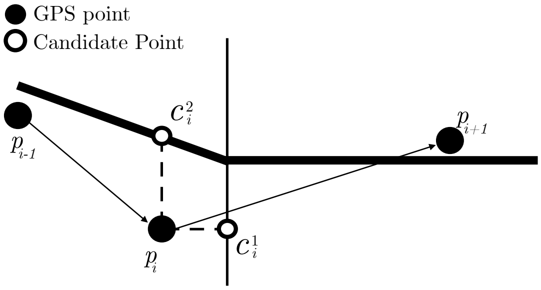


Figure 7: *Transmission probability example.*

Transmission probability: Given two candidate points c_{i-1}^t and c_i^s for two neighbouring GPS sampling points p_{i-1} and p_i respectively, the transmission probability from c_{i-1}^t to c_i^s is defined as the likelihood that the true path from p_{i-1} to p_i follows the shortest path from c_{i-1}^t and c_i^s. The transmission probability is defined as

$$V(c_{i-1}^t \to c_i^s) = \frac{d_{i-1\to i}}{w_{(i-1,t)\to(i,s)}}t, \tag{7}$$

where $d_{i-1 \to i} = dist(p_i, p_{i-1})$ is the Euclidean distance between p_i and p_{i-1}, and $w_{(i-1,t) \to (i,s)}$ is the length of shortest path from c_{i-1}^t to c_i^s.

Combining equations (6) and (7) the spatial analysis function, denoted by $F_s(c_{i-1}^t \to c_i^s)$, can be calculated as the product of observation probability and transmission probability:

$$F_s(c_{i-1}^t \to c_i^s) = N(c_i^s) * V(c_{i-1}^t \to c_i^s), \qquad \forall\, 2 \leq i \leq n \tag{8}$$

where c_{i-1}^t and c_i^s are any two candidate points for two neighbouring GPS points p_{i-1} and p_i respectively.

Expression (8) computes the likelihood that an object moves from c_{i-1}^t to c_i^s using the product of two probability functions; thus, geometric and topological information are both taken into consideration. Note that in practice, it is unlikely for a moving object to always strictly follow the shortest path. Therefore the observation probability $N(c_i^s)$ cannot be omitted from (8).

With spatial analysis, for any two neighbouring GPS points p_{i-1} and p_i, a set of candidate paths $c_{i-1}^t \to c_i^s$ are generated. Each path is assigned a spatial measurement value computed from (8).

There exists a shortcoming inherent to low-sampling rate GPS data, because even though the map-matching algorithm determines the most likely link for every GPS point, the assumption is still that for two links found for subsequent GPS points not connected to each other, the links traversed between the two links is the shortest path. In real life very similar paths, in terms of distance, might exist between two links or a driver might take a non-shortest path route and still generate points at similar positions in the network. The temporal analysis aims to address part of this problem but without higher sampling rates there is no accurate way of identifying the correct route if many alternatives exist between two GPS points.

3.4 Temporal analysis

Although the spatial analysis can determine the actual path from the other candidate paths relatively accurately, there are situations where the spatial analysis might choose an unlikely path. When two paths that allow very different travel speeds are situated next to each other, spatial analysis might determine a path that spatially makes sense but when considering the speed at which the agent would travel on the chosen path, this is highly unlikely. The inferred speed might seem unlikely due to either the speed restrictions or the practical speed at which an agent would have to travel to reach the next candidate point within the given time between the subsequent samples. See the example shown in Figure 8. The two lines on the left represent a highway, and the lines to the right are roads inside a residential area. The spatial analysis function may produce the same value whether two points p_{i-1} and p_i are matched to the highway or to the residential road. However, if one calculates the average speed from p_{i-1} to p_i on a straight line as $100\,km/h$, the algorithm would match the points to the highway considering the speed limits of the residential road.

More formally, given two candidate points c_{i-1}^t and c_i^s for two neighbouring GPS sampling points p_{i-1} and p_i respectively, the shortest path from c_{i-1}^t to c_i^s is denoted by a list of road

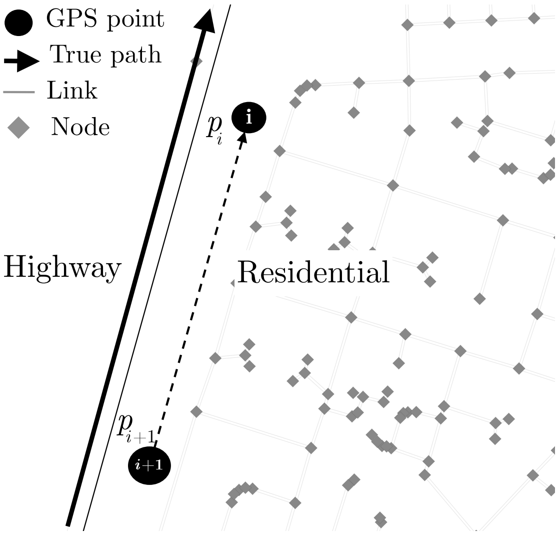


Figure 8: *Temporal analysis example.*

segments $[l^{'}_1, l^{'}_2, \ldots, l^{'}_k]$. The average speed $\overline{v}_{(i-1,t)\to(i,s)}$ of the shortest path is computed as

$$\overline{v}_{(i-1,t)\to(i,s)} = \frac{\sum_{u=1}^{k} \ell_u}{\Delta t_{i-1\to i}}, \tag{9}$$

where $\ell_u = l_u^{'length}$, the length of $l^{'}_u$, and $\Delta t_{i-1\to i} = p_i^t - p_{i-1}^t$, the time interval between two sampling points p_i and p_{i-1}.

Note that each road link $l^{'}_u$ is also associated with a typical speed value $l_u^{'fs}$, referred to as the free speed of a link. One of the benefits of using a MATSim network is that the free speed can be a function of the time. One is therefore able to incorporate more accurate free speed data based on time of day to evaluate the temporal probability of the algorithm.

The cosine distance is used to test the similarity between the actual average speed from c_{i-1}^t to c_i^s and the speed constraints of the path between the two points. Consider the vector that contains k elements of the same value $\overline{v}_{(i-1,t)\to(i,s)}$ and the vector $(l_1^{'fs}, l_{21}^{'fs}, \ldots, l_k^{'fs})^T$.

Temporal analysis: The temporal analysis function is defined by

$$F_t(c_{i-1}^t \to c_i^s) = \frac{\sum_{u=1}^{k}(l_u^{'fs} \times \overline{v}_{(i-1,t)\to(i,s)})}{\sqrt{\sum_{u=1}^{k}(l_u^{'fs})^2} \times \sqrt{\sum_{u=1}^{k} \overline{v}^2_{(i-1,t)\to(i,s))}}}. \tag{10}$$

As in the spatial analysis function, c_{i-1}^t and c_i^s are any two candidate points for p_{i-1} and p_i respectively.

3.5 Result matching

Once the spatial and temporal analyses have been completed, a candidate graph $\boldsymbol{G}'_T$ is generated with every candidate point for every GPS point representing a node in the graph and every link in the graph representing the movement between the neighbouring candidate points.

Formally the candidate graph for trajectory $\boldsymbol{T} = \{p_1, p_2, \ldots, p_n\}$ is $\boldsymbol{G}'_T(\boldsymbol{L}'_T, \boldsymbol{N}'_T)$, where $\boldsymbol{N}'_T$ is a set of candidate points for each GPS sampling point, and $\boldsymbol{L}'_T$ is a set of links representing the possibility of moving from one candidate link to another, as depicted in Figure 9. Each link also contains attributes for the connection between the two nodes, namely the value of the ST algorithm's probability that the object traversed from the one candidate point to the other, stored as the length of the link. The link also stores some metadata such as the shortest path on the original road network between the candidate links if they are not incident with a common node. The shortest path is used to reconstruct the inferred route once the most probable candidate points have been identified. Each node in $\boldsymbol{G}'$ also stores the observation probability $N(c_i^s)$ and each link the average speed, $V(c_{i-1}^t \rightarrow c_i^s)$ and temporal analysis $F_t(c_{i-1}^t \rightarrow c_i^s)$ for detailed analysis afterwards.

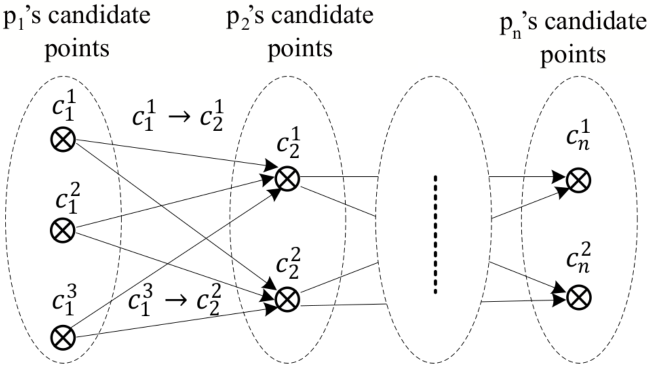


Figure 9: *Schematic illustration of candidate graph* $\boldsymbol{G}'_T(\boldsymbol{L}'_T, \boldsymbol{N}'_T)$.

Lou *et al.* [5] combine the spatial and temporal analysis functions and define the *ST function* for $c_{i-1}^t \rightarrow c_i^s$ as:

$$F\left(c_{i-1}^t \rightarrow c_i^s\right) = F_s\left(c_{i-1}^t \rightarrow c_i^s\right) \times F_t\left(c_{i-1}^t \rightarrow c_i^s\right) \qquad \forall\ 2 \leq i \leq n. \tag{11}$$

A candidate path sequence for the entire trajectory $\boldsymbol{T}_{\mathrm{SAA}}$ is a path in the candidate graph, denoted by $\boldsymbol{P}_c = \{c_1^{s1}, c_2^{s2}, \ldots, c_n^{sn}\}$. The overall score for such a candidate sequence is $F(\boldsymbol{P}_c) = \sum_{i=2}^n F(c_{i-1}^t \rightarrow c_i^s)$. From all the candidate sequences the aim is to find the one with the highest overall score as the best matching path for the trajectory. More formally, the best matching path $\boldsymbol{P}$ for a trajectory $\boldsymbol{T}$ is selected as

$$\boldsymbol{P} = \underset{\boldsymbol{P}_c \in \boldsymbol{G}'_T(\boldsymbol{L}'_T, \boldsymbol{N}'_T)}{\arg\max} F(\boldsymbol{P}_c). \tag{12}$$

To obtain a candidate sequence with the highest overall score, the entire graph needs to be traversed for all the possible combination of routes from the start to the end point.

The score of a route is the the sum of the link lengths in a route where the link length is the ST algorithm's score for the link being the correct link which the agent traversed while generating the GPS points. This process can be executed using existing graph theoretic algorithms like Dijkstra's algorithm. But since Dijkstra's algorithm uses the shortest path as an objective function, instead of using the length of the links as the travel distance utility this implementation uses $1 - l^{\text{length}}$. Since the length of the link is the ST algorithm's probability that the object traversed from the one candidate point to the other, *i.e.* the probability that it is the right route, Dijkstra's algorithm is actually calculating the routes that is least likely to *not* be the wrong route. Changing Dijkstra's algorithm to essentially calculate the longest route is only possible because the graph is a directed acyclic graph, that is, it is a finite directed graph with no directed cycles, thus no infinite loops will occur when looking for a longest path.

The Dijkstra method requires a single starting and ending node for the algorithm to start at, and navigate to. Since the graph has multiple starting points, for point p_1, and multiple end points, for point p_n, a dummy starting node and an ending node were created. The starting dummy node was connected to the first tier of nodes with a length equal to the observation probability of the node it was connected to. The ending dummy node was connected to the last tier of nodes with length 1.

Figure 10 contains a graphical representation of the map-matching example problem in Figure 2. Every tier in the graph, excluding the starting and ending node, represents the eight possible candidate points for each GPS point, $p1 \rightarrow p5$. Once the longest path has been calculated from the graph, the original links from the road network can be inferred from the graph link and node data.

In Table 3, the probability for each chosen link can be seen as well as the average link probability, which was used to calculate the overall probability of the IP. Graph link F has a length of 1 and did not form part of the calculation as it was only used to create an end point for the Dijkstra algorithm.

Graph link	ST probability
A	0.0159
B	0.0083
C	0.0119
D	0.0153
E	0.0079
Average	0.0119
Min	0.0079
Max	0.0159
Std Dev	0.0033

Table 3: *Graph link ST probability data.*

3.6 Analysing results

Since a TP is available in this example, one can analyse the results of the algorithm using the criteria defined in Section 2.2 as per Table 4.

To analyse the results of the algorithm in the absence of a TP one can review the inferred

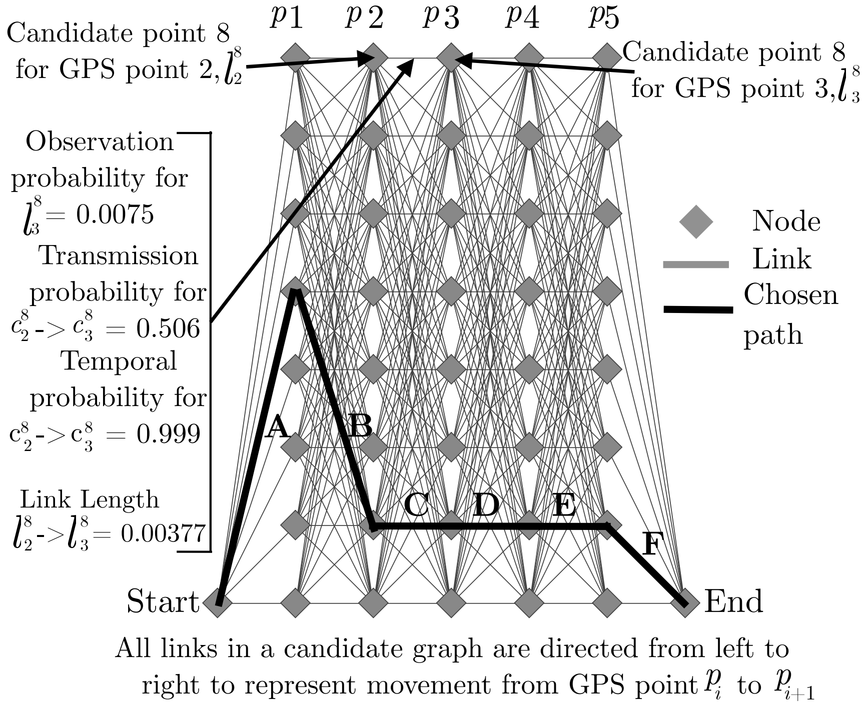
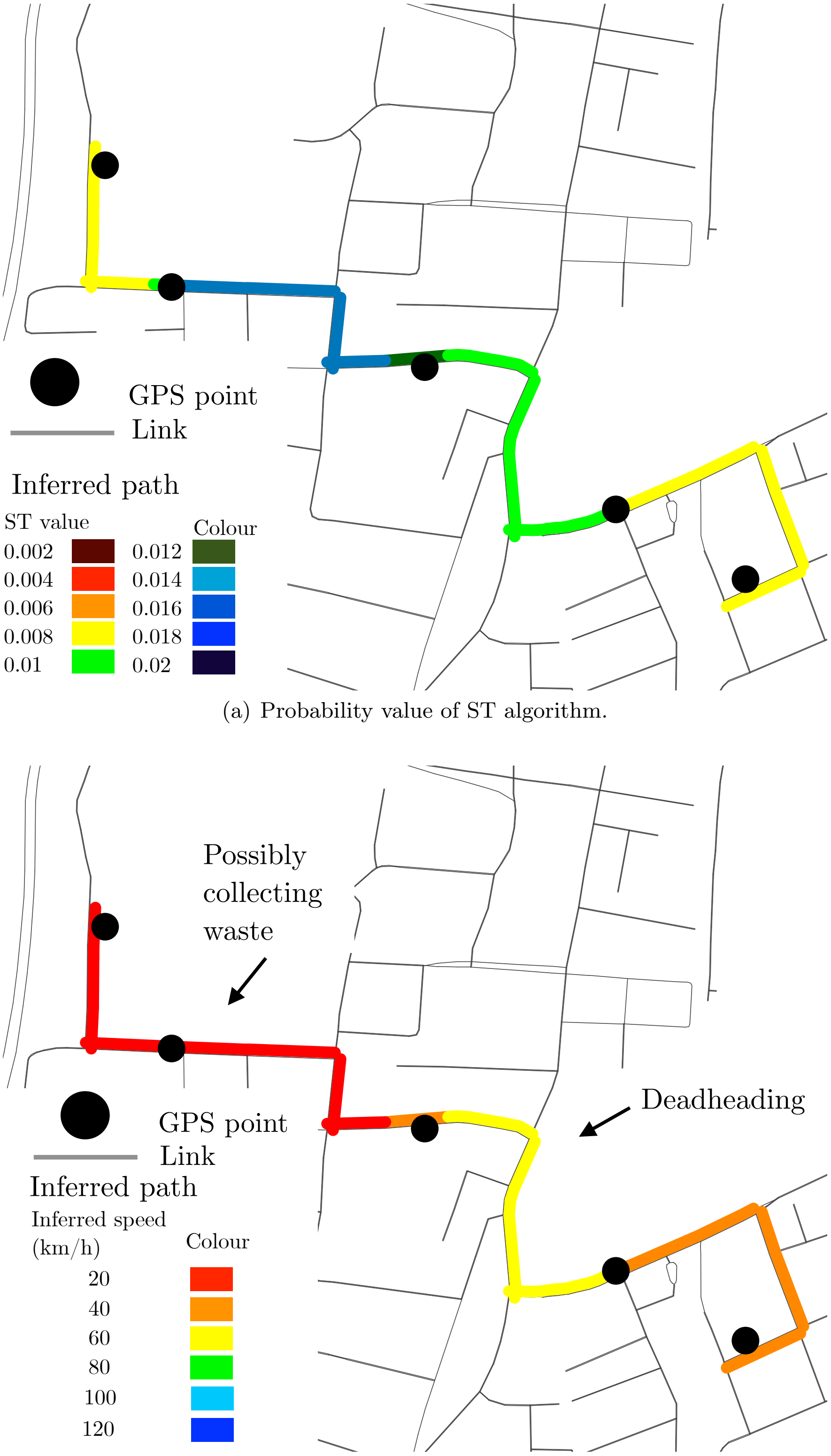

$p1$ $p2$ $p3$ $p4$ $p5$
Candidate point 8 for GPS point 2, l_2^8
Candidate point 8 for GPS point 3, l_3^8
Observation probability for l_3^8 = 0.0075
Transmission probability for c_2^8 -> c_3^8 = 0.506
Temporal probability for c_2^8 -> c_3^8 = 0.999
Link Length l_2^8 -> l_3^8 = 0.00377
A
B
C
D
E
F
Start
End
Node
Link
Chosen path
All links in a candidate graph are directed from left to right to represent movement from GPS point p_i to p_{i+1}

Figure 10: *Candidate graph for sample data.*

Measurement	Value
ARR	0.997
IARR	0
ARRn	0.977
Al	0.997
Average ST value	0.0119

Table 4: *Example results analysis against TP.*

speed as well as the ST probability value. The inferred speed can be calculated from the IP using the time stamps of the GPS points while the probabilities can be transferred from the graph links to the associated network links. Since subsequent GPS points can be allocated to candidate links that are not connected, some graph links contain a list of multiple network links representing the shortest path between the one candidate link and the subsequent candidate link. An original network link can also be associated with more than one graph link if the link is a candidate link for more than one GPS point. In order to assign a probability to the original network links, the probability of each graph link can be assigned to the candidate link and intermediate links with which it was associated. If there are shared original network links between graph links, an average between the probabilities and calculated speed was assigned to these links. The result of this analysis can be seen in Figure 11. From Figure 11(b), it can be inferred what the waste collection vehicle was doing on each part of its journey by using the speed as a proxy. It appears



(a) Probability value of ST algorithm.

(b) Inferred speed of vehicle.

Figure 11: *Analysis of example map-matching results.*

from the red lines that the vehicle might have been collecting waste, while the yellow lines imply it was most likely deadheading, *i.e.* driving to a new service area and not collecting any waste. This simple analysis of inferred speed answers the specific question posed by the use case of this study.

To gauge some *confidence level* of the inferred route one can analyse the assigned probabilities from the ST algorithm, as per Figure 11(a). But without further experimentation to create a benchmark for probabilities versus accuracy, it is difficult to conclude a (formal) confidence level in the results purely based on the probabilities of different route sections. It should be noted that low speeds travelled by the vehicle would lead to lower probabilities because the temporal part of the ST algorithm analyses the calculated speed against the free speed of the network and the more it differs, the the lower the value. This poses a potentially significant impact on the analysis of waste collection vehicles since they regularly travel at very low speeds while servicing an area.

4 Results and discussion

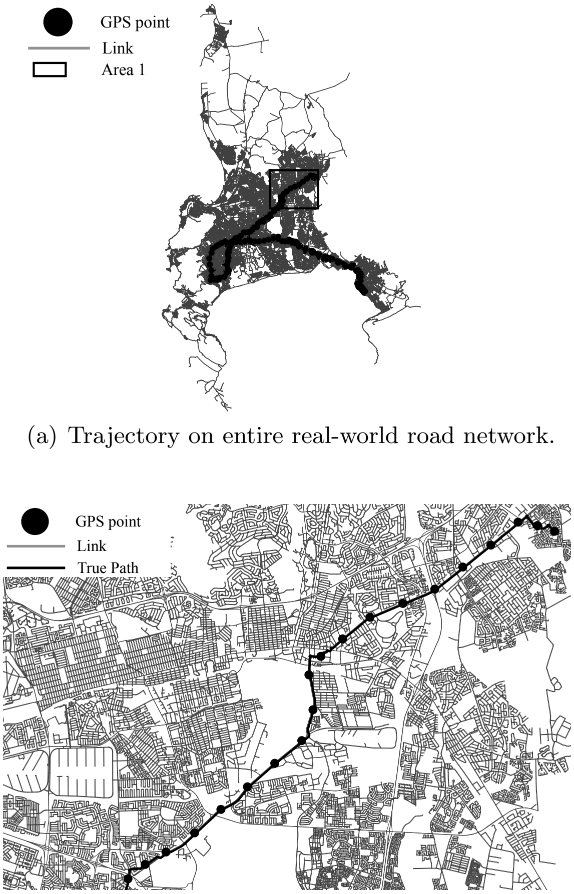
To test the effectiveness and efficiency of the map-matching algorithm, the authors needed to be able to test it in a controlled environment with representative data sets as well as a true path for every trajectory to evaluate the accuracy. Since actual data containing true paths for vehicle trajectories is hard to come by and may be fraught with data anomalies, the authors decided to generate data sets to test in controlled experiments.

Two networks were used for generating experiments. One was an elementary grid network with fixed free speed and lengths and the other was the actual road network of the City of Cape Town, South Africa. For both networks, random paths and corresponding random GPS points at different frequencies were generated. Experimentation proved that the elementary grid network was not only too simple to extrapolate findings to the real-world network, but also resulted in too many feasible options from the possible routes when generating low sampling rate trajectories, resulting in very low accuracies. The nature of the simple grid does not lend itself to providing options with a high variability in the probabilities as segments are all the same length and speed. Due to this only the experiments from the real-world network will be discussed. Future work could include developing more realistic grid networks with varying lengths and free speed and more variable trajectories in terms of the travel speed of the object as well as the sampling rate of the GPS generation.

4.1 Experiment setup

The real road network used in this study was sourced from OpenStreetMap, a collaborative project to create a free editable map of the world, and converted it to a MATSim road network object. The first step in generating experimental data was to generate a path a vehicle could take on the road network. The path generation algorithm used the Dijkstra method to find the shortest path between two random nodes within the network. To create more complex paths, the algorithm can perform this action a number of times and combine the paths by using the end node of the previous step as the starting node for the next step. Three random nodes were used in all paths generated.

Once the path has been generated, a second algorithm will take the path and plot random GPS points along the path based on a specified trajectory sampling rate and GPS error. The GPS trajectory-generating algorithm starts at the first node in the path and moves a distance on the path based on the free speed of the path and the sampling rate. A simulated GPS point is created by sampling a single distance from the distribution $N(\mu = 0, \sigma^2)$, and taking its absolute value to represent the distance from the actual point on the link to the GPS point. The direction of the point is based on an angle sampled from a uniform distribution in the range $[0, 2\pi]$. Using Pythagoras' theorem the coordinates for the GPS point is calculated and stored. Figure 12 shows the final output of a single trajectory generated on the road network.



(a) Trajectory on entire real-world road network.

(b) Details of area 1 of trajectory on real-world road network.

Figure 12: *Overview of experimental data generated on real-world network.*

4.2 Efficiency

Figure 13 shows that the algorithm execution time increases linearly with an increase in GPS points in the trajectory as well as that the rate of increase is higher the more candidate links are being evaluated. The efficiency of the algorithm decreased considerably in the real-world network compared to the simple grid network. In the simple grid experiments, trajectories with 600 GPS had an execution time of around 7 s while on the real-world network trajectories below 500 points had a duration of over 200 s. The main reason for this is that as the network size and complexity increases, the possible routes that have to be assessed for the temporal part of the algorithm increases. The temporal part of the algorithm uses the Dijkstra method, which is the most computationally expensive part of the algorithm. Further experiments on the simple grid network where only the network size increased proved that the execution time increased even though all other variables remained constant. This means that there exists a significant efficiency benefit to a user if the algorithm would only use the part of the network that is most likely to be matched to the GPS trajectory or if the user trimmed the network to include only parts that would most likely be used before applying the algorithm. Figure 12(a) illustrates just how little of the entire road network a trajectory is used. It is recommended that future work focus on developing such an algorithm.

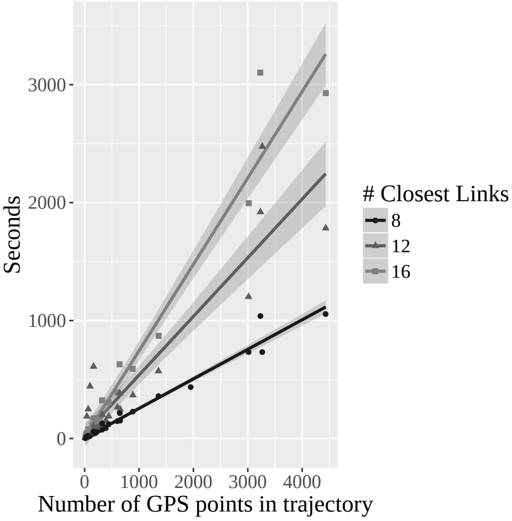


Figure 13: *Efficiency versus number of closest links.*

4.3 Effectiveness

Figure 14 illustrates that all GPS periods yielded a similar level of accuracy. However, GPS periods close to 1 had a slightly higher IARR because the higher density of GPS points caused possible U-turns to occur in the matched routes, resulting in more incorrect segments included in the IP.

Figure 15 is an example of an erroneous match the algorithm performed. Both points i and $i+1$ were recorded so close to the off- and on-ramp, respectively, that the observation- and spatial probabilities trumped the temporal probability. It should also be noted that the free speed on the highway as well as on the ramps are 90 km/h and the free speed on the side road is 60 km/h for only ±250 m, leading to a smaller effect of the temporal probability. Furthermore, due to the GPS error, if both these points were generated with

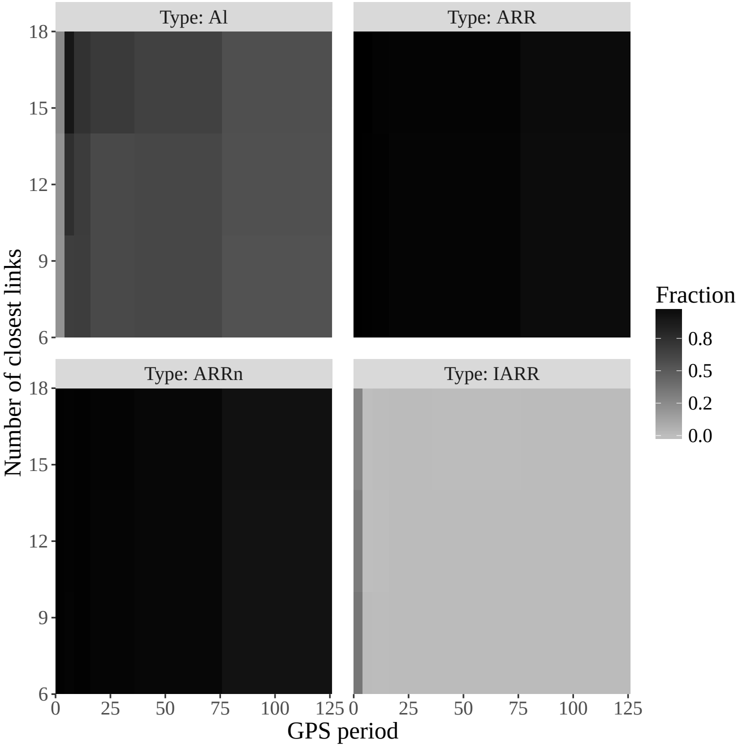


Figure 14: *Accuracy versus number of closest links.*

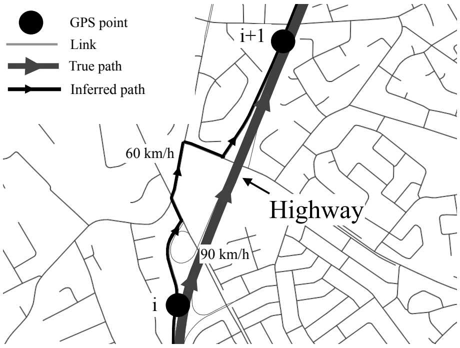


Figure 15: *Erroneous road segments identified.*

an offset in the direction of each other then the inferred speed could actually be lower than the highway speed and further diminish the effect of the temporal probability on the overall outcome. Future studies could possibly investigate whether assigning more weight to the spatial analysis versus the temporal analysis, or using alternative similarity measures for temporal analysis, can result in higher accuracies.

4.4 Speed analysis

The analysis of free speed versus the calculated speed in order to gauge the accuracy of the IP is a novel approach that could not be found in other studies. The calculation is to simply compare the calculated speed on every link versus the free speed of the link and express the delta as a ratio of the free speed. In Figure 16 the speed analysis on real-world network shows that the probability of choosing realistic road segments increases for trajectories when the GPS period is 5 s and starts to decrease significantly when the GPS period is 50 s. It also shows that when increasing the number of links for all GPS periods above 5 s it does not seem to make a significant difference in either the average or the variation of the deviations.

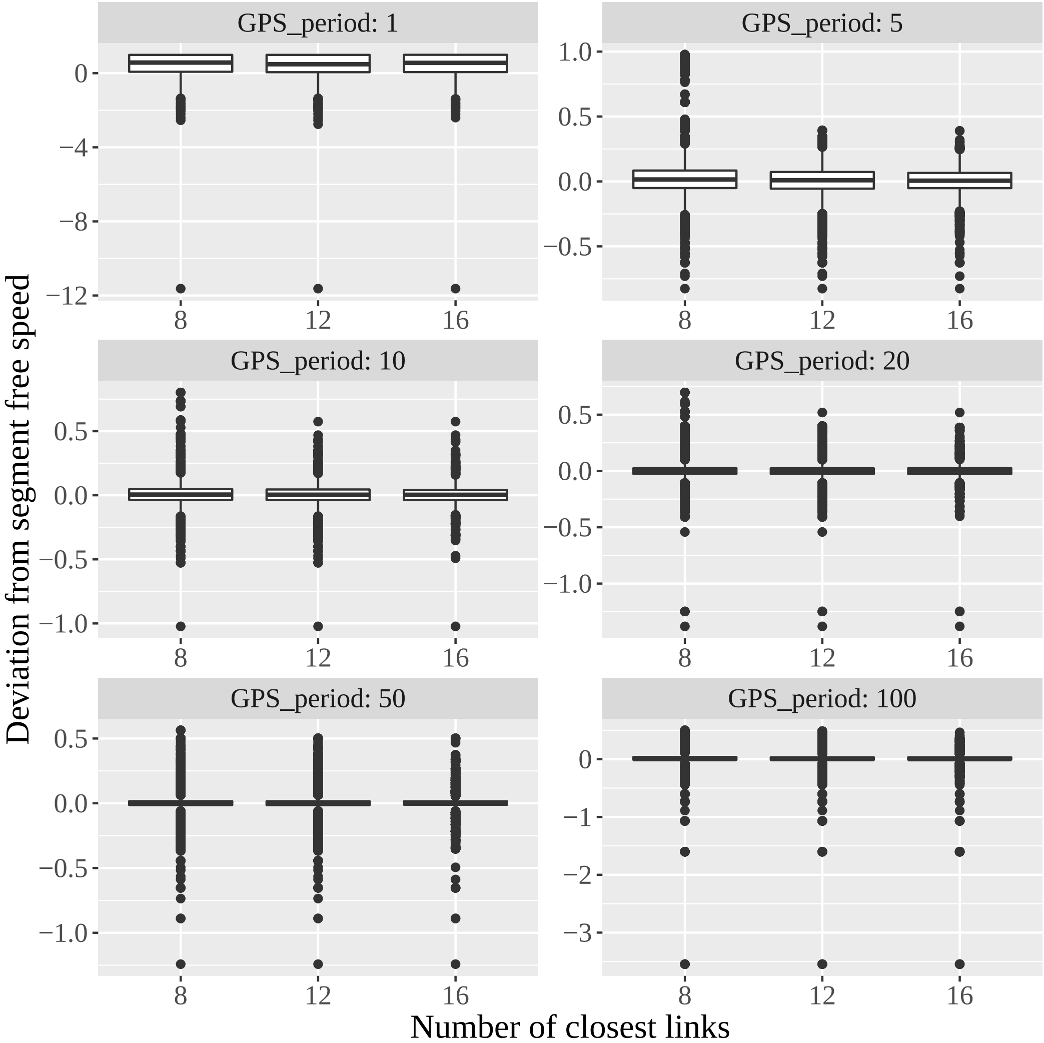


Figure 16: *Speed analysis versus number of closest links.*

4.5 Case study

The data set available for this study was the complete set of GPS trajectories for a specific waste collection vehicle during the period December 2013 to July 2014 operating in the City of Cape Town. The sampling rate for GPS trajectories were very dynamic and had no fixed time interval between records. The waste collection vehicle tracking system records GPS points more frequently if the vehicle is moving or experiencing acceleration or deceleration. If the vehicle is motionless, but the engine is running, the sampling rate decreases significantly, if the engine is switched off the recordings are 30 min apart. Significant data cleansing was done in order to group the data set into usable samples of single trajectories representing a specific trip.

Figure 17 displays the GPS points of a specific trajectory as well as the IP the algorithm matched. Since there is no TP to match the IP with, a subjective visual analysis of the match was conducted. The alternative options include manually creating a TP from the GPS points by making use of expert knowledge, for example asking experienced drivers to map the most likely route that a driver took in that area. This is similar to the approach discussed in Section 2.2. It is typical for a waste collection vehicle to travel from one service area to the next, called *deadheading*, close to or lower than the speed restriction of the road it is travelling on. Once they reach the service area, they will travel at a much lower speed to enable workers to load the vehicle with waste. The portions of the trip that the vehicle was most likely servicing an area, versus deadheading, can be identified by the road segments where it was travelling far below the free speed.

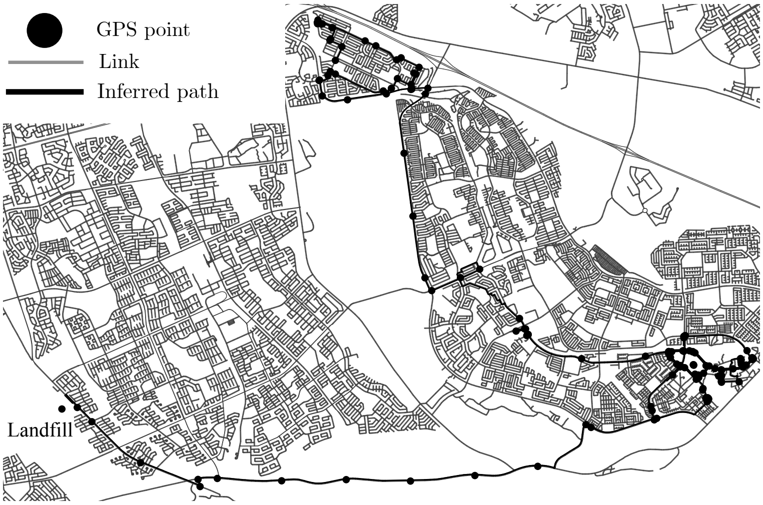
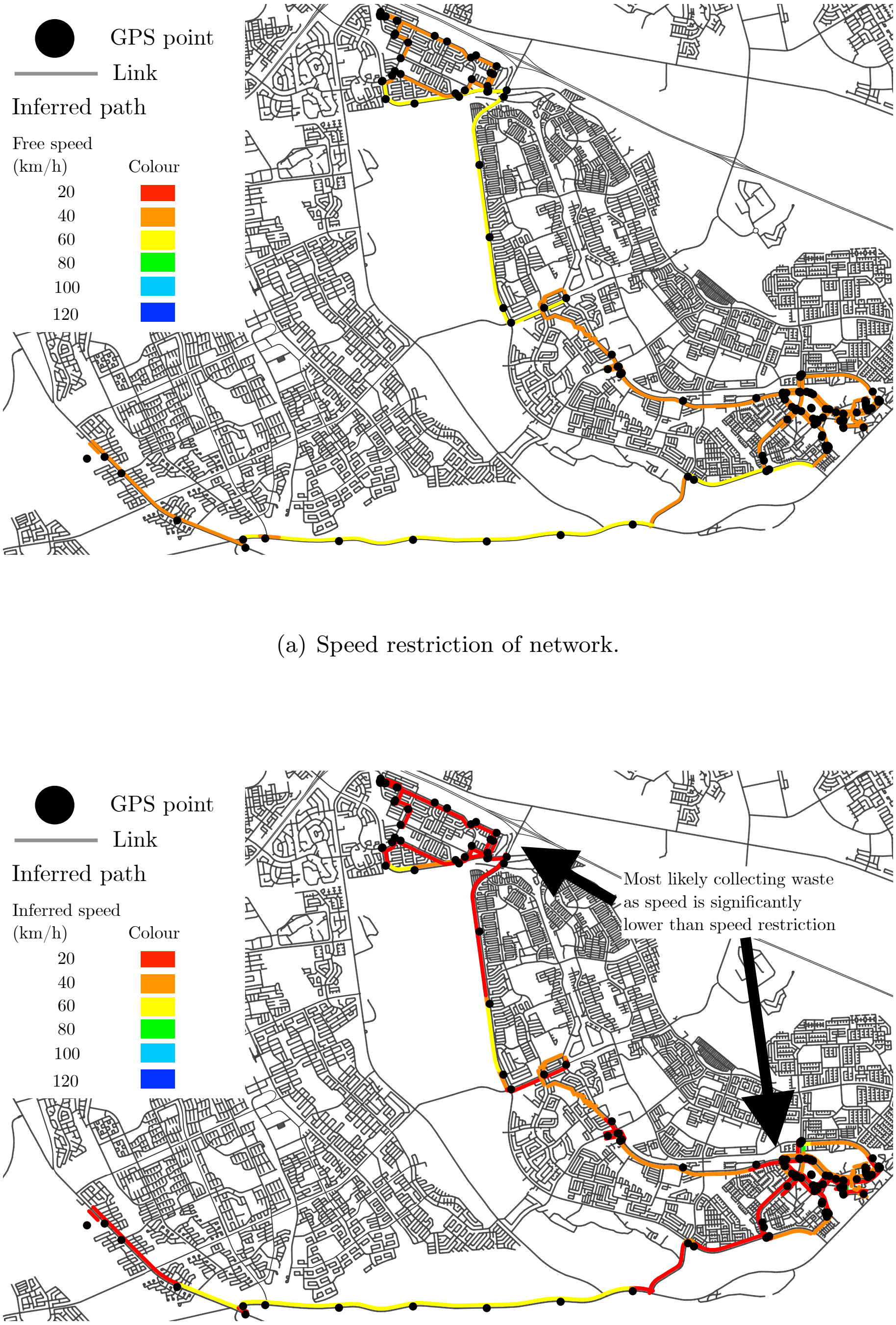


Figure 17: *IP of real-world data sample.*

Figure 18 contains a plot of the allowed free speed on the road network in the top image, and a comparison of the inferred travel speed to the free speed in the bottom image. This analysis illustrates the areas where the vehicle was most likely collecting waste by comparing the free speed of the links to the inferred speed. Figure 19 contains a comparative plot of the inferred speed to the free speed across the entire inferred route of the vehicle. It can be noted that there are some significant outliers in the inferred speed which most likely indicates incorrectly matched links. These discrepancies occurred where the GPS points were recorded within close proximity to each other and the algorithm incorrectly identified U-turns and loops in the matched route, as can be seen Figure 20(a). When comparing the abnormal inferred speeds to the probabilities it can be noted that the areas



(a) Speed restriction of network.

(b) Inferred speed of vehicle.

Figure 18: *Comparison of inferred speed versus free speed on the IP.*

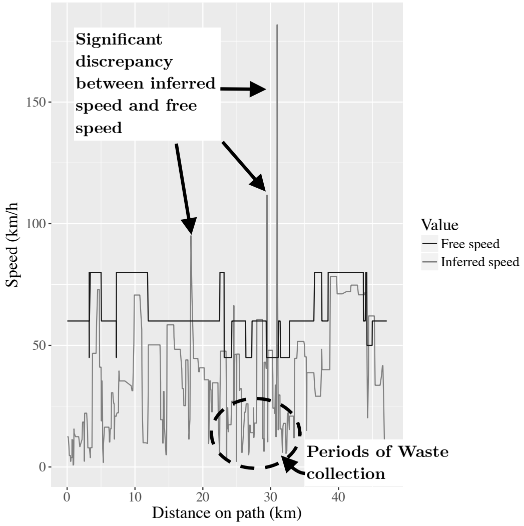


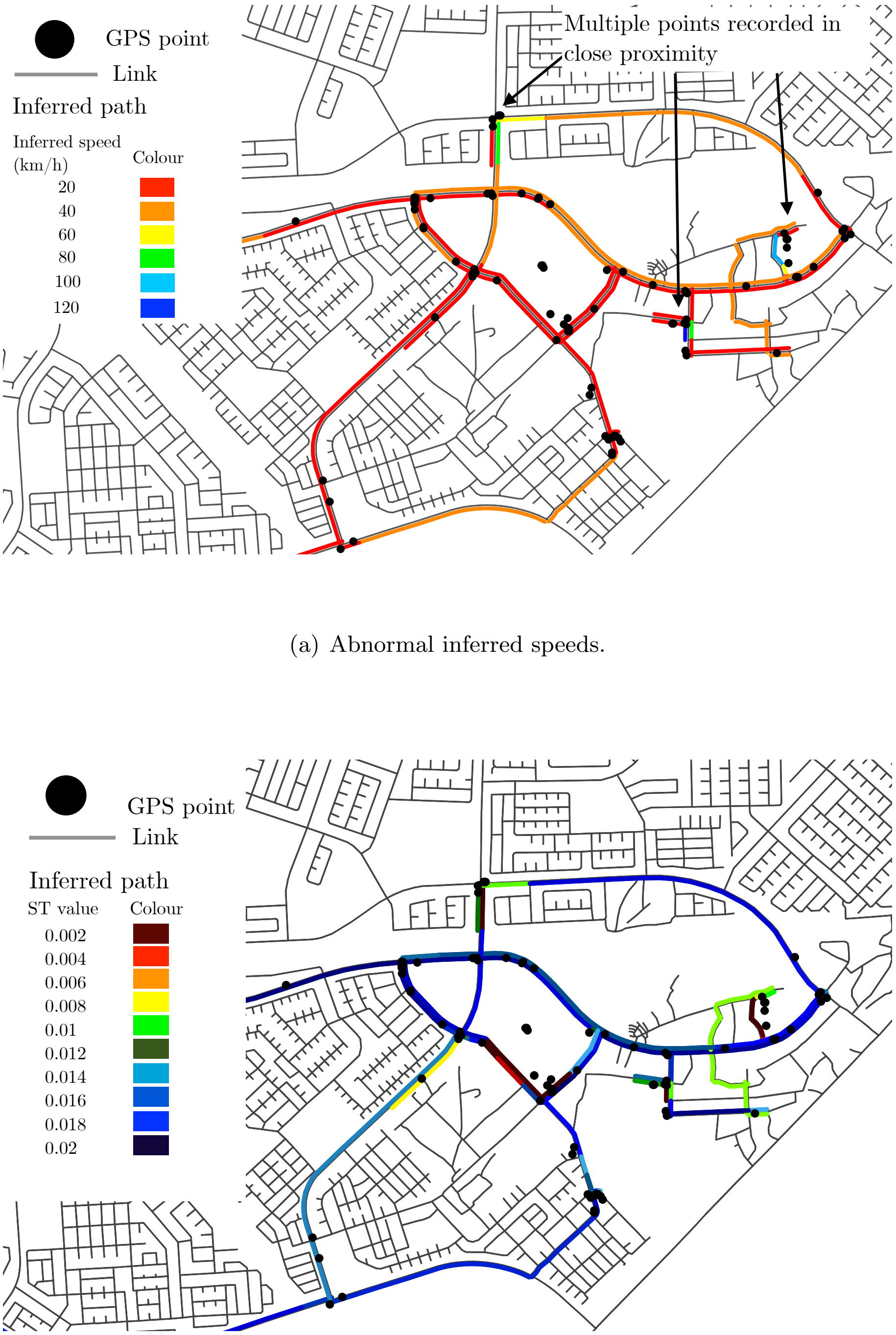
Figure 19: *Inferred speed versus free speed on inferred path.*

with abnormal inferred speeds also have low probabilities. There also appears to be sections with low probability values that do not have abnormal speeds and where the results seem credible.

The probability of incorrect matches occurring when data points are clustered close together is due to the inherent GPS error that can cause certain points to be recorded in such a way that it appears as if the vehicle travelled backwards, which can sometimes be the case with waste collection vehicles. The algorithm will determine that a U-turn was made and that a circular route is the only feasible solution. This phenomenon can either be addressed by cleaning up the data to remove unnecessary data points, which will also yield a decrease in computational time, or the algorithm should be improved to detect such situations and a sensitivity defined to ignore such cases.

5 Conclusion

The map-matching algorithm presented in this paper appears very robust in handling different GPS sampling rates but is not adept at using sampling rates of less than 5 s and will start to output U-turns and loops (cycles) in the IP when the sampling rate drops to 1 s. Using 8 or even 12 candidate links appears to give sufficient results with accuracies in the high 80% for both ARR and ARRn, and with very low IARR scores, up to GPS sampling rates of 125 s. Increasing the number of closest links to assess in the analysis increases the execution time considerably but his can be mitigated by reducing the road network to include only the part of the network that will most likely be used in the analysis.



(a) Abnormal inferred speeds.

(b) Probability analysis of abnormal inferred speeds.

Figure 20: *Analyses of abnormal inferred speeds*

In analysing the waste collection sample, the algorithm is able to match GPS trajectories to a subjectively acceptable level of accuracy, but it seems as if it cannot handle situations where GPS points are recorded in close proximity. The algorithm appears to be very robust and can handle dynamic sampling rates that vary over time, something very common in real data sets.

From the algorithm output, the authors are able to use the calculated speed as a proxy to infer where a waste collection vehicle was most likely collecting waste and where it was simply deadheading. This proves the algorithm's usefulness.

The study also provided a novel way of using the probabilities from the ST algorithm to analyse the results in the absence of a TP. The use of the probabilities together with the inferred speed versus free speed could provide substantial insight into the confidence level of the IP.

This comparison between inferred and free speed could be improved by making use of historical actual speeds on the network. The historical speeds can also be used in the temporal probability calculation which will provide the algorithm with a high degree of accuracy.

Future work on waste collection data will have to include a significant level of data cleaning to yield trajectories that only represent a single trip of an object and possibly remove or smooth out sections where points were generated in close proximity.

References

[1] Alt H, Efrat A, Rote G & Wenk C, 2003, *Matching planar maps*, Journal of Algorithms, **49(2)**, pp. 262–283.

[2] Brakatsoulas S, Pfoser D, Salas R & Wenk C, 2005, *On map-matching vehicle tracking data*, Proceedings of the 31st International Conference on very large data bases, pp. 853–864, ACM Digital Library, Trondheim, Norway.

[3] Ghiani G, Guerrieri A Antonio an Manni & Manni E, 2015, *Estimating travel and service times for automated route planning and service certification in municipal waste management*, Waste Management, **46**, pp. 40–46.

[4] Li Y, Huang Q, Kerber M, Zhang L & Guibas L, 2013, *Large-scale joint map matching of GPS traces*, 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 214–223, ACM SIGSPATIAL, ACM Digital Library, Orlando, Florida.

[5] Lou Y, Xie X, Zhang C, Wang W, Zheng Y & Huang Y, 2009, *Map-matching for low-sampling-rate GPS trajectories.* Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 352–361, ACM Digital Library, Seattle, Washinton.

[6] Lui K & Morikawa T, 2006, *An analysis of the cost efficiency of probe vehicle data at different transmission frequencies*, International Journal of ITS Research, **4(1)**, pp. 21–28.

[7] Miwa T, Kiuchi D, Yamamoto T & Morikawa T, 2012, *Development of map matching algorithm for low frequency probe data*, Transportation Reserach Part C, **22**, pp. 132–145.

[8] Quddus MA, Ochieng WY & Noland RB, 2006, *Integrity of map-matching algortihms*, Transportation Reserach Part C, **14(4)**, pp. 283–302.

[9] Qudus M & Washington S, 2015, *Shortest path and vehicle trajectory aided map-matching for low frequency GPS data*, Transportation Reserach Part C, **55**, pp. 328–329.

[10] Yin H & Wolfson O, 2004, *A weight-based map matching method in moving objects databases*, 16th International Conference on Scientific and Statistical Database Management, pp. 437–438, IEEE, Santorini Island, Greece.

[11] Ying JJC, Shi BN, Lan KC & Tseng VS, 2014, *Spatial-temporal mining for urban map-matching*, Tech. rep., UrbComp.

[12] Yuan J, Zheng Y, Zhang C, Xie X & Sun GZ, 2010, *An interactive-voting based map matching algorithm.* MDM '10 Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, pp. 43–52, IEEE Computer Society, IEEE Computer Society, Kansas City, MO, USA.

[13] Zheng K, Zheng Y, Xie X & Zhou X, 2012, *Reducing uncertainty of low-sampling-rate.* IEEE 28th International Conference on Data Engineering, pp. 1144–1155, IEEE Computer Society, Washington, DC, USA.