Efficient distributed computation of human mobility aggregates through User Mobility Profiles

Mirco Nanni, Roberto Trasarti, Giulio Rossetti, Dino Pedreschi
KDD Lab - ISTI CNR
Pisa, Italy
name.surname@isti.cnr.it

ABSTRACT

A basic task of urban mobility management is the real-time monitoring of traffic within key areas of the territory, such as main entrances to the city, important attractors and possible bottlenecks. Some of them are well known areas, while while others can appear, disappear or simply change during the year, or even during the week, due for instance to roadworks, accidents and special events (strikes, demonstrations, concerts, new toll road fares). Especially in the latter cases, it would be useful to have a traffic monitoring system able to dynamically adapt to reference areas specified by the user.

In this paper we propose and study a solution exploiting on-board location devices in private cars mobility, that continuously trace the position of the vehicle and periodically communicate it to a central station. Such vehicles provide a statistical sample of the whole population, and therefore can be used to compute a summary of the traffic conditions for the mobility manager. However, the large mass of information to be transmitted and processed to achieve that might be too much for a real-time monitoring system, the main problem being the systematic communication from each vehicle to a unique, centralized station.

In this work we tackle the problem by adopting the general view of distributed systems for the computation of a global function, consisting in minimizing the amount of information communicated through a careful coordination of the single nodes (vehicles) of the system. Our approach involves the use of predictive models that allow the central station to guess (in most cases and within some given error threshold) the location of the monitored vehicles and then to estimate the density of key areas without communications with the nodes.

1. INTRODUCTION

In the context of urban mobility management, a basic task required by administrators is the monitoring of traffic within a variety of key locations: main gateways to the city, important attractors and possible bottlenecks. Some such locations ar well known, and therefore a monitoring environment can be set up by means of road-side sensors (including cameras), although set up and maintenance costs might be significant for large cities. Other key areas can appear, disappear or simply change with time, due to seasonality or special events. For instance roadworks, accidents or events such a strikes, demonstrations, concerts, new toll-road fares can change the status of the city, and make some areas more critical than usual. In these cases, it would be useful to have a traffic monitoring system able to dynamically adapt to reference areas specified by the user.

A solution can come from recent, growing trends in the deployment of on-board location devices in private cars mobility. Such devices continuously trace the position of the vehicle, and periodically communicate it to a central station, that stores it. Such vehicles provide a statistical sample of the whole population, and therefore can be used to compute a summary of the traffic conditions for the mobility manager. The analytical power of detailed and massive GPS trajectory in unveiling the patterns of human mobility behavior data has been shown in [1]. However, the large mass of information to be transmitted and processed to achieve that might be too much for a real-time monitoring system, the main problem being the continuous communication from each vehicle to a unique, centralized station. In this paper, we use a massive trajectory dataset consisting of approx. 1.5 million travels, sampled at a high rate from more than 40,000 private cars tracked for a month in a 50 square km area in Tuscany, Italy — a dataset which clearly illustrates the computational and economic challenge of continuous transmission to a central server.

Recently, *safe zones* were introduced as a principled mechanism for the efficient distributed computation and monitoring of a global aggregate function, consisting in minimizing the amount of information communicated through a careful coordination of the individual nodes (vehicles, in our domain) [2, 3]. The basic idea is that each node is instructed on how to check locally whether its changes of position can have a relevant impact on the global function, or not. In the negative case, no communication is needed. Of course, that implies a reasonable definition of *relevant impact*, as well as some computational capability at the node level to check it. The safe zone idea, realized through clever computational geometric methods, has the potential of drastically reducing the number of communications between the distributed nodes and the central station, and we checked empirically that this is the case also in our urban mobility setting.

In this paper, we ask the following question: can the amount of needed data transmissions from distributed cars to central station be further reduced by taking into account the regularity of human mobility? We know that the way people move is highly predictable: we tend to follow daily routines, dictated by our social constraints, so that the degree of entropy of our whereabouts very small, as shown by many recent empirical studies on large scale data on human mobility patterns and profiles [4, 5, 6]. Our idea is consequential: if human travel is often systematic and repetitive, we can exploit such regularity to avoid transmitting data whenever we follow our routines, and instead transmit when we are movements are outside our typical behavior. In this sense, our aim is to exploit the fact that the distributed system of cars and central station is *techno-social*, and therefore it follows not only general laws dictated by geometry and mathematics, but also statistical laws dictated by human behavior. We want to use both properties to optimize the distributed computation, and empirically measure the obtained results over realistic scenario. We describe in this paper how to achieve this goal based on mining different kinds of mobility profiles from the GPS trajectory data, and show how this novel data-driven approach significantly improves over the safe-zone approach.

2. RELATED WORKS

The topic of this paper lies at the crossroad of two research fields: the distributed computation of global functions (a specific instance of which is treat in this work), and the computation of predictive models for mobility.

The global function considered in this paper is essentially a sum of variables, each of them derived from the location of an object. Existing works in literature provide solutions for this case, for instance [2] deals with the problem of checking whether a linear sum of variables crosses a given threshold, and develops conditions that allow the central node to correctly test the threshold check even if some node does not communicate its latest values. More recently, also some general approach for very general classes of functions have been proposed in [3], essentially allowing any function that can be expressed as $f(\bar{\mathbf{x}})$, where $\bar{\mathbf{x}}$ is the average of the individual vectors of variables (one vector for each node of the network) and f is any function. The latter is based on the concept of *Safe Zones*, i.e. sets of values that an individual vector of variables can assume without affecting the global function significantly, i.e. as long as the vector lies within its Safe Zone, the global function is guaranteed not to cross the threshold even if the coordinator still uses older values of the vector. The Safe Zone approach works particularly well when the individual vectors are expected not to change too much in time, while they might be less effective when significant variations are common. In the specific context considered in this paper, the individual vectors are locations of vehicles and derived quantities, which typically can have large variations during the day, therefore the Safe Zone approach is expected meet efficiency issues. Similar considerations have been performed in [7], in the area of distributed query tracking. Their basic idea consists in combining data compression methods for limiting the size of the transmitted data (namely, *sketches*) with a predictive model that allows to avoid communication whenever a node *behaves as expected*. In our work we try to merge the Safe Zones ideas (though limited to the simplest case, since we deal with a linear function) with the use of predictive models suitable for mobility data.

The kind of predictive model required by our application should describe the expected mobility of a moving object throughout a typical day. Therefore, we are interested in extracting periodic patterns of movements, that link the routes followed with their time within the period (e.g. the hour of the day). In literature, the work in [8] approaches this problem by partitioning the period (e.g. the day) into time slots (e.g. hours), and defining a periodic pattern as a description of a representative location for each time slot (or * if no such representative can be found). Another approach, described in [5], consists in looking for typical trips, i.e. trips that repeat themselves approximately several time in the history of an individual, thus considering whole routes. The basic analytical methods derive from trajectory mining tools previously developed and combined into the M-Atlas framework [1]. Both approaches – location-based and route-based – are considered in this paper, adapted to our context and experimentally compared.

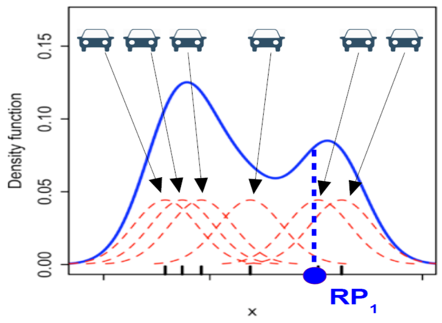

Figure 1: Example of vehicle density estimation for a reference point RP_1, on a single dimension, with a Gaussian kernel.

3. PROBLEM DEFINITION

Our reference application consists in evaluating the density of vehicles in correspondence of a given set RP of n_{RP} points in space, called *reference points*. In particular, density is estimated through a kernel-based approach, i.e., the density in a point is computed by counting all vehicles in space, yet weighted according to their distance from the point.

The application involves a central controller that computes (or estimates) the vehicle densities, and a set of nodes, each representing a vehicle. Each node receives a stream of location updates (coming from the on-board GPS device) and communicates the new location to the controller whenever needed to keep the global density estimates correct.

Definition 1 (DMP: DENSITY MONITORING PROBLEM). *Given a set $RP = \{RP_1, \ldots, RP_{n_{RP}}\}$ of n_{RP} reference points, a set $V = \{V_1, \ldots, V_{n_V}\}$ of vehicles and a kernel function $K(.)$, the* density monitoring problem *consists in computing, at each time instant, the function $f_{DMP}(V)$, defined as $f_{DMP}(V) = [K_1, \ldots, K_{n_{RP}}]^T$, where:*

$$\forall 1 \leq i \leq n_{RP}. \quad K_i = \sum_{j=1}^{n_V} K(V_j^{xy} - RP_i^{xy}) \tag{1}$$

Here, $V_j^{xy} \in \mathcal{R}^2$ and $RP_i^{xy} \in \mathcal{R}^2$ represent, respectively, the actual position of vehicle V_j and the position of reference point RP_i.

In this paper the kernel function used is a Gaussian as shown in Figure 1 where the the DMP for a single reference point is represented as sum of the contributions given by six vehicles.

Whenever the number n_V of vehicles or their location update frequency (or both) reach high values, it is necessary to trade the exactness of the estimation defined above with a reduction of information exchange and processing. The loss of precision, in our context, is bounded by a parameter ϵ, that represents the deviation from the exact output for the DMP.

Definition 2 (ADMP: APPROXIMATE DMP). *Given a DMP with reference points $RP = \{RP_1, \ldots, RP_{n_{RP}}\}$, vehicle set $V =$*

$\{V_1, \ldots, V_{n_V}\}$ and kernel function $K(.)$, and given an error tolerance parameter ϵ, the approximate density monitoring problem *consists in computing, at each time instant, a function $f_{ADMP}(V)$ that approximates f_{DMP}. In particular, given the following error function:*

$$error(K^A, K) = \max_{i=1}^{n_{RP}} |K_i^A - K_i|$$

where $K = f_{DMP}(V)$ and $K^A = f_{ADMP}(V)$, it always holds that $error(f_{ADMP}(V), f_{DMP}(V)) \leq \epsilon$.

In other words, given an error threshold ϵ we require that the density estimate of each single RP provided by f_{ADMP} differs at most of ϵ from the corresponding value provided by f_{DMP}.

Solving a DMP or a ADMP consists essentially in defining a process able to satisfy their requirements in every possible status and evolution of the overall system. The latter aspect can be modeled by a stream of status changes that each node senses during the monitoring period; the "process", then, basically defines a protocol used by nodes and controller to communicate only the essential information needed to satisfy the requirements of the (A)DMP.

In this paper several different ADMP solutions will be explored, in order to evaluate the impact of applying different levels of intelligence (in particular, learning from history) and usage of background knowledge.

4. BASIC APPROACHES FOR DISTRIBUTED DENSITY MAP MONITORING

Level 0: Communicate all

The trivial solution to the ADMP problem consists in having all the nodes sending an update to the controller for each update they receive. Obviously, that allows the controller to produce a perfect estimate of the global function (it actually yields a solution for the DMP problem, equivalent to $\epsilon = 0$), since it always knows the actual value of the variables it involves, at the price of communicating everything.

Level 1: static Safe Zones

This solution follows strictly the ideas based on Safe Zones [3], and therefore assumes that most objects are static or most of the time they move around some specific points in space, such as the home or work location. The basic idea, then, is to define a *default location* for each object v, and when no update arrives to the controller, it assumes that v is inside its default location.

More concretely, through analysis of historical data each node can be assigned to an optimal location that is used as its default position; then, basically the controller computes densities assuming that each node lies in its default position. Each node has assigned a geographical area such that, as long as it moves within that area the value computed by the controller is still a good approximation (w.r.t. the error threshold ϵ provided as parameter of the application). When the node moves outside its given area, it communicates the location update to the controller, which will use it to compute a correct estimation.

However, the context of mobility is characterized by massive and rapid changes in the data, since locations are highly dynamic, making this approach inadequate. For this reason, we will not further consider it, and instead will propose a variant that (in principle) better fits our scenario.

Level 2: adaptive Safe Zones

The basic assumption behind this approach is that the objects are not necessarily static, yet their movements are relatively slow. As an effect, when an object visits a given location, its associated region (see description of static Safe Zones above) will most likely contain several of the next locations of the object, yet no single location is able to capture a large part of the mobility of the object.

The protocol works as for static Safe Zones, but when an update must be communicated, the node is assigned to a new default location and to its corresponding geographical area, computed around its most recent measured location. Recomputing a new region (essentially, a new Safe Zone) is made possible and easy by the linearity of the global function to monitor (a sum of contributions), which enables to modify the Safe Zone of a node without compromising those of other objects. This kind of approach is much more problematic in contexts where the global function is more complex, since in those cases a change to a single object might involve changes to several other objects to reach an overall balance.

5. DISTRIBUTED DENSITY MAP MONITORING BASED ON PREDICTIVE MODELS

Since recent studies on human mobility claim that the latter is dominated by regular movements and therefore is highly predictable [4], here we analyze a segment of recent history of each node, in order to identify its regularities and use them as models to predict their locations in the next future. In particular, two variants of this idea are considered:

Most Frequent Location (MFL): we assume that the average user tends to visit (or cross) everyday the same places at the same hour of the day, therefore we look for single spatio-temporal locations that occur frequently, i.e., in many different days of the period. In this approach we do not try to link the frequent locations of a node, therefore the predictive model might contain consecutive sequence of locations that do not form consistent trajectories.

Mobility Profiles: here we make a stronger assumption, i.e. that the user tends to repeat the same trips everyday (home-to-work, and vice versa, for instance), thus involving an higher level concept of *frequent trip*, that requires a coherent sequence of spatio-temporal locations.

Both approaches create a typical daily schedule of each user, possibly with gaps for those moments of the day where the historical data did not reach a consensus on which location/trip to associate to them. The protocol, then, consists in letting the controller use at each instant the location predicted by the predictive model. In case of gaps (therefore no suggestion is provided by the predictive model) a *default model* is applied whose prediction is equal to the last known real location of the object. This is equivalent to adopt an *adaptive Safe Zones* solution limited to the gaps.

We remark that the Mobility Profiles approach implicitly adds a coherence constraint in the predictive model generation, therefore it will tend to produce predictive models with more gaps than the Most Frequent Location approach, yet the predictions provided are more likely to be reliable. Essentially, here we trading coverage for accuracy (to use information retrieval terms), and it is not clear a priori which solution might yield the best trade-off.

We collectively name the approaches mentioned above as protocols of the family *Level 3: predictive models.* In the following we describe the extraction and usage of the two variants.

5.1 Level 3.1: Most Frequent Location

5.1.1 MFL definition

In order to exploit the mobility habits of people, we start to build *schedules* of expected behaviors by using their most frequent visited locations. To do so, we need to acquire mobility information during a *training* period where the learning of habits will take place, then define frequency thresholds and build for each user a schedule that associates each time slot of the day to the most frequent location that occurred in that time slot throughout the training period, filtering out those locations that have an insufficient frequency w.r.t. the given threshold. The kind of model built with this approach is similar to the one described in [8].

Definition 3 (MFL USER DAILY SCHEDULE). *A MFL schedule is defined as the time-ordered set of the most frequent locations visited by an user within a specified observation periods. The daily schedule is discretized in time slots of equal durations.*

In order to identify what are the most frequent locations, defined by their GPS coordinates, we impose two constraints: (i) a time constraint, (ii) a spatial constraint. These two information are needed to build and align the daily schedule of a user.

Definition 4 (TIME SLOTS). *To determine to which time interval belong each single location we split each day in several slots of the same size. We define Δt as the time span that identifies the width of time frame reserved to each slot.*

Once defined a set of time slots we assign to each of them the set of locations visited by the user during the time slot. From this set we want to obtain a single representative location. A wide set of alternatives are possible to decide which location to choose as representative of the set (compute the center of mass, took the centroid, etc.). In this paper we choose to maintain the most dense location, i.e. the location that has the largest number of observations close to itself.

Definition 5 (SPATIAL RADIUS, NEIGHBORS). *Given a threshold Δs, called* spatial radius*, two locations A and B are considered* neighbors *if $||A - B||_\infty \leq \Delta s$, i.e. all their coordinates differs at most by Δs.*

5.1.2 MFL extraction

Once we have for a specified user a complete schedule of the visited locations during different days we need to synthesize a general schedule. In order to do it, we align all the daily schedules by collecting for each time slot all the observed locations that correspond to the time slot over the whole period; then, we calculate the most frequent location for each time slot. To avoid situations where the most frequent location appears only in a small fraction of the analyzed period, we impose a minimum support threshold.

Once this model is built we can use it as a proxy for the user mobility behaviour to the extent of predicting the location in which the user will be at a given time.

5.1.3 MFL-based prediction

The prediction phase rely on a direct query to the MFL schedule for the desired user. Given a query, defined as couple (u, t) composed by the user u and a timestamp t, we map t into the relative time slot and retrieve the MFL for the user u in that time slot, if it is defined. If a MFL for timestamp t does not exist, we apply a *default model*, that always suggests the last known location (i.e., the last one communicated to the controller).

5.2 Level 3.2: Mobility Profiles

5.2.1 Mobility profiles definition

We recall the concepts introduced in [5] where the user's history is defined as ordered sequence of spatio-temporal points $H = \langle p_1 \dots p_n \rangle$ where $p_i = (x, y, t)$ and x, y are spatial coordinates and t is an absolute timepoint. This history contains different trips made by the user, therefore in order to distinguish between them we need to detect when a user stops for a while in a place. This points in the stream will correspond to the end of a trip and the beginning of the next one:

Definition 6 (USER'S TRIPS). *Given the history H of a user and the thresholds $th^{stop}_{spatial}$ and $th^{stop}_{temporal}$, a* potential stop *is defined as a maximal subsequence S of the user's history H where the points remain within a spatial area for a certain period of time: $\overline{S} = \langle p_m \dots p_k \rangle | 0 < m \leq k \leq n \wedge \forall_{m \leq i \leq k} Dist(p_m, p_i) \leq th^{stop}_{spatial} \wedge Dur(p_m, p_k) \geq th^{stop}_{temporal}$. Finally we define a trip as the subsequence T of the user's history H between two consecutive stops in the ordered set $\overline{S}$ or between a stop and the first/last point of H.*

where $Dist$ is the Euclidean distance function defined between the spatial coordinates of the points, and Dur is the difference in the temporal coordinates of the points. Our objective is to use the user's trips in order to find his/her routine behaviors, this can be done grouping together the trips using a spatio-temporal distance function and extracting the medoid trip:

Definition 7 (ROUTINE). *Given a trip group g with at least th_{supp} elements and the distance function δ used to compute it, its* routine *is defined as the medoid of the set, i.e.:*

$$routine(g, \delta) = \arg\min_{t \in g} \sum_{t' \in g \setminus \{t\}} \delta(t, t')$$

where th_{supp} is the minimum size threshold used to reeve small groups which are not considered useful. Now we are ready to define the users mobility profile as the set of routine discovered over the history of the user:

Definition 8 (MOBILITY PROFILE). *Given a set of trip groups G of a user and the distance function δ used to compute them, the user's* mobility profile *is defined as his/her corresponding set of routines:*

$$profile(G, \delta) = \{routine(g, \delta) \mid g \in G\}$$

The mobility profile in other word represents a summarization of the movements of the user discarding the small variations which appear occasionally in his history.

5.2.2 Mobility profiles extraction

The extraction of mobility profiles from the user history is implemented as a sequence of modules which realizes the steps described above: Stop detection, Trip generation, T-Clustering equipped with a spatio-temporal function called Synch Route Similarity. The first module analyzes the user's history checking if the spatial distance between two consecutive points is lower than the threshold $th^{stop}_{spatial}$, when this happens the modules incrementally checks, and eventually stores, the following points until the constraint is not satisfied anymore. At the end of this process the module checks if the the sequences found satisfy also the temporal constraint using the $th^{stop}_{temporal}$ threshold, if this is satisfied the sequence will be considered as a stop for the user. The second module builds the trip

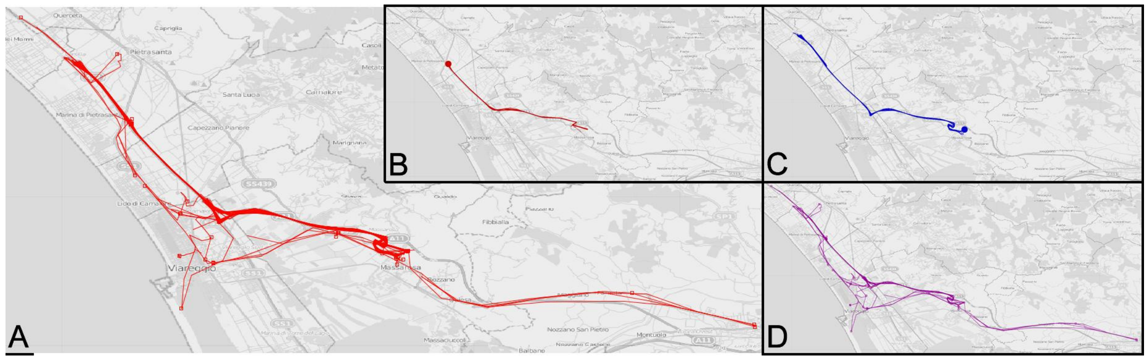

Figure 2: An example of mobility profile extraction: (a) The entire set of trips of a user, (b,c) the two clusters extracted, and (d) the remaining trips which are not periodic.

as sub-sequences of points between the begin and the first stop, each two consecutive stops and between the last and the end of the history. The last module runs a density-based algorithm called *T-Clustering* [9] using a spatio-temporal distance D_{SRS} which is a slight modification of the *Route Similarity*. This distance function starts comparing the initial timepoints of the two trips and it the temporal distance between the two are more than a give threshold (i.e. one hour) it returns an infinite distance without any further computation, otherwise it returns the distance computed as the route similarity. To obtain the clusters the T-Clustering algorithm checks if the following predicate is satisfied:

$$D_{SRS}(t1, t2) \leq (t1.lentgh + t2.length) * c_{PRadius}$$

where $c_{PRadius}$ is a the *Spatial Profile Radius* representing the tolerance used in the profile construction.

At the end of the process the clusters are filtered by their size, defined as number of trips, using the th_{supp} threshold. Finally, from each survived clusters a medoid is extracted and grouped obtaining the mobility profile of the user. In Figure 2 a real example is presented: here the user's trips are shown (a) including both the systematic and occasional ones, in (b) and (c) the two clusters extracted are presented showing a group of trips which are similar and synchronous, and in (d) the other trips which are the occasional movements that will be not considered. It is important to notice how the two clusters are very similar but reversed in the direction, this is usual due the fact that a big percentage of the users have two main reasons to move: going from home to work and viceversa.

5.2.3 Mobility profile-based prediction

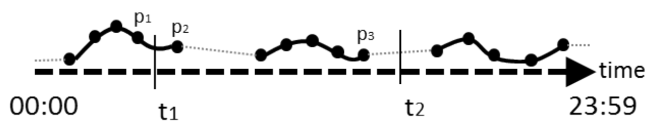

Figure 3: A profile composed by three routines. Only part of the day is covered, while *holes* are filled by the default model

Having extracted the user's mobility profile, we want to use it to predict the user's position at a certain time. It is important to notice that a mobility profile does not necessarily cover the whole daily schedule of a user. Let consider the two possible cases shown in Fig.3: (i) the prediction is made for the time instant t_1, corresponding to a period of the day where the profile is *defined*, and (ii) the prediction is made for the time instant t_2 corresponding to a period of the day where the profile is *not defined*.

In the first case the prediction will be the spatial interpolation between the two temporally closest points which surround t_1, namely p_1 and p_2. In the other case the prediction will be the last known point of the routine preceding temporally t_2, namely p_3. This corresponds to adopt a *default model* that always suggests the last known location, as done for MFL.

6. EXPERIMENTS

In this section we evaluate the different approaches presented in the paper, measuring the communications they save over the trivial protocol ("communicate all").

6.1 Dataset description

The dataset used in the following experiments is produced by a set of 40,000 cars, which represents the 2% of circulating cars in the coastal area of Tuscany. These points were tracked using GPS receivers with a sampling rate of 30s and a positioning system error of 10-20m in normal conditions over a period 5 weeks. The area covers a large territory with mixed land usages (residential areas, industrial zones, countryside, suburbs, etc.). The dataset was collected by Octotelematics S.p.A.[10], and a small sample is shown in Figure 4. Previous experiences on this data source (e.g. [1]) provided strong evidence of its validity and representativeness.

6.2 Experiment setup

A crucial aspect of the application is the position of the RPs in space. In order to test the effectiveness of the methods on a real scenario, we decided to use the positions of a set actual sensors used by the mobility agency in Tuscany, placed on the main gates of the city of Pisa plus one over the main bridge of the city center and two on two important neighboring towns. In Fig.5 the complete set of sensors is shown, and in Fig.6 a detail of Pisa is shown where each entrance of the city is monitored. The physical devices placed on the territory are permanent sensors based on laser technology, which can count the number and estimate the speed of cars passing nearby.

The testing of the methods presented in this paper requires to consider the following kinds of parameters:

- data-dependent parameters: in particular, we consider the sampling rate of the input GPS data in terms of average time

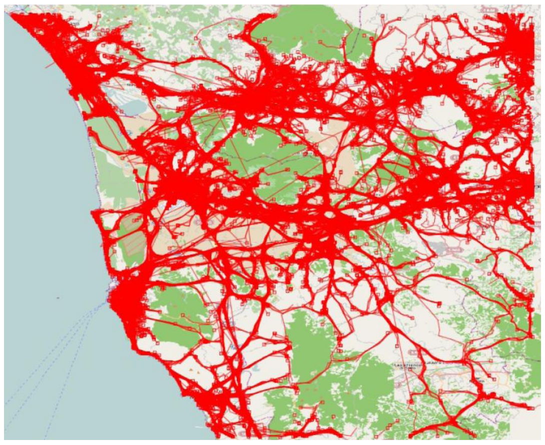

Figure 4: Sample of the dataset used for experiments

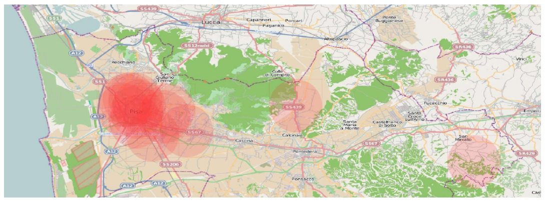

Figure 5: Location of RPs adopted in the experiments, and buffers representing kernel widths for the density computation

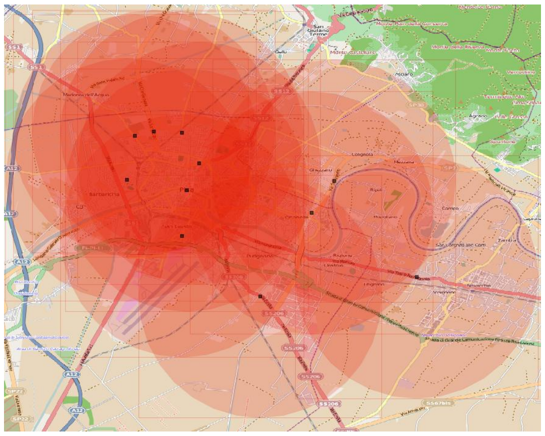

Figure 6: A detailed view of the sensors and the focal area in the city of Pisa

gap between consecutive location updates received by the vehicles. Where not explicitly mentioned, the sampling rate will be set to the default value of one point every 5 minutes (average);

- application-dependent: beside the set of RPs, which was chosen and fixed above, the application requires to specify (i) the width of the kernel adopted in computing the density over each RP, and (ii) the maximum (absolute) error tolerated in computing such densities. The width of the kernel is expressed as the distance for the vehicle at which its weight in the density computation is equal to 0.1. Where not explicitly mentioned, such width is set to 4 km. The error threshold, instead, is set to 5% of the overall average density of all RPs;
- predictive model-dependent: each predictive model is built on the base of its own parameters. In particular, we will explore the impact of the model spatial radius used, which defines how accurate must be the model. The lower is the radius, the higher is the accuracy but also the higher is the number of gaps in the model, since it is more difficult to find satisfactory predictive models. Where not explicitly mentioned, the spatial radius for the Most Frequent Location model is set to 1 km, and the Profile spatial radius is set to 0.3. Moreover, the temporal granularity adopted in MFL (i.e. the Δt used to define time slots) is set to the double of the data sampling rate, in order to have on average two points for each time slot.

In the following sections we will study the impact of the approaches proposed, and provide some spatial exploration of the results.

6.3 Data sampling rate

In this section we present an overall study of the performances of the system using the different methods discussed in the paper, compared against the trivial protocol *Level 0* ("communicate all"). In Fig.7 we show the communication rates varying the sampling rate of the data:

Adaptive SZ : the increasing trend shows that the adaptive Save Zones solution is affected by the sampling rate. Indeed, longer temporal gaps between location updates means an higher spatial distance between them, rising the probability of crossing the actual Safe Zone, and therefore requiring to communicate and update the Safe Zone more frequently;

MFL : we can see that the communication rate increases very quickly, due the fact that with an higher sampling rate the method cannot find (dense) groups in the time intervals. This affects mostly the MFL models of users with a small number of points, which become less stable or disappear completely;

Profiles : the Profiles-based solution appears extremely stable while changing the sampling rate, thanks to the fact that it tries to find a systematic whole trip of the user, with the result that the profiles which are extracted with different sampling rates are composed by less points but maintain their semantics, i.e. they still describe the same trip (though less accurately).

6.4 Application-dependent parameters

The impact of these parameters is very regular. Therefore, due to space limitation, we simply summarize their overall effect.

The width of the kernel (expressed as a distance, as described above) was studied in the range of values between 1 km and 10 km. In all methods applied, the communications increase monotonically with the kernel width.

Similarly, the density error threshold was studied in the range of values between 1% and 10% of the overall average of densities over all RPs. In all methods applied, the communications decrease monotonically with the error.

6.5 Model-dependent parameters: MFL

Fig.8 shows the number of MFL models created while changing the spatial radius parameter, hence the number of users which has a

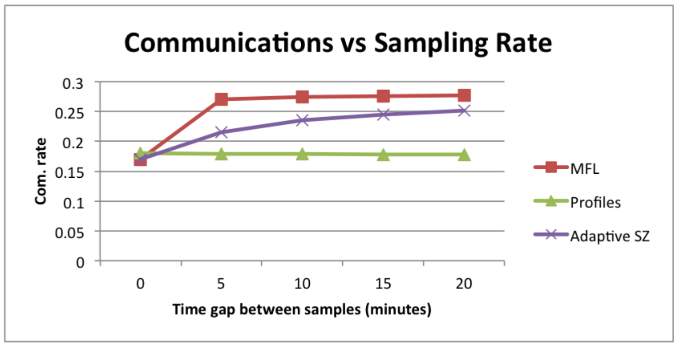

Figure 7: Overall analysis of the four methods varying the sampling rate of the data.

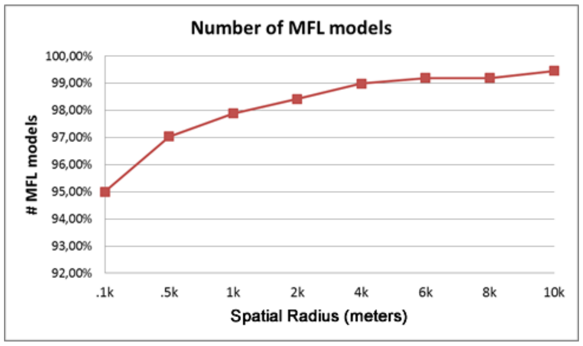

Figure 8: Number of MFL models created by the nodes using different tolerance values.

frequent behavior in at least a time interval. The trend is clearly increasing and tends to reach saturation. However, as shown in Fig.9, this does not mean that the performances of the method increase as well, in fact the total communication rate slightly increases with increasing tolerances, highlighting the fact that the MFL models created are not good in the prediction. The figure also shows the ratio of updates for which MFL could be applied (i.e. MFL provided a prediction) with success, thus saving a communication. Similarly, it shows the ratio of updates for which MFL could not apply, yet the default model successfully avoided the communication. We can see that the two ratios are rather symmetric, therefore resulting in overall very stable communication savings.

6.6 Model-dependent parameters: Profiles

In this section we present the performances obtained using the Profiles approach. As described in Section 5.2 here during the initial phase each node builds a profile and sends it to the controller. Then, when the system starts, the nodes check if their actual positions are coherent with their profiles. If not, they communicate to the controller, otherwise nothing is communicated, since the controller can predict the position using the profiles. In Fig.10 we show how the Profile spatial radius value changes the number of profiles extracted during the initialization phase: increasing the radius the number of profiles increases, i.e. the number of nodes who have a profile. Indeed, higher radii make the similarity between the user's trips less strict, thus making the formation of groups and profiles easier. It is interesting to notice how the number have a big increasing when the radius passes from 0.3 to 0.5 detecting a crucial

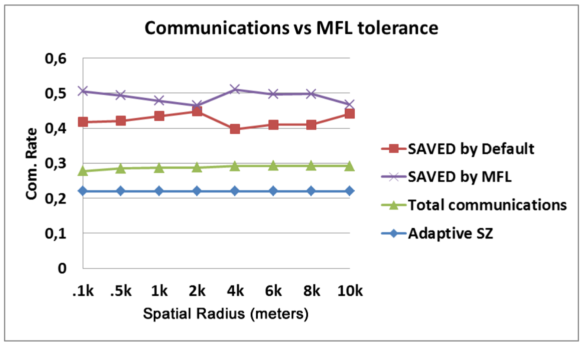

Figure 9: MFL performances compared to adaptive Safe Zones approach and the communication saved by MFL and default model.

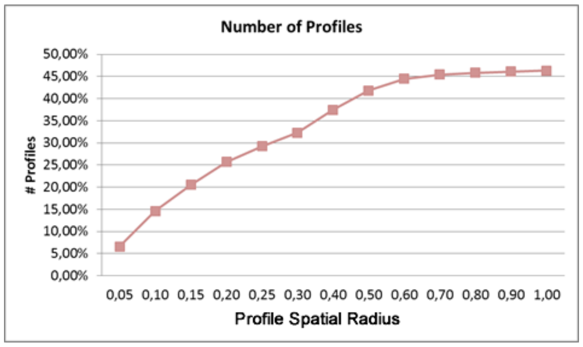

Figure 10: Number of profiles created by the nodes using different tolerance values.

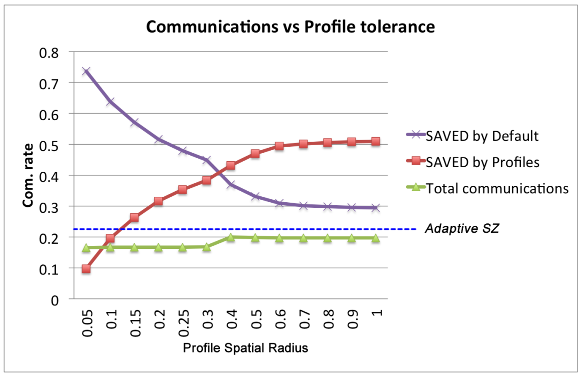

Figure 11: Profiles and adaptive Safe Zones performances and communication saved by profiles and default model.

point for the profile construction. Having more profiles does not mean to have better performances. Indeed, loose profiles lead to loose predictions. This can be seen in the Fig.11 where the performances of the system remain almost the same even if the number of nodes with a profile increases. Moreover, with radius equal to 0.3 the performances decrease, meaning that a critical point is reached and the profiles become too loose and the errors in profile prediction becomes higher. If Fig.11 we compare the performances of the profile approach against the adaptive Safe Zones. As we can see, Profiles gain a saving of 6.5%, meaning that the concept of profiles actually produces significant benefits. More in detail, analyzing the communications saved by the profiles and the default model (used when there is no profile to apply) we can see that profiles tend to replace the default model, improving the overall performances.

6.6.1 Spatial exploration of results

In this section, we provide an exploration of the performance results on the map. Fig.12 shows where the relative errors occur during the execution of the system. The color scale goes from red, representing a big percentage of errors, to blue which represents a small percentage of errors. Clearly, the error percentage is affected by the proximity to the RPs, in fact the error threshold is more likely exceeded in proximity of each focal point where the kernel function reaches the maximum. In other words, in those areas a small error in the location prediction leads to a big error on the density computation, therefore causing more likely a communication from the node to the controller. Fig.13 shows a detailed view of the city at

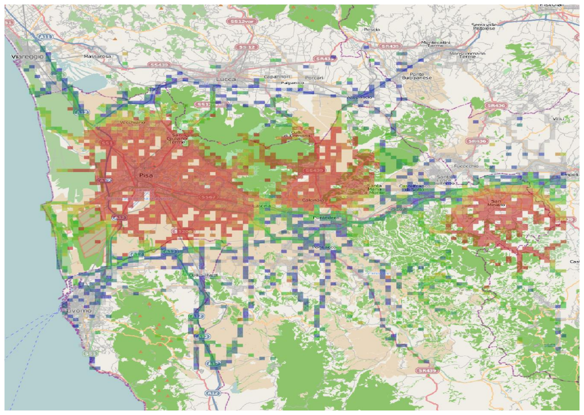

Figure 12: Distribution of relative errors occurred during the system execution.

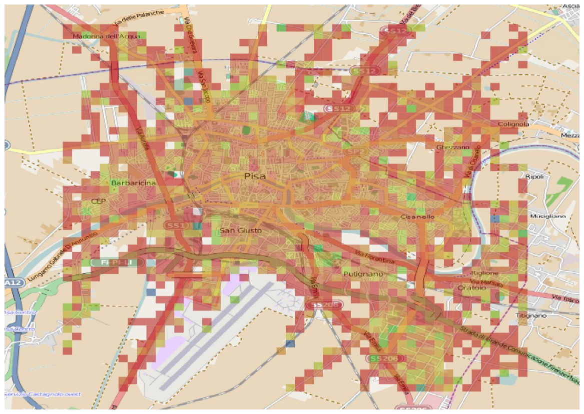

Figure 13: A detailed view of the relative error distribution in Pisa city using a smaller granularity.

a finer granularity, which is covered by several RPs. It is clear how the distribution of the errors in space is not homogeneous, in fact the city center (where an RP is placed) seems to be less affected by errors than the main gates and their relative roads.

7. CONCLUSIONS

In this work we developed and compared several approaches to the problem of computing population density over key areas in a distributed context, trying to reduce as much as possible the communication required. The approaches mainly differed for the way they tried to exploit the recent history of the moving objects involved, in some cases by estimating an optimal static default location for each object, in other cases by learning their mobility habits and exploiting them as prediction means. The experimental comparisons performed provided several insights on the effectiveness of each approach, in many cases with surprising outcomes.

Several new questions and open issues arose during the development of this work. We mention three of them: (i) since the computation involves potentially sensible information about individuals, can the proposed framework be made privacy-preserving? (ii) are there parts of the map more difficult to "learn"? E.g. highways are expected to be difficult, due to the high presence of occasional trips and occasional passers-by; (iii) if taken collectively, individual non-systematic behaviors might form typical paths, e.g. vehicles on the highways: how to integrate them in the framework? Aspects to consider on this way include the fact that typical paths cannot be associated to the vehicle ID (therefore there must be a different way to choose a "model" for a given model-less vehicle, such as prefix match), and to mine typical paths it is needed a centralized computation, therefore nodes might send to the controller, for instance, all trips not described by a profile.

8. REFERENCES

[1] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti, "Unveiling the complexity of human mobility by querying and mining massive trajectory data," *The VLDB Journal*, vol. 20, no. 5, pp. 695–719, 2011.

[2] M. Dilman and D. Raz, "Efficient reactive monitoring," in *INFOCOM*, 2001, pp. 1012–1019.

[3] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," *ACM Trans. Database Syst.*, vol. 32, no. 4, Nov. 2007.

[4] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[5] R. Trasarti, F. Pinelli, M. Nanni, and F. Giannotti, "Mining mobility user profiles for car pooling," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1190–1198.

[6] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabási, "Human mobility, social ties, and link prediction," in *KDD*, 2011, pp. 1100–1108.

[7] G. Cormode and M. Garofalakis, "Sketching streams through the net: distributed approximate query tracking," in *Proceedings of the 31st international conference on Very large data bases*. VLDB, 2005, pp. 13–24.

[8] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung, "Mining, indexing, and querying historical spatiotemporal data," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 236–245.

[9] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti, "Interactive visual clustering of large collections of trajectories," in *IEEE VAST*, 2009, pp. 3–10.

[10] "Octotelematics s.p.a.: http://www.octotelematics.it/."