SMAFramework: Urban data integration framework for mobility analysis in smart cities

Diego O. Rodrigues[1,2], Azzedine Boukerche[2], Thiago H. Silva[3],
Antonio A. F. Loureiro[4] and Leandro A. Villas[1]
[1]Institute of Computing, University of Campinas - Brazil
diego@lrc.ic.unicamp.br,leandro@ic.unicamp.br
[2]School of Electrical Engineering and Computer Science, University of Ottawa - Canada
boukerch@site.uottawa.ca
[3]Dept of Informatics, Federal University of Technology, Parana - Brazil
thiagoh@utfpr.edu.br
[4]Dept of Computer Science, Federal University of Minas Gerais - Brazil
loureiro@dcc.ufmg.br

ABSTRACT

Smart cities emerge in computer science as a topic to cover how the technology of information and communication can be used in the urban centers to monitor its dynamics and allow the improvement of services for the citizens. In these urban centers, different methodologies are used in order to collect data and provide them to applications. These data come from several heterogeneous sources, thus there is an effort to integrate and standardize them before their use. Also, a significant amount of this data has spatio-temporal annotations, which may be used to analyze the city dynamics, such as the mobility flow. Due to these characteristics of the data generated in urban centers, and also the possibilities brought by their use and analyses, this work presents a novel approach to collect, integrate and perform some analysis tasks in mobility data from smart cities. Thus, the SMAFramework can analyze mobility patterns based on a Multi-Aspect Graph (MAG) data structure. To show the potential of the framework, it is proposed a method to analyze the saptio-temporal correlation between data from two different data sources in the same city. Real data collected from social media and a taxi system of the city of New York are used to evaluate this method. The obtained results allowed to understand some of the applicabilities of the framework and also provided some insights on how to use the framework to resolve specific problems when analyzing mobility in urban environments.

KEYWORDS

Smart Cities, Mobility Analysis, Big Data, Urban Data

ACM Reference format:
Diego O. Rodrigues[1,2], Azzedine Boukerche[2], Thiago H. Silva[3],
Antonio A. F. Loureiro[4] and Leandro A. Villas[1]. 2016. SMAFramework: Urban data integration framework for mobility analysis in smart cities. In *Proceedings of The 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Miami Beach, USA, November 2017 (ACM MSWiM'17),* 10 pages.

ACM MSWiM'17, Miami Beach, USA
 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recently, the complexity of cities has increased considerably due to several factors such as the bigger number of inhabitants living in urban environments [12]. With that, urban computing has emerged as a topic in computer science envisioning the application of information and communication technology to improve services provided to city dwellers [12, 16, 26]. Through the improvement of city services, the new cities, also called smart cities, have the opportunity to offer a better quality of life to their inhabitants in different aspects, ranging from more efficient city mobility and less bureaucratic and transparent governmental processes.

In order to achieve that, it is fundamental to obtain data to better understand the city functioning. Data generated by cities can be originated from different sources. For example, one typical approach is to use the traditional sensor networks that can be seen as a sensing layer of a particular phenomenon. There are several types of sensors available nowadays, and the combination of different sensor networks may allow the understanding of complex phenomena. However, building sensor networks may not be scalable in certain circumstances, for instance, to cover a metropolitan area [19]. Thus, a useful alternative is to consider user participation in the sensing process, where users with their mobile devices act as sensors in the city. This approach has become practical due to the decrease of the prices of mobile devices (e.g., smartphones) and the popularization of location-based social networks, such as Twitter, Instagram, and Foursquare [6, 19].

Several approaches have been used to analyze data collected from single urban data sources and led to remarkable results. However, methodologies to explore data from multiple sources still need to be further investigated [14, 20]. Particularly, to combine urban data to obtain more precise insights about the city impose several issues, among them data integration [14, 20, 27]. This issue is especially critical for spatio-temporal data, data of a phenomenon containing geographic coordinates and annotations about the time when it happened.

Knowing that representing and integrating different sources of spatio-temporal data is a challenge [7, 25], the goal of this study is to propose a novel framework, called SMAFramework, to help

city planners and others on these issues. The framework eases the process of dealing with multiple heterogeneous sources providing an approach to standardize these data in order to facilitate information extraction regarding urban mobility. This framework envisions to provide different tools to help professionals as city planners, researchers, and engineers to extract insights about urban mobility dynamics to better plan the city to its citizens. In this paper, we propose one of these tools and, to show the potential of the framework, we design an algorithm based on fuzzy-logic scores to evaluate spatio-temporal correlations between entries from different sensing layers of an area. The approach, entitled Fuzzy Matcher, aims to provide a tool for evaluating urban areas in space and time – rather than performing an aggregated analysis not aware of the time variance – where data collected from different sources are highly correlated; or evaluate areas where this correlation does not exist but could be stimulated. We use real data from New York City using two distinct data sources: (i) spatio-temporal annotated tweets (i.e., short messages shared in the Twitter platform); and (ii) trips performed in the New York City using yellow taxis. We demonstrate how to explore the proposed framework using our datasets to establish a better positioning of the taxis, which could, potentially, improve taxi services to users and increase revenue for taxi drivers.

This work is organized as follows. Section 2 presents the related work. Section 3 describes how data is collected and represented. Section 4 presents and explains the framework architecture. Sections 5 and 6 present the algorithm developed to demonstrate the applicability of the proposed framework. Section 7 presents the results of mobility analysis exploring our framework in our studied real-world urban datasets, showing the potential for new services and applications. Finally, Section 8 summarizes the contributions obtained with the proposed methods and presents some future directions.

2 RELATED WORK

The use of graphs to represent mobility data is recurring in the literature. This recurrence allied to the need of representing the time variance of this data has resulted in different graph-based models with time representation support. The most common examples of these models are: (i) Snapshot based [5, 21], where the graph is formulated as N disconnected graphs where each graph represents a moment in time, with all its connections and nodes; (ii) Continuous Time Intervals [2], where the links between the nodes are described as functions in a certain fashion that allows to identify whether the edge exists or not given a specific moment in time; (iii) Spatio-Temporal Edges [10, 13], which use two kinds of links to represent the interactions between the nodes (spatial edges and temporal edges); (iv) Time Mixed Edges [8], similar to the Spatio-Temporal Edges, however, in this model the existence of mixed edges (i.e., edges that connect nodes in different moments of time and locations in space) are allowed; and finally (v) the Multi Aspect Graphs [9, 23]. One of the advantages of selecting the MAG to represent the smart city data is its capability of representing all kinds of connections present in the other graphs. Table 1, extended from the paper of Wehmuth, et al. [23], shows a representation map indicating for each model the other models that can be represented with its definition. Also, the last column was added to emphasize that only the MAG has native support for multilayer.

Table 1: Representation map for the different models used to extend the graphs with time variance data

	Snapshots	CTI	STE	TME	MAG	Multilayer Support
Snapshots	X	X				
CTI	X	X	X			
STE	X	X	X			
TME		X		X		
MAG	X	X	X	X	X	X

Furthermore, other frameworks for mobility analysis, based on different models, have emerged in the literature. These frameworks were developed focusing not only on mobility in the cities, but also other scenarios as discussed below. Thakur & Helmy [22] investigate how current mobility models are adequate to represent the human mobility. Also, the authors propose a framework, namely COBRA, which extends the mobility model metrics to match with real traces. The framework aims to create a model for mobility to allow simulation of network protocols dependent on that mobility. To evaluate their framework, the authors use some network protocols, combined with other mobility models and also traces from real world in specific environments within the city (e.g., university campus, parks). This framework uses archived data collected from the scenarios when using real world data, this data also can be combined from different sources or scenarios in order to create more complex models. Their framework was not built with the purpose of analyzing mobility, but, instead, model it, thus it lacks some of the needed features to perform analysis tasks, such as the layer-based structure (i.e., all data collected are mixed together), and the possibility to use real time data.

Patroumpas [15] propose a methodology to model traces of generic positional information. That model introduces a sliding window operator that allows an incremental examination of streamed motion traces. On top of this model, there is a query language able to express spatio-temporal queries to retrieve data stored in the sliding window format. That work presents the approach as a tool to analyze online streamed data, but does not show how it can be used to stored data, which is an important source of mobility data. For example, some of the mobility data cannot be published in real time due to privacy constraints and/or other issues (e.g., the New York Yellow Taxis Trip dataset used in this paper), thus the unique way to work with this data is by collecting it from archived sources. These two ways of accessing data, streamed and archived approaches, may be used to ensure the exploration of the data available, and, thus, frameworks that can handle both formats have extra advantages, such as the ones following.

Silva et al. [20] discuss the concept of sensing layer division for different types of data available in geographical regions. The authors focus on analyzing data about urban regions provided by users using their portable devices through location-based social networks platforms (e.g., Twitter). They provide a generic way of structuring data for analyses, however, the abstract model is not specialized, i.e., does not contain functions to help for instance in mobility analysis. Besides, the time dimension in that model presents some limitations. In our work, the focus is on creating a tool to analyze mobility

patterns, thus the structure of the data is well defined and all the sensing layers can be standardized in a common data structure (i.e., the MAG structure).

In order to perform maritime traffic analysis, Salmon & Ray [17] introduce a spatio-temporal framework capable of analyzing archived and streamed data sources. The proposed architecture is oriented to receive data in real time as data streams. The authors explore also a query specification standard, which allows the creation of persistent and one-time queries, reflecting respectively queries that will stay listening to new incoming data, or queries that will run over the data already stored. That study is focused on maritime traffic, where the data comes from sensors, but it is not shown how data from different sources can be used in that model.

The SMAFramework, proposed in this paper, is focused on analyzing data generated in cities observing their characteristics, such as the high heterogeneity of the data sources, which may be archived or streamed in several different standards. The architecture is extensible, allowing to collect data from data sources not currently present in the framework in a plug-and-play fashion. Its focus is on mobility, allowing defining more specifically details for this purpose. Table 2 summarizes the contents of this section and indicates the differences from the models and frameworks discussed and our proposed model. Also in Table 2, it is possible to identify the number of different sensing layers supported by the model.

Table 2: Summary of related work emphasizing the differences from other studies and the SMAFramework

	Archived Data	Streammed Data	Unified Data Structure	Combination of Data Sources	Number of Independent Layers
Thakur & Helmy (2013)	X			X	1
Patroumpas et. al. (2013)		X	X	X	1
Silva et. al. (2014)	X	X		X	N
Salmon & Ray (2016)	X	X	X		1
SMAFramework (2017)	X	X	X	X	N

3 URBAN SENSING LAYERS AND ITS REPRESENTATION

One way to classify data generated in a city is to use sensing layers. Silva et al. [20] define sensing layers as datasets describing specific aspects of a given geographical location. The raw data in these layers must be collected and processed in order to be used for analysis purposes. An example of sensing layer is data collected from the Twitter platform, which provides *tweets* that may be spatio-temporal annotated. These annotations can be used to identify the position of the person by the time of sharing a message. This is done using the latitude, longitude and time fields of the tweet. Also, more sophisticated analysis can extract other information from the content field of the message.

Each sample of data in the sensing layer includes a time interval when the data was generated, a location where it was produced, a specialty data and one or more IDs (or UIDs – user ids) to represent the entity who created the data. The specialty data could be the message in the tweet, or the photo itself shared in a photo sharing service. In this work, we consider the data location itself as the specialty data. This division of sensing layers brings a series of advantages in different applications of urban computing because it helps to extract useful information [20]. Currently, many approaches explore data from single sensing layers and get useful results. However, data combination of different sources, as complementary data, to improve results is an emerging research topic when studying data in urban areas. The unification of these datasets is not a trivial task, thus resulting in different efforts proposing models and frameworks capable of representing and aiding the use of these datasets.

Zheng [27] identifies some features that may facilitate the success of a framework to represent and analyze urban data. That model must be able to (i) integrate data from heterogeneous sources: organizing the data in an efficient way for retrieval and mining, while keeping the consistency of the data for each independent source; (ii) allow cross-domain data combination to allow knowledge extraction that cannot be extracted from single data sources; and (iii) be able to manipulate sequences of time-stamped locations coordinates.

Inspired by the concept of sensing layers and by features needed to facilitate the success of a model to represent data from urban centers, we propose the use of a Multi Aspect Graph (MAG) [9, 23] to represent the data inside the framework. The MAG is an extension of the Graph model that allows the representation of different data features by using the so-called aspects. In the literature, there are different descriptions of the model, but the models proposed by Kivela et al. [9] and Wehmuth et al. [23] are the most well accepted by the academic community. Both models are equivalent with a few peculiarities distinguishing them. In the proposed framework, the model presented by Kivela et al. [9] was used because of its simpler and clearer mathematical definition, what reflects in being better referenced in the literature. Figure 1 shows a representation of a MAG indicating how some of the real-world characteristics are mapped to the structure in the framework. Specifically, as for example, Twitter and Taxi Trips are shown as data sources mapped as layers in the MAG.

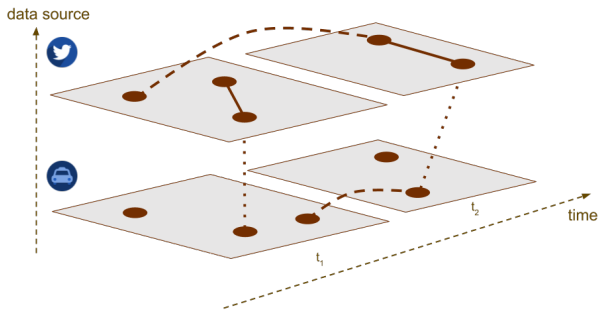

Figure 1: Sample representation of a MAG structure

In Figure 1, it is shown a MAG with two aspects. An aspect is one axis where layers can be created, each of these layers is represented as a square in Figure 1. The first aspect, represented vertically, is the data source aspect, thus every surface (i.e., elementary layer, or row of layers) contains data from a different source. It is important to mention that even the same source can produce information for

different layers when the source produces entries that represent different specialties (e.g., an entry from a taxi trip can represent either the presence of a person or the presence of a vehicle in a given location). The second aspect, represented horizontally, is the time. In the model, the time is discretized, so every elementary layer in the temporal axis represents a moment in time when the data was generated. Every node in the graph is a piece of collected data and the links connecting them represent their relationships. The model allows the representation of any kind of relationship, which is able to adapt to different analysis cases, what is accomplished by using labels to identify the links.

In the example, two kinds of relationships are shown. The thick dashed line represents a movement of the data producer, thus two nodes connected by a thick dashed line have the same UID. The thin dashed lines represent nodes in different layers, which were created by the same entity. Since the data was collected in different sources, their UID may not be the same, thus this kind of connection must be created through analysis tasks. These links can be created when the UIDs in different layers are the same, or when the map of UIDs in different layers is known, however, it is not clear yet how these connections can be created when this information is not present (i.e., it is an open question, especially when studying user-based sensors [20]). Yet, our work shows initial insights on possibilities of creating these connections. We call this mapping of sensors/users in different layers as the Cross-layer sensor/user mapping problem.

According to Kivela et al. [9], the MAG is a model that uses a composition of aspects and layers to add information to graphs. The aspects are the dimensions (i.e., axis) in which the layers are in. Every aspect a has a family of elementary layers L_a, which represent a unique value of one aspect of the data (e.g., one moment in time in the temporal axis permeating all the data sources). This way, if we have d aspects, we can define all the elementary layers as $L = \{L_a\}_{(a=1)}^{d}$. Thus, the set of all layers in the graph is the Cartesian product of all elementary layers $L_1 \times L_2 \times \cdots \times L_d$. Since any of the nodes in V can be in any layer, the set of nodes that are present in the graph is $V_m \subseteq V \times L_1 \times L_2 \times \cdots \times L_d$. Any of the edges in the graph can connect two different nodes in any layer or aspect. Thus, the set of edges is $E_m \subseteq V_m \times V_m$. Using this information, we can define a MAG as $M = (V_m, E_m, V, L)$.

One advantage of using the MAG model is because it keeps some of the characteristics of the traditional static graphs. The traditional graphs were used to represent static information, and also sometimes to do analytical tasks over aggregated area over time (i.e., not time-aware analysis), but some peculiarities of the mobility data, for example, cannot be expressed in a timeless fashion. Since these graphs have been used to represent mobility data before the advance of time-varying graphs, some of the techniques and methodologies already available may be reused. This model is also adequate to represent the division of the sensing layers to facilitate the development of the proposed framework and enables the execution of analysis tasks over the data.

4 SMAFRAMEWORK: SMART CITIES MOBILITY ANALYSIS FRAMEWORK

In this section, we introduce the framework to aid developers, researchers and city planners to analyze mobility data generated in the city environment. In order to accomplish that, the framework helps gathering data from the different sources available in the city and standardize them to facilitate data management and analysis. Furthermore, the framework also provides a base structure to perform trivial and common tasks to deal with any data, including mobility (e.g., clean invalid data, remove duplicates and filter data). Nowadays, every data scientist working with mobility data tends to build his/her own solution, which raises a challenge on data standardization among others, preventing its reuse in an easy way. Providing a tool that allows users to avoid rebuilding these procedures and also perform them in an optimized way can, potentially, help to save time to work on more specific details of their analysis. Finally, the framework also envisions to provide ways to deal with some challenges regarding mobility data analysis tasks. More specifically, matching traces in different sensing layers and analyzing the correlation of layers from different perspectives.

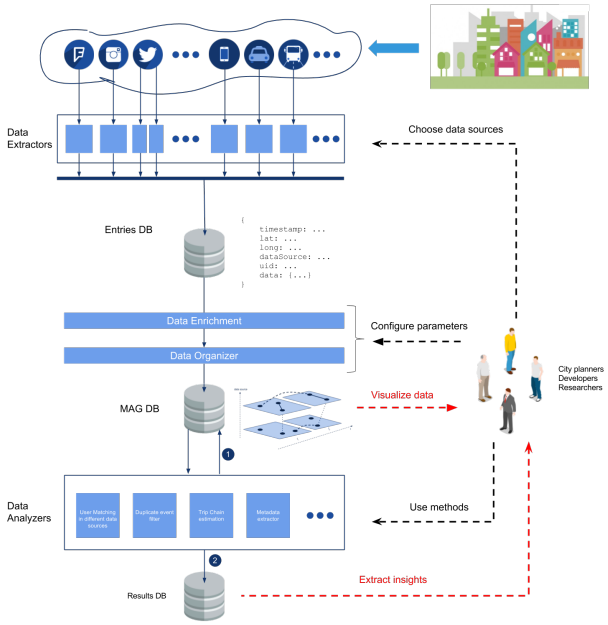

Figure 2: The SMAFramework architecture

Figure 2 shows the framework architecture. At the top, it is shown the city generating data through different sources. Every icon inside the cloud represents a different source of raw data that is being collected to be analyzed, using Data Extractors. These Data Extractors are pieces of software that collect the raw data from its source and adequate it in a basic initial format called Entries. For example, a Data Extractor can use an API to collect data from a social media platform and put it in the Entry format. This data may also be provided in different ways rather than through APIs. The objective of the Data Extractors consists in looking at these peculiarities of each data source and performing the initial data extraction step. In the example shown in Figure 2, the Twitter source is been explored by two Data Extractors that can, for instance, be a stream extractor, collecting real-time data, and another archived extractor, collecting stored data in other formats.

Every entry represents sample data, space and time annotated that will be stored in the framework. These samples consist of geolocation (i.e., latitude and longitude), time, data source and user ids, i.e., represent the entity, which generated the sample. In the absence of information indicating the UID, the framework will generate values assuming that every entry was created by different entities. Also, these samples may contain additional data that might help eventual analysis cases. These entries are initially stored in the database, as shown in Figure 2. Once in database, there are common tasks that read the entries and organize them into the MAG structure and perform data enrichment both according to the users' configurations. These tasks can add more data samples that were not initially in the entries, but can be inferred. For example, Mahrsi et al. [11] present two ways of enriching data from bus system traces. In those scenarios, it is known when the users take a bus, but it is not known when they leave it. Thus, in order to infer this information, the authors raise two assumptions: (i) closest-stop – for a given transaction, the passenger arrives in the closest station of the one where the next transaction starts; and (ii) daily-symmetry – for the last transaction of the day, the passengers arrive in the closest station of the one they used to do the first departure of the day.

After the Data Organizer and Data Enrichment steps, the data can be structured on the MAG model. This structure will be stored by the framework in a so-called MAG database. At this point, the data can be checked by the users of the framework, for example, using a visualization tool. On top of the MAG structure, the Data Analyzers run tasks that help the extraction of knowledge from the mobility collected data. Similar to the Data Extractors, the data analyzers can be conveniently added to extend the framework and adequate it to the user needs. The framework is based on a two-step organization of the data, i.e., first entries, then the MAG, to reduce the amount of work needed to add a new module. Thus, if users need to use the Data Analyzers that already exist in a new data source, they can only deploy the Data Extractor to be able to do it. A similar situation occurs in the case where the user is going to use the already supported data sources, but with a new Data Analyzer. This division also organizes the framework in a layer-based design, making easier to add new layers between the existing ones to improve metrics as performance and/or scalability, such as adding a caching system to enhance the data access.

The MAG structure and the Data Analyzers are the core of the framework. These modules allow the framework users to explore the data and extract knowledge. The Data Analyzers can perform two types of tasks: they can change the MAG structure to allow the visualization of certain patterns, or also can output summarized results to a result database. Since these analyzers are a key part of the framework, the present paper shows a use case with a proposed Data Analyzer called Fuzzy Matcher. This analyzer aims to create spatio-temporal matches between two different sensing layers, and its full description appears in the next section.

5 FUZZY MATCHER

The analysis provided by the SMAFramework can provide important insights for different city stakeholders. For instance, citizens helping other people to better use the resources available in the city; or city managers, aiding themselves to build the policies to manage the city. To show a use case exploring the framework, we propose a Data Analyzer, called Fuzzy Matcher, to investigate spatio-temporal correlation of data from different sensing layers.

Fuzzy Matcher is an algorithm that identifies spatio-temporal matches between nodes of the MAG from different sensing layers of the smart city. After the identification of the match, the algorithm also evaluates a temporal and spatial score of the matches. This score is evaluated based on the spatio-temporal distance between the matching nodes and a dispersion function. While using this procedure from the framework, the users are able to specify parameters such as distance, time precision, and also distance and time depreciation function. The change of this function can be used to adapt the analysis for specific cases. For example, the way crowds move in a city may vary according to a particular scenario. A crowd as a parade, for instance, tends to walk longer distances to make its cause more visible, whereas a crowd attending a concert does not move a lot. The way that crowds, and other mobility flows, behave in the city can be better described by different dispersion functions.

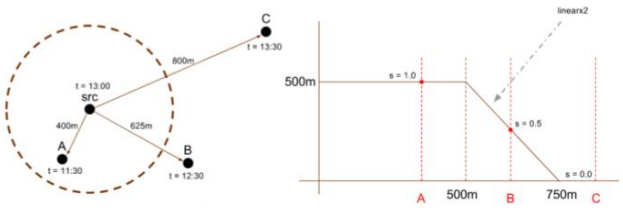

Figure 3: Fuzzy matching process

Figure 3 shows how the Fuzzy Matcher algorithm works and also how its parameters are used. The distance precision D, time precision T, and the temporal and spatial depreciation functions $td(t)$ and $sd(d)$, respectively, where d and t are the spatial and temporal distances, are used to create a curve to evaluate the score of the match. The spatial score is given by $SS(d) = f(d)/D$ where $f(d) = \{D$, if $d < D$; $sd(d - D)$, otherwise$\}$, and similarly for the temporal score: $TS(t) = f(t)/T$ where $f(t) = \{T$, if $t < T$; $st(t - T)$, otherwise$\}$. For example, given the scenario in Figure 3, the match (SRC, B) has spatial score $SS(625) = 0.5$, for $D = 500$, and the spatial depreciation is a linear function (i.e., $sd(d) = D - d$). In the same scenario, since the pair (SRC, C) has $SS(800) = 0$, this pair does not become a match. The same is valid in case of $TS(t) = 0$.

Considering the proposition of this method, this work aims to add a few possibilities to aid researchers and developers in analyzing mobility data when compared to other methodologies. For instance, geographical databases offer a variety of tools to match spatial data, such as geo-located queries, that allow searching a specific area; or even observer functions that make possible to watch for changes in the database within a region. However, many of these approaches lack temporal awareness, not performing any analysis over time. Also, despite only matching samples, the Fuzzy Matcher provides scores to represent how strong/weak is a given match. By using these scores, thresholds can be identified to allow classification of the matches, for instance. Finally, Fuzzy Matcher introduces a way of using different depreciation functions to analyze datasets that might have different dispersion behaviors.

One of the issues related to finding matches in datasets is the algorithmic complexity. If the algorithm is built in a naive manner, its complexity can become $O(n^2)$. This complexity might be a barrier to use the Fuzzy Matcher algorithm when it comes to huge datasets with dozens of millions of samples. To reduce the algorithmic complexity, we propose a method of walk through the dataset comparing samples to their neighbors. This method was built inspired in the Bucket Sort [3] algorithm, leading to the so-called Bucket Walk. The method is described as a three-dimensional walk in the section 6, but it can be extended to any number of dimensions.

6 BUCKET WALK

In this paper, to reduce the $O(n^2)$ complexity of the Fuzzy Matcher algorithm, it is introduced the Bucket Walk approach to visit samples in the datasets and compare them with their neighborhood. In this section, the Bucket Walk is described as a three-dimensional method to walk through a dataset, however, the method can be extended to work with any number of dimensions. Also, this method is not bound to work only in the Fuzzy Matcher algorithm, rather it can be used with any algorithm that aims to compare dataset samples against their neighbors. The use of this walking method reduces significantly the time and memory usage to run the Fuzzy Matcher. This section shows some results collected from experiments using the Bucket Walk and the naive approach. The same experiments could not be obtained from the complete datasets since the naive experiments were taking much time to run and exhausting the memory even if some of the best servers are available to be used (i.e., 24 cores, 40Gb RAM). By using the Bucket Walk to execute Fuzzy Matcher, the server was able to perform the task saving significant resources (i.e., using 16 cores and 8Gb RAM) and time.

As mentioned, the Bucket Walk was built using as inspiration the Bucket Sort algorithm [3]. The Bucket Sort consists in splitting the samples to be sorted into smaller groups called buckets (or bins), sort the samples within the bins and then sort the bins themselves. In order to split the original dataset, the Bucket Sort creates hashes of the samples. For example, a hash function, when sorting strings, could be to get the very first character; each character would be mapped to one bin, so every string starting with the same character would be placed in the same bin. We argue that this approach can be extended to organize multi-dimensional data by creating buckets and hashes at each data dimension in the samples. For instance, the data analyzed in the Fuzzy Matcher is spatio-temporal aware, thus resulting in a 3D data model, with the latitude, longitude and time dimension – two spatial dimensions and one temporal.

The Bucket Walk consists in creating hashes to split the original dataset in a 3D space (i.e., latitude, longitude and time). Given this division, while walking through the dataset, it is possible to compare samples in neighbor buckets, rather than the whole dataset. We need to choose a hash function to use this approach, and its choice will affect the performance of the walk, and consequently, the algorithm that uses it (e.g., Fuzzy Matcher). One possibility of choosing the hash function is to define a base sample, which may or may not be in the dataset, and compute the distance of every sample to this base sample, and then get the integer part of the division of this distance by a factor K. This has to be done for the three dimensions of the data separately. If the base sample has the lowest (or highest) values for every dimension the result of the hash and split tasks will be a cube of data buckets. Every bucket will contain the samples that obtained the same result from the hash function. Once identified the base sample, the hash procedure has to be performed for every data sample, which means it is $O(n) \subset O(n^2)$ operation, then we do not have the complexity problems to run the hash over all data. Furthermore, if the base sample is not known initially, all data samples will have to be visited before performing the walk to identify it, resulting in a $O(n) \subset O(n^2)$ procedure, which also does not imply in any complexity increase compared to the naive approach.

The factor K, mentioned in the hash function, has also to be defined. To do so, it is necessary to look at how the neighborhood of a data sample is defined. In the Fuzzy Matcher algorithm, the neighborhood is defined by a custom function $f(x) = \{SS(x), TS(x)\}$, where the factor can be defined as $K = x$ when $f(x) = 0$. Conveniently, for the sample presented in Section 5, with the linear depreciation function and a precision $P = \{D, T\}$, $K = 2P$. Defining K this way allows the algorithm, for a given sample, to only compare data in adjacent buckets in the cube. It is important to note that two K values were obtained, one for spatial hash and another one for the temporal hash. This approach allows to have one different K value for each dimension.

Once obtained the hash function, the K factor splits the dataset samples into the cube of data buckets. The worst-case scenario would still result in a $O(n^2)$ complexity. This is the scenario where all data samples would be allocated to the same bucket. However, it is known that the usual distribution of the studied data has no such trend. To show this distribution, the proposed hash function was used to split the data samples in the experiments. These data samples were collected from the Twitter platform and the Yellow Taxis from the New York City, as previously mentioned, and contain respectively 399, 024 and 10, 580, 378 samples. More details about the datasets are provided in Section 7.

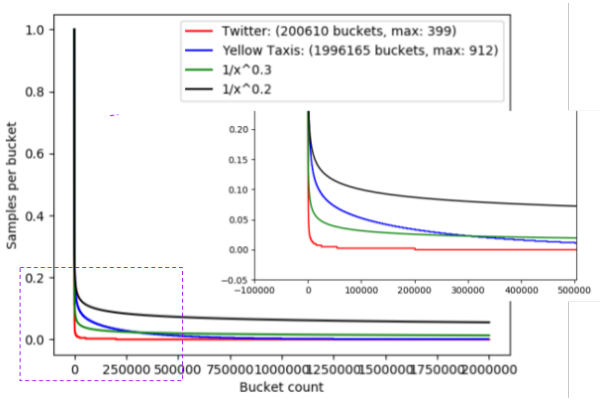

Figure 4: Distribution of the buckets usage

Figure 4 shows the distribution curve of data samples in buckets. In that figure, the horizontal axis represents the buckets while the vertical one represents the number of samples in the buckets. The buckets were sorted according to the number of samples in it. Also, to facilitate the visualization, a number of samples per bucket were normalized to the $[0, 1]$ interval. The legends of Figure 4 show a number of buckets with at least one sample and the number of samples

in the most popular bucket. It is visible that the datasets have a quite sparse distribution where most of the samples are centralized in a few buckets, but not all in the same bucket as described in the worst-case scenario. Indeed, the Twitter and Yellow Taxi curves can be approximated by an equation of the form: $D(x) = \frac{\alpha}{x^{\beta}+1}$, as shown in Figure 4, where the factors α and β will define the shape of the curve, also the +1 was ripped without loss of generality. By using this approximation we can estimate $n \simeq \sum_x \frac{\alpha}{x^{\beta}}$, resulting in:

$$n^2 \simeq \left(\sum_x \frac{\alpha}{x^{\beta}}\right)^2 = \sum_x \left(\frac{\alpha^2}{x^{2\beta}}\right) + 2\sum_{i>x}\left(\frac{\alpha^2}{(xi)^{\beta}}\right). \quad (1)$$

This way, the equivalent complexity will be:

$$O(n^2) = O\left[\sum_x \left(\frac{\alpha^2}{x^{2\beta}}\right) + 2\sum_{i>x}\left(\frac{\alpha^2}{(xi)^{\beta}}\right)\right] = O\left[\sum_x \left(\frac{\alpha^2}{x^{2\beta}}\right)\right]. \quad (2)$$

The last step in Equation 2 is only possible because of the big O notation, where only the higher order factors in a sum affect the algorithmic complexity. In Equation 2, there is still a quadratic term, as predicted in the worst-case scenario, however, this quadratic term is attenuated according to the samples distribution. For the specific datasets studied in this work, some upper bounds of the values of β were identified as $\beta \leq 0.3$ for Twitter, and $\beta \leq 0.2$ for the Yellow Taxis. These bounds were simply defined by observing the behaviors of the $\frac{1}{x^{0.3}}$ and $\frac{1}{x^{0.2}}$ curves. The obtained equation represents the expected result, where the full complexity of the algorithm is evaluated according to the sum of the complexity contained at each bucket. In this case, the value of α is the maximum number of samples in a single bucket. In the case that $\alpha = n$, i.e., the worst scenario, only one bucket exists, thus $x \in \{1\}$ and the complexity becomes again, as predicted, quadratic, as follows:

$$O\left[\sum_{x\in\{1\}}\left(\frac{n^2}{x^{2\beta}}\right)\right] = O\left(\frac{n^2}{1^{2\beta}}\right) = O(n^2). \quad (3)$$

Another advantage of using the Bucket Walk is the fact that it does not require all the data to be loaded into the memory of a server. Only chunks of data containing the buckets of a region to be analyzed need to be loaded, making it more convenient to run in low memory environments. Also, when running the algorithm over the whole dataset, the nodes within the same bucket can be analyzed in sequence, which facilitates the usage of a caching hierarchy. This works resembles the Iteration Space Tiling optimization [24].

There are in the literature some other access methods that can be used to retrieve spatial and temporal data. For example, the R*-tree model [1] is based on tree structures commonly used to query spatial data samples. That method builds a tree with two types of nodes: leaves and pages. Starting from the top page, which covers all the samples, the search algorithm starts looking for a child page that covers the desired query. Once the algorithm identifies this page, the process will repeat recursively until the algorithm reaches a page with the leaf children. At this point, the algorithm will look at the leaf nodes to find the queried sample. In this model, the hierarchy is used to organize the nodes, however, even for sibling pages, there are no indications if they are from neighbor regions. The absence of a notion of neighborhood in the R-tree may result in repeated visits to the higher levels of the tree while performing the nearest neighbor queries [18]. In the Bucket Walk, this information of neighborhood is used to reduce the number of comparisons needed to execute the Fuzzy Matcher. There are still different extensions of the R-tree algorithm in the literature, which may improve its performance in the nearest neighbor queries, however, they are not using all the information of the search domain of the Fuzzy Matcher. For instance, a depreciation function to estimate the maximum search area is not used.

StreamCube [4] is another algorithm to search data samples that is aware of the spatial and temporal variations. It consists in organizing the data in a 3D cube, making it easier to access data querying over latitude, longitude and time. The difference of this approach to ours is that it queries items as continuous values. To quickly navigate over the data, StreamCube creates some hierarchical divisions based on temporal and spatial regions. Furthermore, data items can be queried over week periods, and these periods can be accessed on a daily basis. In our solution, we we a hash strategy, leading to indexes that may result in a more efficient data access when compared with a continuous representation. In fact, indexes allow Fuzzy Matcher to have a better data access, retrieving less unneeded data.

7 EVALUATION

We use Fuzzy Matcher to analyze data collected from two different data sources in the New York City. In order to analyze the correlation of both datasets, we defined an area of interest that covers the Manhattan region and also some near neighborhoods. The first dataset consists of tweets collected during the month of January/2016. During that month, we extracted 399,024 tweets geo-annotated (i.e., tweets that contain information about the geo-location where it was shared). The second dataset is available on the NYC Open Data portal. It has information about the New York Yellow Taxi trips. In January/2016, there were 10,580,378 valid trips within the boundaries of our analysis.

The first experiment consists in using the Fuzzy Matcher scores to check the spatio-temporal correlation of zones of the city. In this experiment, we analyze if we can use Twitter data to better position taxis within the city so these taxis will be more accessible for the citizens, and also more trips will be requested. The Yellow Taxis in New York only provide their services through street-hails, thus they must be well positioned to have a better coverage of the city area, which consequently results in a more intelligent transportation system. Our hypothesis is that a region with a relevant number of tweets may indicate a significant amount of possible taxi users. In this way, Twitter data, collected in real time, may be used to indicate regions where taxis could be positioned to serve citizens, or to quickly identify changes in the city environment caused by events, such as demonstrations and traffic jams.

We used the Fuzzy Matcher algorithm to analyze the correlation of samples collected from Twitter and the Yellow Taxis sensing layers. The algorithm identifies spatio-temporal matches with a distance precision of 100 meters and a temporal precision of 2 hours. We use linear functions as depreciation functions in the Fuzzy Matcher[1]. The scores of the identified matches were evaluated and used to build a heatmap of the matches, shown in the left part of Figure

[1]Note that other functions could be used, however, the investigations of better configurations are out of the scope of the current study.

Figure 5: Heatmap of the twitter data distribution (left) and matches distribution (right)

5. We used the Twitter data to build the heatmap shown in the right part of Figure 5. It is important to notice that the matches in the Fuzzy Matcher heatmap reflect the amount and scores of the identified matches, rather than only a given amount in the Twitter heatmap. This correlation does not exclude the possibility of having the same Twitter sample matching different Yellow Taxis samples. These two factors may lead to greater values in the Fuzzy Matcher heatmap than in the Twitter one, which is expected. Furthermore, the Fuzzy Matcher heatmap considers the time variation, i.e., the scores of a region only increase if this region has matches closer in time and space, rather than an aggregated analysis where the time is not considered.

Looking at the heatmaps in Figure 5, it is possible to identify that regions with higher volumes of tweets resulted in regions with higher volumes of matches. This information by itself is not enough to prove the initial hypothesis, however, it acts as an indicator of its validity. After looking over all the heatmap, there are some specific regions that may be highlighted, identified with circles in Figure 5. In these regions, there is a significant amount of tweets, however, the scores of the matches are not relevant. The first region to observe is the one indicated by the red circle A (top of left image). This represents the Central Park area and it is clear that taxis will not be available there since they cannot access the park area. In the other two red circles B and C of the left image, it is possible to notice a smaller amount of tweets that are not reflected in matches, and, apparently, there is no reason for this lack of matches. In this case, we require a more elaborated analysis of the datasets, complemented by other sources, to ensure that assigning more taxis to those regions may increase the number of requested trips.

The second experiment aims to provide initial insights on the study of the Cross-layer User Mapping problem, i.e., to map data generated from the same entities in different sensing layers. The experiment consists in finding trips of taxis that have fuzzy matches, with the Twitter layer, in the beginning and at the end of the trip. We argue that the algorithm may capture trips that were performed by the owner of a given Twitter account. To check this hypothesis, we initially generated data to simulate a scenario where users actually tweeted, took a taxi and then tweet again. In this scenario, 500 Twitter users sent 20,000 tweets in one day. Also, 5,000 taxi trips were generated. We, then, executed a procedure on top of the results obtained by the Fuzzy Matcher in the simulation scenario. We analyzed the user ids contained in the matches in both layers. Thus, if we have a persistent match generated by two fuzzy matches $P = \{(A, B), (C, D)\}$, where A, B, C and D are nodes in the MAG structure, and also, without loss of generality, A and C belong to the Twitter layer, and B and D to the Yellow Taxis layer. Then, for P, it is valid that $A.uid = C.uid$ and $B.uid = D.uid$. Figure 6 shows the results obtained in the simulation.

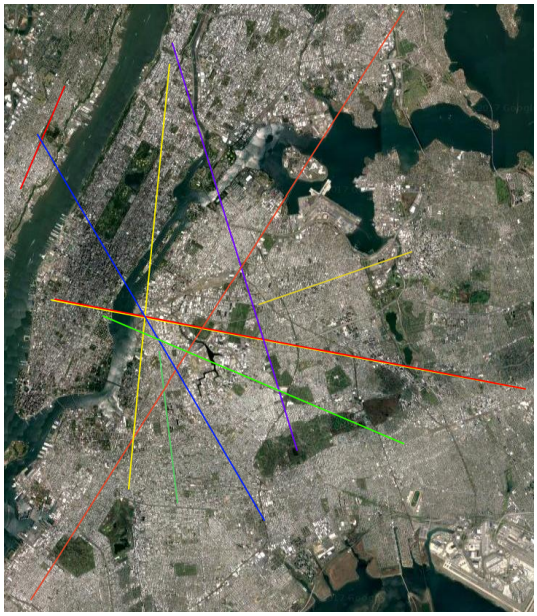

Figure 6: Simulated scenario to evaluate hypothesis

The Fuzzy Matcher algorithm allows the same Twitter layer node to match many Yellow Taxis nodes, and, thus, we can establish a lower bound of the distance, as depicted in Figure 6. To consider a persistent match and not only noise produced by the Fuzzy Matcher algorithm,

the persistent match distance needs to be at least 500 meters. Also, to ensure that the data has no noise, we defined a max speed limit used in the persistent matches to be 27.8 meters/second, or 100 Km/h, which may still be high. We discarded trips leading to a taxi or a Twitter user moving faster than this speed. These distance and speed limits are based on usual speeds and distances practiced by taxis in big cities, however, these values were only meant to waive the noise in the data and may not have implications on the analysis. As it can be observed, the algorithm identifies the persistent matches, showing that it can be used to investigate this sort of correlation between the sensing layers. There were identified 229 persistent matches in the simulated scenario, where the top 10 most relevant matches (i.e., greater distance and match score) are shown. After the simulation, the same method was applied to real data from New York City, and the result is shown in Figure 7.

Figure 7: Persistent matches indicated by Fuzzy Matcher

In the Yellow Taxi layer, we only have the times of the beginning and the end of each trip. After finishing a trip, there is no information if the same taxi was, or was not, used in another trip. Due to this limitation of the UID control in the Yellow Taxis dataset, every link represents a trip assumed to be performed by a different taxi. Also, colors of the lines represent the day when the trip happened. Finally, the icons associated with the persistent matches represent the Twitter user who created the data entry that resulted in that match. At the end of the procedure, 20,368 persistent matches were identified after filtering the noise. As in Figure 6, Figure 7 shows the top 10 most relevant matches. It is important to notice that the real data shows patterns of mobility. For example, most of the data is from a central region, which reflects the distribution shown in Figure 4. Furthermore, specific Twitter users are more likely to use the platform, resulting in more persistent matches for the same users, indicating their individual preferences.

Figure 7 shows evidence that the Twitter data may be used to map trips in the Yellow Taxis sensing layer. Silva et al. [20] suggest the use of the Twitter UIDs to work as a global identifier for data generated by human entities in different sensing layers. We could show that the Fuzzy Matcher algorithm can be one of the tools used to help creating this map of entities in different sensing layers. The information present in the experiment may not be enough to ensure all the persistent matches that reflect a real person using a taxi. However, due to the high amount of persistent matches and the alignment of trips' start and end locations with the user's locations, we claim that some initial analysis toward the solution of the cross-layer user mapping problem can be obtained with the SMAFramework and the use of the Fuzzy Matcher algorithm. As mentioned before, the problem of mapping the same entities in different layers is still an open issue in the literature [20].

8 CONCLUSIONS

Solutions for urban computing are enabling cities to provide better services for their citizens and optimize the usage of city resources. This work contributes in this direction, by introducing the SMAFramework to analyze mobility data in smart cities. This framework focuses on collecting spatio-temporal annotated data from different data sources, merge and standardize them to facilitate their analysis. To show the applicability of the SMAFramework, we analyzed data collected from two distinct sources in New York City. For that, we proposed an algorithm for mobility analysis, called Fuzzy Matcher. This algorithm allows its users to analyze spatio-temporal correlation (i.e., alignment/closeness of data entries in space and time) between data collected from different data sources. We argue that this algorithm may have different use cases, such as helping to deal with the cross-layer user/sensor mapping problem. There is still open opportunities to extend the framework, including more Data Extractor, Data Analyzers and Data Enrichment tasks. Indeed, the architecture of the SMAFramework was proposed to ease the addition of new modules in a plug-and-play fashion to maintain its evolution.

REFERENCES

[1] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record* 19, 2 (1990), 322–331. DOI: http://dx.doi.org/10.1145/93605.98741

[2] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. 2012. Time-Varying Graphs and Dynamic Networks. *CoRR* abs/1012.0 (2012), 20. DOI: http://dx.doi.org/10.1007/978-3-642-22450-8_27

[3] Thomas H Cormen, Clifford Stein, Ronald L Rivest, and Charles E Leiserson. 2001. *Introduction to Algorithms* (2nd ed.). McGraw-Hill Higher Education.

[4] Wei Feng, Chao Zhang, Wei Zhang, Jiawei Han, Jianyong Wang, Charu Aggarwal, and Jianbin Huang. 2015. STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream. *Proceedings - International Conference on Data Engineering* 2015-May (2015), 1561–1572. DOI: http://dx.doi.org/10.1109/ICDE.2015.7113425

[5] Daniel Figueiredo, Philippe Nain, Bruno Ribeiro, Edmundo de Souza e Silva, Don Towsley, and Edmundo De Souza E Silva. 2011. Characterizing Continuous Time Random Walks on Time Varying Graphs. *Sigmetrics* cs.SI (2011), 1–30.

[6] Vanessa Frias-Martinez, Victor Soto, Heath Hohwald, and Enrique Frias-Martinez. 2012. Characterizing urban landscapes using geolocated tweets. *Proceedings - 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust and 2012 ASE/IEEE International Conference on Social Computing, SocialCom/PASSAT 2012* (2012), 239–248. DOI: http://dx.doi.org/10.1109/SocialCom-PASSAT.2012.19

[7] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. 2013. Exploring temporal effects for location recommendation on location-based social networks. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13* (2013), 93–100. DOI: http://dx.doi.org/10.1145/2507157.2507182

[8] Hyoungshick Kim and Ross Anderson. 2012. Temporal node centrality in complex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 85, 2 (2012), 1–8. DOI: http://dx.doi.org/10.1103/PhysRevE.85.026107

[9] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. 2014. Multilayer networks. *Journal of Complex Networks* 2, 3 (2014), 203–271. DOI: http://dx.doi.org/10.1093/comnet/cnu016

[10] Vassilis Kostakos. 2009. Temporal graphs. *Physica A: Statistical Mechanics and its Applications* 388, 6 (2009), 1007–1023. DOI: http://dx.doi.org/10.1016/j.physa.2008.11.021

[11] Mohamed K El Mahrsi, Etienne Côme, Latifa Oukhellou, and Michel Verleysen. 2016. Clustering Smart Card Data for Urban Mobility Analysis. (2016), 1–17.

[12] Taewoo Nam and Theresa A Pardo. 2011. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference on Digital Government Innovation in Challenging Times - dg.o '11*. ACM Press, New York, New York, USA, 282. DOI: http://dx.doi.org/10.1145/2037556.2037602

[13] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. 2013. *Graph Metrics for Temporal Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 15–40. DOI: http://dx.doi.org/10.1007/978-3-642-36461-7_2

[14] Zhaolong Ning, Feng Xia, Noor Ullah, Xiangjie Kong, and Xiping Hu. 2017. Vehicular Social Networks : Enabling Smart Mobility. *IEEE Communications Magazine* 5 (jan 2017), 49–55.

[15] Kostas Patroumpas. 2013. Multi-scale window specification over streaming trajectories. *Journal of Spatial Information Science* 7, 7 (2013), 45–75. DOI: http://dx.doi.org/10.5311/JOSIS.2013.7.132

[16] Soledad Pellicer, Guadalupe Santa, Andres L. Bleda, Rafael Maestre, Antonio J. Jara, and Antonio Gomez Skarmeta. 2013. A global perspective of smart cities: A survey. *Proceedings - 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2013* (2013), 439–444. DOI: http://dx.doi.org/10.1109/IMIS.2013.79

[17] Loic Salmon and Cyril Ray. 2016. Design principles of a stream-based framework for mobility analysis. *GeoInformatica* (2016), 1–25. DOI: http://dx.doi.org/10.1007/s10707-016-0256-z

[18] Mehdi Sharifzadeh and Cyrus Shahabi. 2010. VoR-Tree : R-trees with Voronoi Diagrams for Efficient Processing of Spatial Nearest Neighbor Queries âĹŮ. *Proceedings of the 36th International Conference on Very Large Data Bases* 3 (2010), 1231–1242. DOI: http://dx.doi.org/10.14778/1920841.1920994

[19] T.H. Silva, P.O.S. Vaz De Melo, J.M. Almeida, and A.A.F. Loureiro. 2014. Large-scale study of city dynamics and urban social behavior using participatory sensing. *Wireless Communications, IEEE* 21, 1 (Feb 2014), 42–51.

[20] T. H. Silva, P. O. S. V. de Melo, J. M. Almeida, A. C. Viana, Salles J., and A. A. F. Loureiro. 2014. Participatory Sensor Networks as Sensing Layers. 386–393. DOI: http://dx.doi.org/10.1109/BDCloud.2014.27

[21] J. Tang, S. Scellato, M. Musolesi, C. Mascolo, and V. Latora. 2010. Small-world behavior in time-varying graphs. *Physical Review E* 81, 5 (2010), 055101. DOI: http://dx.doi.org/10.1103/PhysRevE.81.055101

[22] Gautam S. Thakur and Ahmed Helmy. 2013. COBRA: A framework for the analysis of realistic mobility models. In *Proceedings - IEEE INFOCOM*. 3351–3356. DOI: http://dx.doi.org/10.1109/INFCOM.2013.6567163

[23] Klaus Wehmuth, Artur Ziviani, and Eric Fleury. 2014. A Unifying Model for Representing Time-Varying Graphs. *Computing Research Repository arXiv.org* I, January (2014), 1–28.

[24] M Wolfe. 1989. More Iteration Space Tiling. In *Proceedings of the 1989 ACM/IEEE Conference on Supercomputing (Supercomputing '89)*. ACM, New York, NY, USA, 655–664. DOI: http://dx.doi.org/10.1145/76263.76337

[25] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Time-aware point-of-interest recommendation. *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13* (2013), 363. DOI: http://dx.doi.org/10.1145/2484028.2484030

[26] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 38.

[27] Y U Zheng. 2015. Trajectory Data Mining : An Overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 1–41. DOI: http://dx.doi.org/10.1145/2743025