Mapping Passenger Trajectories to Train Schedules - industrial paper

Lorenzo Padoan
Motion Analytica s.r.l
Mestre, Italy
University of Padova
Padova, Italy
lorenzo.padoan@unipd.it
lorenzo.padoan@motionanalytica.com

Francesco Silvestri
University of Padova
Padova, Italy
francesco.silvestri@unipd.it

Bruno Zamengo
Motion Analytica s.r.l
Mestre, Italy
bruno.zamengo@motionanalytica.com

ABSTRACT

An important task in transportation studies is to accurately map a given set of trajectories representing moving individuals onto specific means of transportation, like trains or buses. In this paper, we consider the following problem: given a trajectory representing train stations visited by a passenger during a trip and a train schedule, extract the set of trains that have been taken by the individual during the trip. Specifically, we introduce a novel algorithm based on a Generalized Suffix Tree (GST) to efficiently link passenger trajectories to train schedules, addressing challenges like large data volumes and noisy input trajectories. Our method constructs a GST from train schedules and allows for integrating multiple schedules into a single searchable structure, enabling rapid and precise matching of trajectories to train routes. Although we use trains as an example, the approach can be used for other means like buses or trams. To analyze our solution, we construct a synthetic dataset of passenger trajectories built over the Italian train schedule; the datasets contain trajectories with transfers and noise both in timestamps and stations. The experimental analysis shows that our solution perfectly reconstructs noiseless trajectories even with transfers, and achieves an accuracy of at least 86% with noisy data.

CCS CONCEPTS

• **Information systems** → **Data structures**; **Location based services**; *Global positioning systems.*

KEYWORDS

Suffix tree, train-passenger match, mobility data

ACM Reference Format:
Lorenzo Padoan, Francesco Silvestri, and Bruno Zamengo. 2024. Mapping Passenger Trajectories to Train Schedules - industrial paper. In *Proceedings of (SIGSPATIAL '24).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

SIGSPATIAL '24, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Understanding and managing the flow of urban transportation is a critical aspect of modern city planning and personal mobility. A challenging problem is to efficiently map the movements of individuals onto public transportation resources, like trains, busses, or subways: this problem aids in optimizing operational capacities and planning future infrastructure developments. Traditionally, this task has relied on manual surveys and direct tracking methods, which are labor-intensive and thus expensive, prone to inaccuracies, and limited in scale [12]. In some public transportation systems[1], a passenger has to electronically register her/his one-ride ticket every time there is a change in the means of transportation: this simplifies the above task, but cannot capture passengers traveling with daily or seasonal passes or fare dodgers.

The proliferation of big data and advanced data processing technologies has significantly transformed numerous fields, including urban transportation management, and now allows us to address the above challenge more efficiently and effectively. The continuous recording of spatial-temporal data in smartphone applications and mobile network logs allows the collection of large sets of high-resolution (anonymized) trajectories of moving passengers. Furthermore, learning tasks detect whether an individual is moving by train, car, bike, or other means of transportation with a good approximation (see e.g. [1]).

In the context of railway systems, it is also possible to extract the list of train stations visited by a passenger during a rail trip [10]. Therefore, it is possible to collect large amounts of trajectories, where each one represents the stations visited by a passenger during a rail route and the respective timestamps of arrival/departure for each station. The goal of this paper is to enrich this information with the one provided by public-available train schedules, providing the sequence of stations where a train has stopped with the respective arrival/departure time. More specifically, we aim to match each passenger trajectory with trains in a given schedule: that is, to determine the most likely train, or trains in case of transfers, taken by a passenger. We refer to this problem as *Train-Passenger Matching (TPM) problem.* This information will aid the construction of train origin/destination matrices, the identification of the most common routes, and the measurement of the "centrality" of a station in terms of how much it is used to make changes. Moreover, if the original data well captures significant parts of the population (e.g., in mobile data or navigation apps) it is possible to derive stronger indicators

[1]See e.g., Danish train, metro, and bus system https://www.rejsekort.dk/en/hjaelp/saadan-rejser-du

than those obtained with standard tools (e.g., surveys, electronic ticket stamp machines, ...) which may fall short in tracking the whole passenger journey. By providing these new insights, we may help railways companies and public managers optimizing time tables, gaining insights and optimizing connections and service lines.

The first challenge in matching is that passengers might perform one or more transfers: starting with a train, then getting off a station for some time (ranging from a few seconds for a just-in-time transfer, to even hours for a lost just-in-time transfer), and then getting on a new train from the same station. We cannot easily distinguish a transfer from just a train stop, since the transfer stations and the waiting time are unknown. Furthermore, there can exist a train that spatially matches the passenger sequence, but might not match from a temporal point of view.

Moreover, passenger trajectories are noisy due to the nature of these data. Timestamps do not correspond to the train's scheduled time: for instance, the arriving time of a passenger in a station s might be a few minutes later than the scheduled arrival time of the train in station s due to train delay. Moreover, a train station where a passenger stopped might not be present in the trajectory (e.g., due to a small stop time of the train) or register with the wrong identifier (e.g., due to two close train stations like "Gare du Nord" and "Gare de l'Est" in Paris). Finally, a trajectory might contain a station where the passenger didn't stop (e.g., when a fast train crosses without stopping a minor train station).

It is crucial to derive efficient methods for supporting the train-passenger matching problem with large collections of noisy trajectories. We thus propose an adaptation of the Generalized Suffix Tree (GST), a data structure for supporting string search and traditionally utilized in bioinformatics for managing genetic sequences [2]. Our work explores the novel application of GSTs to effectively manage and query complex datasets comprising train schedules and passenger travel itineraries. We first show how to encode the information of a train schedule within a GST. Then, we design a query procedure that allows, for each passenger trajectory, to reconstruct the trains taken by the passenger; the query procedure can handle inherent noise and inaccuracies from passenger trajectories. As the query requires $O(\ell)$ time for processing a trajectory of ℓ stations, the approach can efficiently process massive volumes of trajectories. The effectiveness of our algorithm is demonstrated through a series of experiments with a synthetic dataset that shows that our solution is to perfectly reconstruct noiseless individual trajectory even with a train transfer, and an accuracy ranging from 86% to 93% with different kinds of noise in individual trajectories (i.e., temporal and spatial noise).

To test our method, we construct a synthetic dataset of passenger trajectories with 10 000 rides based on the Italian train schedule of Spring 2024, covering the over 30 companies operating on the Italian rail system. The dataset contains both synthetic passenger trajectories with and without transfer, that have been created by randomly selecting train rides from the Italian train schedule. For each trajectory we store the exact noise-less trajectory that works as ground-truth, and a noisy version: the noise is both on the timestamps and on the station identifiers. As true datasets are usually not publicly available due to privacy constraints, we remark that the dataset is of independent interest.

While our primary focus is on train systems, the principles and methodologies of our approach could potentially be extended to other forms of public transportation, such as buses, trams and public transport in general. This adaptability could make it a versatile tool in the broader domain of urban mobility research. Moreover, the GST can be constructed from aggregated train schedules, enabling the integration of multiple schedules into a single searchable structure; this significantly enhances the ability to match individual travel sequences with potential train routes efficiently and accurately.

The paper is structured as follows: in Section 2 we describe previous works; in Section 3 we provide a formal definition of the problem; in Section 4 we detail the construction of the GST, describe the matching process used to align travel data with train schedules, and discuss the handling of various edge cases that typically complicate the matching process; in Section 6 we provide some experimental results using some synthetic datasets.

2 PREVIOUS WORK

The task of associating individuals with specific means of transportation involves the intersection of transportation studies, big data analysis, and machine learning techniques. Significant steps have been made in this domain, leveraging diverse data sources ranging from GPS and smartphone sensors to mobile network data and social media inputs. For a complete review of the results on the topic, we refer to the survey [1].

Most of the works have been using GPS data, with potentially the integration of other sensors for dealing with issues like signal interruptions and variable data quality. For instance, the paper [9] uses accelerometer features when the GPS trajectory cannot be reconstructed with sufficient accuracy. In the case of subways where popular localization methods such as GPS and Wi-Fi are not accurate, the work [6] proposes to use barometer measurements to estimate the motion state of the train by the amount of change in air pressure.

Another significant direction in this field uses cellular data from mobile phone services to detect public transportation modes. The approach in [7] uses cellular data with public transportation maps to effectively bypass the need for direct device-based data collection and public transportation sensors as people counter. In [5], the authors use aggregated, anonymized cell phone data to derive public transportation timetables, which could be correlated with individual data points to infer public transport means associations.

Some works have also used social media for analyzing crowd movements. In [3], by leveraging data from platforms like Twitter, the authors provide insights into crowd dynamics and transportation usage without traditional sensor data. This contribution illustrates the potential of non-traditional data sources in enhancing the understanding of transportation systems and could be integrated with other data types for comprehensive mobility analysis.

Concerning the analysis of train movements with big data, the paper [14] studies the correlation between the number of mobile devices connected to selected base stations near railway tracks and the actual ridership data, establishing an approach to estimate train ridership without direct access to traditional data sources. In industrial settings, big data from mobile data and GPS data have

been used by the Italian State Railways for studying passenger train flows and the methodology is described in [10]. To the best of our knowledge, no previous works have addressed how to reconstruct the sequence of train rides from a trajectory representing the stations visited by a passenger.

3 PROBLEM FORMULATION

The goal of this section is to provide a formal definition of the TPM problem. Let $\mathcal{S} = \{s_0, \dots s_{p-1}\}$ be the set of all train stations. We denote with $\mathcal{T} = \{t_0, \dots, t_{m-1}\}$ the scheduling of m trains: for each $0 \leq i < m$, each t_i represents the stations visited by train i with the respective departure time, specifically $t_i = ((s_{i,0}, \tau_{i,0}), \dots (s_{i,\ell-1}, \tau_{i,\ell-1}))$; the j-th pair $(s_{i,j}, \tau_{i,j})$ denotes that the j-th station is $s_{i,j} \in \mathcal{S}$ with departure time $\tau_{i,j}$ (for the last station, $\tau_{i,\ell}$ denotes the arrival time), and we have $\tau_{i,j} < \tau_{i,k}$ for any $j < k$. For notational simplicity, we assume that all trains visit $\ell > 1$ stations. For any $0 \leq i < m$ and $0 \leq o < d < \ell$, we define the o, d-*train segment* of t_i the subsequence of t_i starting from the pair containing station o and ending in the pair with station d.[2] Moreover, we say that a $s \in t$, where s is a station and a t a train, if there exists a pair (s', τ) in the sequence t for some $\tau > 0$.

We now denote with $\mathcal{P} = \{p_0, \dots, p_{n-1}\}$ the sequence of train stations visited by n passengers: for each $0 \leq i < n$, we have $p_i = \{(s_{i,0}, \tau_{i,0}), \dots (s_{i,\ell'-1}, \tau_{i,\ell'-1})\}$ which represents the train stations visited by passenger i. The syntax is equivalent to the one adopted for trains and we assume for notational simplicity that each passenger visits $\ell' > 1$ stations.

The *Train-Passenger Matching (TPM) problem* consists of determining the trains taken by each passenger, potentially including some transfers. In a transfer, a passenger p on train t gets off in a train station $s \in t$ and then gets on another train t' from the same station s. Specifically, the TPM problem requires to find, for each passenger i, the smallest set of c_i train segments $c(p_i) = \{t_{i_0}[o_{i_0}, d_{i_0}], \dots t_{i_{c_i-1}}[o_{i_{c_i-1}}, d_{i_{c_i-1,0}}]\}$ that cover the stations in p_i, with the following constraints:

- Transfer stations: for any $0 \leq j < c_i - 1$, $s_{i_j,d_{i_j}} = s_{i_{j+1},o_{i_{j+1}}}$.
- Covering of passenger's stations: for any pair $(s, \tau) \in p_i$, there exists exactly one segment t_j containing the pair (s, τ).

The above definition has however some limitations. The first one is that it does not support discrepancy between passenger and train timestamps: the time identifying when a passenger leaves a train station might differ from the time of the train where the passenger was. The first obvious reason is due to train delays: however, in this case, it would suffice to record the effective scheduling of a train, which is usually publicly available on train companies' website. However, even when the effective schedule is used, there can be a discrepancy due to the methods used to register passenger timestamps. For instance, in the case of passenger trajectories obtained from mobile operator data, the departure of a passenger can be registered when her/his phone connects to another cell tower which in general happens a few minutes after the train has left the station. The second limitation is due to errors in passenger trajectories: a station visited by a passenger might miss the trajectory, or it can be recorded with a wrong identification. For instance, when a train slowly passes without stopping at a train station s, the station might be erroneously recognized as a station where the passenger has stopped.

We therefore need to change the above problem definition to deal with this limitation. The (λ, δ, μ)-TPM problem requires to match each passenger with at most λ segments $c(p_i) = \{t_{i_0}[o_{i_0}, d_{i_0}], \dots t_{i_{c_i-1}}[o_{i_{c_i-1}}, d_{i_{c_i-1,0}}]\}$, with $c_i \leq \lambda$, that cover the stations in p_i, with the following constraints:

- Transfer stations: for any $0 \leq j < c_i - 1$, $s_{i_j,d_{i_j}} = s_{i_{j+1},o_{i_{j+1}}}$.
- Covering of passenger's stations: by removing at most μ station-timestamp pairs in p_i, then for any of the remaining pair $(s, \tau) \in p_i$, there exists exactly one segment t_j containing the pair (s, τ'). We call (s, τ') the matching pair of (s, τ)
- Timestamp mismatch: for any passenger pair (s_j, τ_j) matched with (s_j, τ'_j), let $\delta(j) = |\tau - \tau'|$ be the delay at station s_j; then, the average value of the delay $\tilde{\delta}$ over all the stations of passenger p_i (except the removed stations in the previous constraints) is at most δ.

We observe that other timestamp measures could be changed to be more consistent with additional information given by the source of data: we could for instance give more emphasis to the initial and final train stations, or different weights to positive or negative delays. For instance, in the case of mobile network data, a negative indicator might indicate that a passenger is not on the train since he/she left the station before the train.

4 PROPOSED METHODOLOGY

In this section, we provide our approach to the TPM problem. The proposed method introduces the application of suffix trees in the domain of transport systems, specifically trains. This adaptation leverages the inherent efficiency of suffix trees in processing sequential data to manage the dynamic and voluminous nature of train schedules and individual trajectory data. By employing a generalized suffix tree (GST), we can efficiently index and query large datasets, establishing it as an efficient and scalable approach to match trajectories with train schedules. This technique not only promises substantial improvements in data processing speeds but also enhances the accuracy of matches, thereby offering profound implications for transportation planning and the analysis of urban mobility patterns. Furthermore, this matching methodology represents an efficient data-enrichment strategy by integrating the train information (i.e. local train vs high-speed train) to the otherwise unlabeled trajectory.

Consider a set of n strings $s_0, s_1, \dots s_{n-1}$. A Generalized Suffix Tree (GST) is a tree containing all suffixes of the n strings. More specifically, the tree contains the suffixes of $s_0\$_0, s_1\$_1, \dots s_{n-1}\$_{n-1}$, where $\$_i$ denotes a special string to avoid that a suffix of a string is merged with the suffix of another one. The GST has been widely studied and it is known how to construct them in $O(m)$ space and time, where m is the total number of characters in the n strings; we refer to [4] for further details.

The algorithm for mapping passenger trajectories to train schedules is divided into two key phases: the Building Phase and the Matching Phase. These phases collectively enable the efficient and

[2] We note that if we require a o, d-segment and station o (similarly for d) appears more times in the sequence, the segment should define which occurrence of o we should take as origin. However, we ignore this fact for notational simplicity.

accurate matching of individual travel patterns to specific train routes, leveraging the structure of GST.

In the Building Phase, the focus is on constructing the GST from the train schedule dataset. The Matching Phase utilizes the GST to align individual passenger trajectories with the most likely train routes. This phase starts by processing each passenger's sequence of visited stations, navigating the GST from the root node to match these sequences against the indexed train routes. Among the candidates, the time dimension will be used to select the most likely. The Matching phase will also deal with failed station matches: if the passenger sequence is not found due to a station mismatch, then the mismatched station will be removed. Then the search will be repeated from the beginning on the remaining part of the sequence and, according to the results, we will tackle two situations: the mismatch is only due to an error, or the mismatch is due to a train transfer. These phases work in tandem to achieve high accuracy in matching travel users to train routes. The detailed explanation of each phase follows, delving into the specific processes used in the construction and querying of the GST.

4.1 Building Phase

The building phase focuses on constructing a GST from the train schedule dataset. In this dataset, each train is represented as a sequence of stations it visits during its route. This sequential data is critical as each train's journey can be viewed as a string where each station is a character in the string.

To efficiently handle this data, we process the input, to extract each train's itinerary. The itineraries detail the sequence of stations for each train focusing solely on the spatial sequence of stops: specific schedule data will be used in the matching phase to choose the most likely train among the set of trains that share the same itinerary. Although not using time information during a search might increase the number of candidate trains, it significantly simplifies the tree construction and query.

During the construction of the GST, we treat each train's route as a continuous string of station identifiers. This technique allows the GST to index every possible sub-route, which are the sub-sequences of stations that trains pass through during their journeys. Each sub-route is represented within the tree structure, where each node corresponds to a specific sequence of stations traveled.

One of the unique capabilities of the GST is its efficiency in indexing all possible sub-sequences within the train routes. By encoding the routes into the tree, we enable the GST to function as a queryable repository of all the routes taken by different trains. Each node in the tree not only represents a sequence of stations but also stores the list of all train IDs that have traveled through that particular sequence of stations up to that point.

This metadata is crucial as it transforms each node into a rich information hub within the GST, linking station sequences to specific train identifiers. This setup enhances the tree's utility in the subsequent phase.

4.2 Matching Phase

The matching phase utilizes the GST to align individual trajectories with the most likely train routes. This phase begins by processing each trajectory, where a trajectory is the sequence of stations visited by a passenger. The analysis starts at the GST's root node, serving as the starting point for navigating through potential paths that match each passenger's travel pattern to a train. During the matching process, each station in an individual trajectory is examined against the nodes in the GST. The algorithm searches for a child node that matches the current station in the sequence. If such a node is found, it suggests a possible continuation for a train route including this station, allowing the traversal to proceed deeper into the tree. This step is repeated for each station until either the entire sequence is matched within the tree or a point is reached where no further matches are found, indicating a discrepancy between the passenger's path and the recorded train routes.

Upon completing the traversal, the node where the sequence ends contains the list of all train IDs that have traversed the matched sequence of stations. These train IDs are potential candidates for having transported the passenger.

Selecting the most probable train among these candidates involves an analysis based on temporal alignment between the individual's and the train's schedules. Specifically, the selection criterion is the train segment that minimizes the average time difference between the passenger's departure times and the train's scheduled departure times for all matched stations. This is represented as:

$$\tilde{\delta} = \min \left(\frac{1}{\eta} \sum_{i=1}^{\eta} \left| \tau_{\text{passenger},i} - \tau_{\text{train},i} \right| \right)$$

Where:

- η is the number of stations matched in the passenger's trajectory.
- $\tau_{\text{passenger},i}$ is the departure time of the passenger at station $i \in 0, \ldots \ell' - 2$ and the arrival time of the passenger at station ℓ'.
- $\tau_{\text{train},i}$ is the scheduled departure time of the train at station $i \in 0, \ldots \ell' - 2$ and the scheduled arrival time of the train at station ℓ'.

The train with the smallest value of $\tilde{\delta}$ is considered the most likely one that the passenger took, as it indicates the closest departure times to those experienced by the passenger across the sequence of matched stations.

To address various complexities in matching train travelers to train schedules, it's essential to consider different scenarios that may arise during the matching process. These scenarios can vary significantly, from ideal cases involving straightforward, noise-free data to more complex situations involving noisy data or passenger journeys with transfers between multiple train routes. Here, we will decompose the cases into three specific scenarios:

4.2.1 ***Match of a Single Trip Without Noise.*** In the scenario where a trip is made using only one train without any noise, meaning the data sequence of stations is clean and uninterrupted, the matching represents an ideal case where each station in a passenger's journey directly corresponds to nodes within the GST.

During this process, as the passenger's journey is traced through the GST, each transition from one station to the next one is expected to find a corresponding child node that matches the next station in the sequence. This uninterrupted following through the tree allows

the algorithm to efficiently navigate from the root to a leaf, or to a node that matches the final station in the passenger's journey.

Once the end of the journey is reached within the GST, the node where this sequence terminates contains a set of candidate trains. These candidates are all the trains that have a route passing through the exact sequence of stations as traveled by the passenger. The selection among these candidates then focuses on determining which train best aligns with the passenger's journey not just spatially but temporally as well.

To find the best match from the set of candidate trains the metric $\tilde{\delta}$ is used. The algorithm evaluates each candidate train in this manner, and the train that yields the smallest $\tilde{\delta}$ is selected as the most likely one that the passenger took. The selected train not only follows the exact route taken by the passenger but also aligns as closely as possible in terms of departure times at each station, providing a robust match.

4.2.2 ***Match of a Single Trip With Noise.*** The matching process in the scenario where a passenger trip made by a train contains noise, such as missing station entries or extraneous stops not aligned with any single train route, requires to be managed.

When the algorithm encounters a mismatch where the next station in the passenger's journey, denoted as s_{next}, does not correspond to any child node from the current node n, at this point, the last correctly matched node n_{last} before the mismatch is used to retrieve the set of candidate trains. These candidates represent possible continued journeys up to the point of mismatch, capturing all trains that could have been taken by the passenger up to the encountered noisy point.

Once this set of candidate trains is established, the matching process is restarted from the root of the GST. This reset allows the algorithm to attempt a fresh match starting from s_{next} in the passenger's sequence post mismatch. The journey is then traced forward from this new starting point, navigating through the GST to align the remaining part of the passenger's journey with potential train routes.

This process may result in multiple sets of candidate trains, especially if several breaks or mismatches occur throughout the journey. Each set represents a divergence in the passenger's recorded journey from the expected train paths in the GST, corresponding to each segment of the journey isolated by data noise or errors.

After the passenger's sequence, if more than one set of candidate trains is identified due to multiple breaks, the next task is to find an intersection among these sets. This intersection identifies trains that appear across all segmented matches, suggesting they could feasibly accommodate the entire journey despite the recorded discrepancies. If this intersection is not empty, the algorithm then evaluates these intersecting trains to determine the best fit. The selection of the best train from the intersecting set is based on the metric of the minimum average time distance, and potentially on the maximum number of allowed mismatches between passenger trajectory and the selected train.

4.2.3 ***Match of a Trip with a Train Transfer.*** In scenarios where a journey involves changing trains, the matching process in the GST encounters additional complexities that need to be handled. This case occurs when a passenger's journey spans multiple train routes, necessitating transitions at one or more transfer stations. The primary challenge here is to accurately match each segment of the passenger's journey to different train schedules while ensuring feasible connections between trains.

When processing a trip that includes train transfers, the algorithm needs to identify points where potential transfers might occur, typically when the passenger's trajectory diverges from any single train's route within the GST. These points of divergence are treated as breaks in the matching process, where each break indicates a likely train transfer. At each break, the last successfully matched node before the divergence plays a crucial role; it identifies the station where the train transfer is likely to occur. The set of train candidates associated with arriving at the last matched station is provided. At this point, the algorithm resets, and the matching phase is restarted from the root of the GST. The reset to the root is crucial because it allows the algorithm to comprehensively search from the beginning of the GST for all possible train routes that might align with the subsequent part of the passenger's journey. This approach ensures that all potential pathways are considered, providing a fresh perspective on the journey segments that follow, so the algorithm traces the journey forward from this new starting point, navigating through the GST to find another set of train candidates that could logically continue from the point of divergence.

In contrast to the previous scenarios involving noise, where intersections between sets of candidate trains might yield a viable route, the situation with train transfers results in an empty intersection between the initially identified candidates and those found after the reset step. This lack of overlap is due to the distinct segments of the journey likely being serviced by different trains with separate routes.

5 SYNTHETIC DATASET GENERATION

To evaluate the effectiveness of our algorithm, we construct synthetic datasets of passenger data derived from train schedules. This involves generating virtual passenger trajectories that realistically simulate the movement patterns of commuters across the train network. Each synthetic passenger's trajectory is crafted based on comprehensive train schedules, integrating factors such as varying departure times, multiple routes, and typical commuter behaviors. To enhance the realism of these datasets, we introduce noise in both spatial and temporal dimensions, simulating the inaccuracies and variability that occur in real-world data collection.

5.1 Train Scheduling dataset

The train schedules used in our study were collected from a publicly available resource called e656 [11]. This website offers a comprehensive and detailed collection of train schedules, which include the precise sequence of stations, along with the scheduled departure and arrival times for a vast array of trains.

Table 1 shows the train category in the used schedule; each category represents a specific type of train service, such as long-distance and regional trains. For each category, we report the number of trains and whether it represents high-speed trains or regular ones. Although categories are not directly used in this paper, we provide

	Train Category	Count	Speed
1	Regional	9891	Normal
2	Fast Regional	917	Normal
3	Suburbans	884	Normal
4	Intercity	387	Normal
5	sfm	360	Normal
6	Frecciarossa	287	High
7	Metropolitan	247	Normal
7	Regio Express	244	Normal
8	Regional Ciampino Airlink	230	Normal
9	Eurocity	155	Normal
10	Italo Av	134	High
11	Intercity Notte	132	Normal
12	Leonardo Express	127	Normal
13	Regional Tuscany Line	80	Normal
14	Regional Etruschi Line	47	Normal
15	Frecciargento	41	High
16	Regional Aterno Line	35	Normal
17	Regional Alpeadria Line	32	Normal
18	Malpensa Express	31	Normal
19	Regional Cilento Line	23	Normal
20	Regional Golfo Aranci Line	23	Normal
21	Regional Veloce Etruschi Line	22	Normal
22	Regional Veloce Tuscany Line	22	Normal
23	Historic Train	21	Normal
24	Regional Veloce Ponente Line	20	Normal
25	Euronotte	17	Normal
26	Regional Tropea Line	17	Normal
27	Regional Veloce Trenoblu Line	17	Normal
28	Frecciabianca	13	High
29	Tropea Express	13	Normal
30	Regional Genio Express	11	Normal
31	Regional Cerrano Line	10	Normal

Table 1: Train categories with speed classification

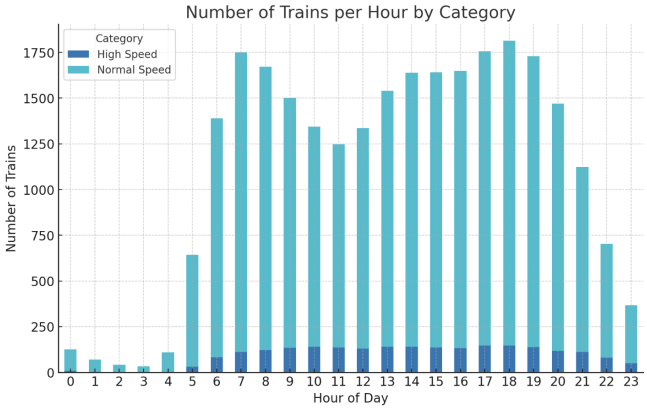

Figure 1: Number of Available Trains per hour. In general, no trains stop during the night.

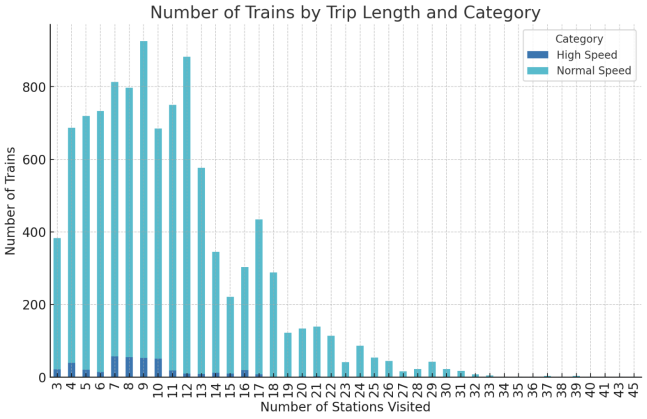

Figure 2: Number of stations in each train.

them both for completeness and for providing an example of data enrichment.

To simplify the various train categories, the following grouping was made: we grouped the fast train categories (Freccia and Italo, corresponding to 6, 10, 15, and 28 in Table 1) into the *high speed* (HS) category, while the remaining categories, generally lower-speed trains whose service lines are developed within a specific Italian administrative region, are grouped into the *regional* (REG) category. The total number of trains is 14 510, divided into 14 035 regional trains and 475 high-speed trains. After removing duplicates (appearing only in regional trains), we kept 10 888 trains.

Furthermore, an analysis of the number of trains available for each hour of the day was conducted. This analysis helps to understand the distribution of train services throughout the day, highlighting peak hours and periods of lower train availability. The results of this analysis are in Figure 1. The figure shows a significant increase in train numbers: starting in the early morning, peaking around mid-day and afternoon hours, and gradually decreasing towards the evening. This pattern reflects typical commuter behavior and the operational focus on accommodating peak travel times [13]. We also provide the distribution of the number of stations in each train trajectory in Figure 2, which shows that most trains stop in the range of 4 to 13 stations.

5.2 Synthetic Passenger Dataset

We utilize the above publicly available train scheduling data from the e656 [11] to generate a comprehensive range of synthetic passenger datasets, including trajectories with layovers. The creation of these datasets is driven by the unavailability of large public real-world datasets, which are often restricted due to privacy concerns or are not available with the necessary granularity for academic research. We have developed eight distinct datasets to explore different aspects of data noise and assess the effectiveness of our algorithm, specifically designed for scenarios involving both pristine and perturbed data conditions. The datasets are divided into two groups: one representing direct rides without transfers, and one representing rides with one transfer. For each group, we have 10 000 noiseless trajectories representing the ground truth with the associated train identifier(s), and then three versions with noise

Trip length	Number of trips
2	5710
3	1947
4	836
5	611
6	400
7	306
8	190

Table 2: Passenger trip lengths without train transfer

Trip length	Number of trips
4	1018
5	1753
6	1318
7	2551
8	3360

Table 3: Passenger trip lengths with train transfer

on stations, noise on timestamps, noise on both stations and timestamps. The datasets with the schedule data and passenger trajectory, as well as the source code to generate them, are available at https://github.com/Silvestri-Lab/TrainPassengerMapping.

5.2.1 *Perfect and Noise-Induced Datasets.*

Dataset of Passenger Trajectories with no noise: This dataset includes 10 000 passenger trips uniformly sampled from all trains in the schedule. Each trajectory precisely simulates one potential subtrip of a train, adhering strictly to the scheduled stations, arrival, and departure times. This dataset represents ideal conditions, where data perfectly aligns with train schedules. Each trajectory is obtained by selecting a train, and then by generating uniformly at random origin-destination pairs (with the origin appearing before the destination in the chosen train). The distribution of trip lengths generated for passengers is described in Table 2, where *Trip length* refers to the total number of stations in the trajectory.

We also create a dataset that includes passenger trips that involve train transfers during the journey. The distribution of trip lengths for passengers with transfers is described in Table 3, where *Trip length* refers to the total number of stations in the trajectory, including those after train transfers.

Dataset of Passenger Trajectories with Spatial Noise: To simulate scenarios where data might fail to capture every station stop due to signal loss or data omission, this dataset introduces trajectories with omitted stations. Specifically, building on the exact dataset, we remove a central station of a trajectory (i.e., any station that is neither the origin nor the destination) with a probability 20%. These trajectories derive from subtrips of trains, testing the algorithm's robustness under conditions of incomplete spatial information. By introducing this probability-based omission, the dataset realistically mimics the potential data gaps in real-world scenarios, thereby providing a more rigorous evaluation of the algorithm's ability to handle incomplete data. At the moment the dataset does not contain trajectories with added noisy stations: however we believe that this is not critical for our experiments since trajectories with removed stations might challenge the suffix tree in a similar way, by stopping the search among the suffix tree o force the search in a wrong subtree.

Dataset of Passenger Trajectories with Temporal Noise: This dataset tests the algorithm's ability to handle deviations from the scheduled times, reflecting the variability in real-world transit operations. To simulate realistic conditions, these inaccuracies follow a right-tailed normal distribution with a mean of 10 minutes and a standard deviation of 10 minutes. This approach accounts for both the unpredictability of train delays and discrepancies due to the methods used to register passenger timestamps. By incorporating these factors, the dataset provides a comprehensive evaluation environment.

Dataset of Passenger Trajectories with Combined Spatial and Temporal Noise: Combining challenges from both spatial and temporal inaccuracies, this dataset presents trajectories that include missing stations and altered timings, providing a comprehensive evaluation of the algorithm under challenging data conditions.

5.2.2 *Datasets Simulating Train transfers.*

Dataset of Passenger Trajectories with Train Transfer: Including 10 000 trips where passengers perfectly switch trains according to the schedule without any data errors, this dataset serves as a baseline for assessing the algorithm's train transfer detection capabilities. The sequences are obtained by randomly selecting among the larger Italian stations, and hence by randomly picking up an incoming train t_1 and an outgoing train t_2, with the constraint that t_1 arrives at the station before t_2 departs. Finally, we randomly select the starting point for train t_1 and the ending point for train t_2. The total trip lengths are reported in Table 3.

Dataset of Passenger Trajectories with Train Transfer with Spatial Noise: This dataset introduces missing stations in segments with train transfers, testing the algorithm's capability to detect and handle transitions even when parts of the journey data are missing.

Dataset of Passenger Trajectories with Train Transfer with Temporal Noise: By incorporating delays and early arrivals at points of train transfers, this dataset helps mimic the effects of time discrepancies on the detection and handling of train transitions.

Dataset of Passenger Trajectories with Train Transfer with Combined Noise: The most rigorous dataset combines spatial and temporal inaccuracies in journeys involving train transfers, testing the algorithm's robustness in managing multiple disruptions simultaneously.

6 EXPERIMENTS

In this section, we evaluate the effectiveness and performance of our proposed algorithm using various synthetic passenger datasets. The datasets, detailed in the Subsection 5.2, are designed to test the algorithm under two primary macro scenarios: train trips without transfers and train trips with transfers. Each macro scenario is further divided into four declinations: perfect data, spatial noise, temporal noise, and combined noise.

Dataset	Phase	Time
10 888 trains	Building Phase	359s
10 000 passenger trajectories	Matching Phase	112s

Table 4: Temporal Performance of the Algorithm

To assess the temporal performance of the algorithm on an M2 Max, we built a GST over a dataset comprising 10 888 trains and tested it using 10 000 synthetic passenger trajectories. The performance evaluation is divided into two phases: the building phase and the matching phase. These results are summarized in Table 4.

These performance metrics demonstrate the efficiency of the proposed algorithm in handling large datasets. The building phase, which involves constructing the GST from the train schedules, is computationally intensive but is a one-time process. The matching phase, which involves aligning passenger trajectories with the train schedules, is relatively quicker, showcasing the algorithm's capability to deal efficiently with query terms.

The two macro scenarios are train journeys with transfers and without transfers. In the former case, passenger trajectories entail completing the journey on a single train without any transfer. In contrast, in the latter case, passenger trajectories involve one or more train transfers during the route. Both scenarios share:

- **Perfect Data**: Trajectories that perfectly align with the train schedules.
- **Spatial Noise**: Trajectories where there is a 20% probability of omitting a station in the middle of the trip.
- **Temporal Noise**: Trajectories with timestamp discrepancies introduced to reflect real-world delays and variances.
- **Combined Noise**: Trajectories that include both spatial and temporal noise.

By applying our algorithm to these diverse datasets, we aim to comprehensively evaluate its robustness and accuracy in matching passenger trajectories to train schedules. The results from these experiments will demonstrate the algorithm's capability to handle incomplete and noisy data, providing insights into its practical application for real-world urban transportation systems.

Starting with the results of the scenario of direct trips without transfers, passenger trajectories were generated starting from the train scheduling dataset, the trajectories were fed into the algorithm, and the algorithm's output was cross-checked against ground truth. To evaluate the goodness of the algorithm, accuracy defined as follows is used as a metric:

$$\text{Accuracy} = \frac{\text{Number of correct matches}}{\text{Total number of trips}}$$

Results are shown in Table 5.

The table shows the algorithm's robustness and effectiveness under various conditions. The algorithm achieves an accuracy of 100% for both scenarios with and without train transfers when the passenger dataset perfectly aligns with the train schedules, indicating that the algorithm is fully capable of matching passenger trajectories to the correct train schedules under ideal conditions. When temporal noise is introduced, the accuracy drops slightly to 93% for trips without transfers and 91% for trips with transfers,

Dataset	Train transfer	Accuracy
Perfect Passenger	no	100%
Temporal Noise	no	93%
Spatial Noise	no	90%
Combined Noise	no	89%
Perfect Passenger	yes	100%
Temporal Noise	yes	91%
Spatial Noise	yes	89%
Combined Noise	yes	86%

Table 5: Accuracy by datasets

highlighting the algorithm's sensitivity to temporal discrepancies but also demonstrating its resilience, maintaining high accuracy despite the noise. The introduction of spatial noise results in an accuracy of 90% for trips without transfers and 89% for trips with transfers, posing a more significant challenge than temporal noise, but the algorithm still manages to correctly match a substantial majority of the passenger trips, reflecting its ability to handle incomplete spatial data to a reasonable extent. The combined noise scenario, which includes both temporal and spatial noise, presents the most challenging conditions for the algorithm, with an accuracy of 89% for trips without transfers and 86% for trips with transfers, underscoring the complexity of handling multiple types of noise simultaneously but also showcasing the algorithm's robustness as it continues to achieve relatively high accuracy.

Overall, the results indicate that the proposed algorithm performs well under ideal conditions and remains robust even when faced with various types of noise, with a slight reduction in accuracy with noise introduction being expected and acceptable, given the inherent challenges posed by temporal and spatial inconsistencies, and the ability to maintain high accuracy across different scenarios highlights the algorithm's potential utility in real-world applications where data may not always be perfect.

6.1 Temporal Noise Impact

To illustrate the impact of temporal noise on the matching phase, we present an example with tables detailing the train schedules and the passenger trip. Table 6a shows the wrongly associated train due to temporal noise, Table 6b shows the train from where the passenger is generated, and Table 6c details the passenger's trip.

The mismatch in the case of temporal noise can be observed by comparing the passenger trip with the schedules of the wrongly and correctly associated trains. The match is coherent with the list of stations of the passenger, and the passenger is associated with the nearest temporal train. Therefore, significant temporal noise can influence the matching phase, leading to incorrect train associations.

6.2 Spatial Noise Impact

To illustrate the impact of spatial noise on the matching phase, we present an example with tables detailing the train schedules and the passenger trip. Table 7a shows the wrongly associated train due to spatial noise, Table 7b shows the correctly associated train, and Table 7c details the passenger's trip. The mismatch in the case of

train_ID	arrival_time	departure_time	station
12377	13:40	13:40	SAVONA
12377	13:44	13:44	ALBISOLA

(a) Wrongly associated train

train_ID	arrival_time	departure_time	station
12379	13:38	13:38	SAVONA
12379	13:43	13:45	ALBISOLA

(b) Right train

arrival_time	departure_time	station_name	delay
13:59	13:59	SAVONA	21
14:05	14:05	ALBISOLA	21

(c) Passenger trip with the noisy timestamp and the random delay added to the correct timestamp.

Table 6: Example of matching error due to noise in time.

train_ID	arrival time	departure time	station
19070	08:45	08:45	CHIUSI
19070	09:02	09:03	TORRITA DI SIENA
19070	09:10	09:11	RIGOMAGNO

(a) Wrongly associated train

train_ID	arrival time	departure time	station
19076	11:00	11:00	CHIUSI
19076	11:15	11:20	MONTEPULCIANO
19076	11:25	11:26	TORRITA DI SIENA

(b) Right train

arrival time	departure time	station	removed
11:00	11:00	CHIUSI	No
11:15	11:20	MONTEPULCIANO	Yes
11:25	11:26	TORRITA DI SIENA	No

(c) Corrected passenger trip; "removed" indicates that the station has been removed when adding spatial noise.

Table 7: Example of matching error due to noise in stations.

spatial noise can be observed by comparing the passenger trip with the schedules of the wrongly and correctly associated trains. The mismatch occurs when a station in the middle is lost in an original trip of length 3, causing the algorithm to fail in reconstructing the proper trajectory. This leads the passenger to follow an incorrect path inside the GST, resulting in a train mismatch.

7 CONCLUSIONS

The study of mobility is particularly benefiting from the vast amount of data and information generated by various sensors, ranging from traffic loops to public transportation ticketing data, GPS data from apps, and various other sources. When appropriately processed, this data can be utilized to study mobility from different perspectives. In this paper, we focused on the opportunities to study public transportation, with particular attention to the railway sector but we believe the proposed approach can be easily generalized to the whole public transport sector. Our aim was to associate a trajectory containing a sequence of stations "visited" by a user with the corresponding sequence of trains actually used, a problem that we named Train-Passenger Matching problem.

We have provided the first formalization of the Train-Passenger Matching problem and the first approach to solve it, based on the Generalized Suffix Tree. The experimental evaluation provides high accuracy ($\geq$ 86%) with noise and perfect matches in the noiseless case. Of independent interest, we have provided a synthetic dataset of passenger trajectories based on the Italian train schedule.

An important open question is to provide a theoretical analysis of the problem and of the proposed solution. We conjecture the TPM to be NP-Complete as it is related to some string covering problems [8], where the goal is to cover a string x with substrings from a string collection C of the smallest size. One of the differences is that, in our case, the collection C of covering strings is given (i.e., schedule) and we aim at minimizing the number of segments (i.e., transfers) needed to cover the string x. It would also be interesting to develop algorithms that can provide rigorous guarantees on the approximation provided.

We believe the final output of TPM algorithm will enable data scientists, machine learning algorithms, and transportation models to utilize new, more comprehensive and enriched data. This will allow for the detection of the most frequently used stations for transfers, the overall train routes utilized by travelers, and the actual current waiting times between exchanged trains. This brand new information can be used to better plan train time tables and service lines as well as planning investments.

ACKNOWLEDGEMENTS

This work was supported in part by the *Big-Mobility* project by the Universisty of Padova under the Uni-Impresa call, by the MUR PRIN 20174LF3T8 *AHeAD* project and by MUR PNRR CN00000013 *National Center for HPC, Big Data and Quantum Computing.*

REFERENCES

[1] Gyözö Gidófalvi Adrian C. Prelipcean and Yusak O. Susilo. 2017. Transportation mode detection – an in-depth review of applicability and reliability. *Transport Reviews* 37, 4 (2017), 442–464. https://doi.org/10.1080/01441647.2016.1246489 arXiv:https://doi.org/10.1080/01441647.2016.1246489

[2] Bieganski, Riedl, Cartis, and Retzel. 1994. Generalized suffix trees for biological sequence data: applications and implementation. In *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, Vol. 5. 35–44. https://doi.org/10.1109/HICSS.1994.323593

[3] Ana Fernández Vilas, Rebeca P. Díaz Redondo, and Mohamed Ben Khalifa. 2019. Analysis of crowds' movement using Twitter. *Computational Intelligence* 35, 2 (2019), 448–472. https://doi.org/10.1111/coin.12205 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12205

[4] Dan Gusfield. 1997. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, USA.

[5] Christopher Horn and Roman Kern. 2015. Deriving Public Transportation Timetables with Large-Scale Cell Phone Data. *Procedia Computer Science* 52 (2015), 67–74. https://doi.org/10.1016/j.procs.2015.05.026 The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).

[6] Satoshi Hyuga, Masaki Ito, Masayuki Iwai, and Kaoru Sezaki. 2016. An online localization method for a subway train utilizing the barometer on a smartphone. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Burlingame, California) *(SIGSPACIAL '16)*. Association for Computing Machinery, New York, NY, USA, Article 50, 4 pages. https://doi.org/10.1145/2996913.2996999

[7] Guanyao Li, Chun-Jie Chen, Sheng-Yun Huang, Ai-Jou Chou, Xiaochuan Gou, Wen-Chih Peng, and Chih-Wei Yi. 2017. Public Transportation Mode Detection from Cellular Data. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 2499–2502. https://doi.org/10.1145/3132847.3133173

[8] Neerja Mhaskar and William F. Smyth. 2022. String Covering: A Survey. *Fundam. Informaticae* 190 (2022), 17–45. https://api.semanticscholar.org/CorpusID:253761364

[9] Philippe Nitsche, Peter Widhalm, Simon Breuss, Norbert Brändle, and Peter Maurer. 2014. Supporting large-scale travel surveys with smartphones – A practical approach. *Transportation Research Part C: Emerging Technologies* 43 (2014), 212–221. https://doi.org/10.1016/j.trc.2013.11.005 Special Issue with Selected Papers from Transport Research Arena.

[10] Italian State Railways. 2024. *Methodological aspects for passenger mobility analysis using big data.* Technical Report. https://www.fsitaliane.it/content/fsitaliane/en/fs-research-centre/studies-and-research.html

[11] Scalzo and Mangione. [n. d.]. Treni e ferrovie italiane: orario treni, servizi di stazione e altri dati di interesse ferroviario. https://www.e656.net/. Accessed: 2024-05-30.

[12] Ainhoa Serna, Jon Kepa Gerrikagoitia, Unai Bernabé, and Tomás Ruiz. 2017. Sustainability analysis on Urban Mobility based on Social Media content. *Transportation Research Procedia* 24 (2017), 1–8. https://doi.org/10.1016/j.trpro.2017.05.059 3rd Conference on Sustainable Urban Mobility, 3rd CSUM 2016, 26 – 27 May 2016, Volos, Greece.

[13] TomTom. 2024. London Traffic Report. https://www.tomtom.com/traffic-index/milan-traffic/ Accessed: 2024-06-07.

[14] Anette Østbø Sørensen, Johannes Bjelland, Heidi Bull-Berg, Andreas Dypvik Landmark, Muhammad Mohsin Akhtar, and Nils O.E. Olsson. 2018. Use of mobile phone data for analysis of number of train travellers. *Journal of Rail Transport Planning Management* 8, 2 (2018), 123–144. https://doi.org/10.1016/j.jrtpm.2018.06.002