Journal of Ambient Intelligence and Smart Environments 1 (2021) 1–5
IOS Press

A UAV Deployment Strategy Based on a Probabilistic Data Coverage Model for Mobile CrowdSensing Applications

Michele Girolami [a,*], Erminia Cipullo [b], Tommaso Colella [b] and Stefano Chessa [b]
[a] *Italian National Council of Research, ISTI-CNR, Pisa, Italy*
E-mail: michele.girolami@isti.cnr.it
[b] *Department of Computer Science, University of Pisa, Pisa, Italy*
E-mails: e.cipullo@studenti.unipi.it, t.colella@studenti.unipi.it, stefano.chessa@unipi.it

Abstract. Mobile CrowdSensing (MCS) is a computational paradigm designed to gather sensing data by using personal devices of MCS platform users. However, being the mobility of devices tightly correlated with mobility of their owners, the locations from which data are collected might be limited to specific sub-regions. We extend the data coverage capability of a traditional MCS platform by exploiting unmanned aerial vehicles (UAV) as mobile sensors gathering data from low covered locations. We present a probabilistic model designed to measure the coverage of a location. The model analyses the user's trajectories and the detouring capability of users towards locations of interest. Our model provides a coverage probability for each of the target locations, so that to identify low-covered locations. In turn, these locations are used as targets for the `StationPositioning` algorithms which optimizes the deployment of k UAV stations. We analyze the performance of `StationPositioning` by comparing the ratio of the covered locations against Random, DBSCAN and KMeans deployment algorithm. We explore the performance by varying the time period, the deployment regions and the existence of areas where it is not possible to deploy any station. Our experimental results show that `StationPositioning` is able to optimize the selected target location for a number of UAV stations with a maximum covered ratio up to 60%.

Keywords: Mobile CrowdSensing, UAV, Mobility, Coverage

1. Introduction

Building representative data sets from urban environments still represents a challenging task. The current literature offers a variety of data sets of different nature, ranging from mobility data sets collected with wearable devices or black-box units, to environmental data sets gathered with IoT-ready devices. However, the representativeness of the data collected still remains a challenge.

Under this respect, the Mobile CrowdSensing (MCS) [1, 2] paradigm represents a valuable paradigm to gather data with the active involvement of citizens. The MCS consists in recruiting volunteer users and provisioning them with a mobile app able to perform specific tasks, such as collecting user's feedback, producing multi-media contents, or sensing the environment. The MCS paradigm may also exploit wearable devices that people generally bring with them, in order to sense the surrounding environment, hence making the sensing capabilities even more sophisticated.

*Corresponding and principal author. E-mail: michele.girolami@isti.cnr.it.

The current literature reports interesting MCS initiatives, demonstrating the benefits of a participatory approach to collect data, such as environmental data sets, mobility traces, mobile phone usage and wireless signal coverage [3]. However, we argue that one of the main barrier to a massive adoption of the MCS paradigm is the capability of recruiting volunteers [4]. Indeed, many works [5, 6] address such issue, by proposing incentive mechanisms to further extend the MCS end-users and, in turn, the extend the representatives of the collected data.

In this work, we address a CrowdSensing scenario in the "urban scale", as from the classification provided in [7], in which we propose the adoption of UAV (Unmanned Aerial Vehicles) as a complementary mobile sensing unit able to further extend the amount and the quality of the collected data. In particular, since even despite of incentives or explicit requests the mobility of the MCS users is not controllable, the coverage of the point of interests for a MCS data collection campaign is in general not uniform and some points of interest for the campaign may result not sufficiently covered. However, these points cannot be determined precisely before the initiation of the campaign as they depend on the future mobility of the users during the campaign itself, while instead it is necessary to determine in advance the places where to physically deploy the UAV stations before the campaign. For this reason we adopt a probabilistic approach to model the expected coverage of points of interests by the users, even considering detours of users motivated by incentives proposed by the MCS platform, and we propose a deployment strategy for the UAV stations designed to optimize the amount of not sufficiently covered locations (according to the probabilistic model) that can be reached by a flock of UAV. Note that by coverage we do not mean any aspect related to wireless communications, but rather the sensing coverage, which is the probability that any user of the mobile CrowdSensing platform visits a location and collect data with his personal device. Hence the problem we address requires first to measure the data coverage of location l_h, namely the probability that a user accepts a detour to l_h to collect data, and then a probabilistic coverage model that considers the jointly contribution of all the user's trajectories to cover a set of locations of interest.

Given a sensing region, our model computes a coverage map from which it is possible to identify locations with low or high coverage probability. In turn, the coverage probability is adopted as heuristic to optimize the deployment of a set of UAV base stations. The deployment strategy analyzes the region of interest, and it maximizes the amount of uncovered locations that can be now covered with a UAV flying in a circle of center (x, y) and radius d_r meters. We do not model UAV flying dynamics, as our primary goal is to assess the effectiveness of the coverage model as a heuristic to deploy UAV in a sensing region. We also consider the not admissible areas, namely urban areas where it is not possible to deploy any station, due to restrictions, obstacles and any kind of flying regulations. In summary the novelties of this work can be summarized as follows:

- We propose a coverage model designed to measure the probability that a set of locations of interested is covered by users of a MCS system;
- We adopt the proposed model to generate a coverage map useful to identify high/low covered regions;
- We exploit the coverage map to drive the deployment of a set k UAV base station. UAVs are used as complementary sensing units with respect to user' devices. The location of the base stations is determined with a greedy approach:
- We evaluate the proposed UAV deployment strategy with a real-world experimental dataset, based on GPS trajectories. We analyse the mobility dataset and we describe the experimental settings.

We validate our solution by means of experiments based on an enhanced version of the GeoLife data set, extended with a trajectory generative method. In particular, we extended the number of users and of the followed trajectories given the top-most populated period of GeoLife (18 months). We show, with an extensive set of experiments, how the coverage probability can be used as a gradient for the optimization of a MCS data collection campaign. In particular, we compare our strategy against 3 representative benchmarks: a random deployment, DBSCAN and KMeans clustering algorithms in two scenarios: high and low density. Our results show that unsupervised approaches cannot identify the optimal location for a UAV station even with optimized clustering strategies. Differently, our strategy greedily identifies the location from which it is possible to reach the highest number of un-covered locations, with a relatively low number of iterations. The paper is structured as follows: Section 2 surveys the current state of the art in the fields of coverage models and base station deployment strategies; Section 3 describes our reference scenario, the coverage model and the deployment strategy we propose. We analyze in Section 4 the extended data set for our

experiments, with the description of our generative method to augment the user's trajectories. Finally, Section 6 covers the experimental settings and the obtained results.

2. Related Work

Monitoring an area by means of sensors so to guarantee a desired level of coverage is a relevant problem for sensing applications. Indeed, there are many studies addressing this problem under different assumptions concerning the sensing architecture, model and even the type of sensors involved. For example, [8] addresses the problem of coverage in a wireless sensor network in terms of the identification of sets with maximal residual energy able to cover all points of interest. The paper in [9] addresses the problem of coverage by taking into account the imperfections associated with sensor readings. Other works instead focus on coverage problems for sensors operating in line of sight [10], [11]: the first addresses the problem of coverage provided by a wireless visual sensor network by defining the Coverage Degree-coverage model, which quantitatively measures target detection probability, while the second formalizes the problem in terms of deployment of the sensors. More recently, due to the increasing availability of mobile CrowdSensing platforms and of UAV, the problem of coverage of a sensing platform enriched with the new dimension of mobility. Concerning the MCS, many works address the problem of coverage from the point of view of recruitment of participants and task injection [12, 13]. However there are also proposals of using interpolation strategies as a mean to increase the coverage of a mobile CrowdSensing platform [14]. Concerning UAV, the majority of studies available in the current literature focus on how to exploit UAV that carry base stations to provide wireless signal coverage to terminals on the ground [15–18]. In [15] is considered the problem of deploying a network of UAV to optimize the coverage of the ground terminals, by taking into account constraints, such as fault-tolerance and redundancy of the network. As the problem is NP-hard, the authors propose a solution based on a genetic algorithm meta-heuristic. The work described in [19] also considers the problem of minimizing the number of base stations mounted over UAV under the constraint of keeping all ground terminals connected. The problem can be formulated as the Geometric Disk Cover problem, whose objective is to cover a set of k nodes (GTs) in a region with the minimum number of disks of given radius r. Since problem is NP-hard, and for this reason the authors propose a low-complexity, nearly optimal (at least for small-sized problems) heuristic. In a similar architectural model, the work described in [16] aims at minimizing the number of UAV required while improving the coverage rate by optimizing the three-dimensional positions of UAV, ground terminals clustering and frequency band allocation. The proposed algorithm is then compared against known approaches like KMeans-based placement, ordered PSO-based placement, unordered ABC-based placement and Edge-Prior Placement algorithms. In [17] it is addressed the problem of optimizing the 3D placement (and repositioning) of a fleet of UAV under realistic constraints of inter-drone interference and connectivity. The problem is formalized as a mixed-integer non-convex optimization problem, for which the authors propose a polynomial time solution. The work in [18] considers two problems related to the deployment of UAV for providing ground connectivity. The first is to minimize the maximum deployment delay among all UAVs and the second is to minimize the total deployment delay. Authors propose a polynomial-time algorithm for the first and a linear approximation algorithm for the second problem. When the terminals on the ground are sensors, the problem of wireless coverage by means of UAV-mounted based stations changes slightly, as terminals on the ground are static and are typically subject to stringent constraints in terms of energy and communication capacity. The work in [20] addresses this specific problem, with the objective of maximizing the total capacity of the network. As the resulting problem is intractable, authors propose a game-theory approach. In particular they show that the resulting game has at least one Nash equilibrium point and they use a online learning approach to find such an equilibrium point. In [21] authors address the task allocation during package delivery activities. Authors refer to the possibility of employing sensing units on boards to collect data (similar to our approach), but the solution proposed in this work is based on a set of POIs (those with low-coverage values) previously computed by analysing the past user's mobility. Authors of [22] also consider a route allocation problem through a number of target points. However, authors do not mention how such POIs are selected. The UAV is constrained to fly-over all of them. Paper reported in [23] details a scenario similar to the one we propose. In particular, authors refer to the use of UAV to collect data from regions scarcely visited by the MCS participants.

Nevertheless, authors put emphasis on the battery consumption and on the quality of the collected data, as reported by authors.

Finally, there are a number of work that replace sensing elements at the ground directly with sensing elements mounted on UAV or on automated vehicles. For example [24] addresses a problem of distributed control and navigation of unmanned, sensing vehicles, with constraints related to the initial energy reserve, scattered charging stations and obstacles, while [25] suggests the use of UAV instead of CrowdSensing or road-side sensors to collect data about traffic. Authors of [26] propose to integrate data from a mobile CrowdSensing platform with 3D data collected by UAV to build a ground signal maps. Authors of [22] investigate a route planning problem for fixed-wing UAV tailored on the MCS scenario. Authors study the problem from a energy perspective with the goal of optimizing the task assignment to UAV and, at the same time, optimize the energy consumption of UAVs. Authors in [27] propose a formal model to identify the optimal number of nodes joining a CrowdSensing architecture, namely MEC nodes, so that to obtain a given spatial coverage. Although this work does not mention the use of UAVs, the approach adopted is interesting to design a MCS architecture. In [28], authors consider a mobile-vehicle-assisted MCS system. In this scenario, vehicles are owned by different operators or individuals competing for limited sensing resources of the vehicles. Authors study the joint task selection and the problem of planning routes for such an MCS system from an energy-efficiency perspective. Authors in [29] approach the problem on how to choose suitable participants to monitor an environment of a critical region for a MCS system. In [30], authors minimize the collection of redundant information to save resources. Authors of the following paper, approach the problem as a user selection strategy.

However, to the best of our knowledge, none of the previous works consider instead the problem of providing sensing coverage to an area by using an hybrid architecture that combines ground sensors of a mobile CrowdSensing platform and sensors mounted over UAV. The resulting coverage problem in such hybrid architectures is rather different than coverage with just UAV or just CrowdSensing because it combines two diverging aspects. On the one hand, the mobility of users in CrowdSensing platform is uncontrollable by the platform and the resulting coverage of the area is, in a sense, opportunistic. On the other hand, the mobility of the UAV is under control (although subject to constraints) and, as a consequence, the coverage of UAV is, in a sense, controllable. More specifically, the main advantages of the proposed model can be summarized as follows:

- data-driven deployment: the proposed algorithm identifies the optimal position for an UAV station by analysing mobility traces of users. This allows us to identify low/high covered locations based on real-world GPS traces. This aspect differentiates with respect to other deployment strategy based only on environmental features such deployment considering the coverage of a set of base stations, deployment by considering interferences and connectivity of terminals and strategies based on reducing the battery depletion of UAV;
- multiple station deployment: the proposed algorithm implements a procedure to deploy multiple UAVs station avoiding any overlapping. Moreover, the algorithm allows skipping regions where it is not allowed to deploy any station.

This is the reason leading us to propose a completely different approach, taking into account first the opportunistic coverage provided by the CrowdSensing platform, and focusing on the optimization of UAV base stations in order to provide the required coverage to the locations that remained uncovered appropriately by CrowdSensing.

3. The Reference Architecture and the Proposed Model

Our reference scenario is characterized by a traditional MCS architecture designed to collect sensing information from a set of locations of interest. The architecture we refer to is composed by a back-end server, a set of end-devices and a flock of UAV provisioned with sensing capabilities and grounded on a set of base stations.

The goal of the MCS data collection campaign is to gather data from locations of interests, by exploiting volunteer users. In particular, the back-end assigns *sensing tasks* [31] to the end-devices and it stores the collected data. A *task* is a specific action to be completed by the selected end-devices. Such action might require an active user intervention. Examples of sensing tasks are:

- environmental tasks: tasks designed to retrieve data measuring some environmental features, such as noise intensity, air quality or Wi-Fi signal coverage from areas of interest;

– user-generated tasks: tasks designed to collect contents provided by users, such as pictures or short move clips.

The end-devices are generally user's pocket devices such as smartphones, as they offer sensing capabilities through virtual and environmental sensors, commonly available on commercial products. Therefore, the locations from which data are collected strictly depend on the user's mobility. It is important to remark that the mobility dataset used for reproducing dynamics of the mobility determines which and how often regions are low/high covered. Indeed, a mobility dataset that has been collected during working weeks days, might reveal that the coverage value for entertainment regions is lower respect to a dataset collected during weekends, as users tend to visit such regions mainly during off-time hours. Furthermore, the time granularity of the mobility dataset also affects the accuracy for determining the coverage value. For this reason, it is important to clarify that the resulting coverage map used to identify high/low covered regions strictly depends on the goals of the CrowdSensing monitoring campaign.

Crowded locations have more probability of being covered in terms of data collected by end-users. Differently, peripheral locations might be highly un-covered as few people visit them along the day. Nevertheless, we observe that those locations scarcely visited might be still covered if at least one user accepts a *detour* towards such location. Detouring to a location implies that a user deviates from her intended path, to pass through a specific location. This is the case of a pedestrian that accepts walking towards a square close to her destination. The probability of detouring is directly influenced by a multitude of factors, such as the overhead of the detour, the existence of any reward, and the user's mean of transport etc. It is important to remark that, in our scenario, we do not require to fetch and process real-time user's locations, rather we only assume to obtain a snapshot of the user's mobility from which building the coverage values for the regions of interest. We stress the fact that the dataset is used only once and for the purpose of computing a coverage map, without the need to collect real-time user's trajectories. UAVs act only as a proxy from users to the back-end. The data produced by users may not be reliable due to many reasons, but the CrowdSensing paradigm is based on the idea of aggregating lot of data from the crowd to obtain a meaningful representation. In this work, we do not consider such aspect, but a variety of techniques can be used to evaluate the quality of acquired data as reported [32, 33]

Our reference scenario also comprises a flock of UAV extending the traditional MCS architecture. UAVs are provisioned with sensing capabilities and they can fly over specific locations in order to collect data with their sensors [34, 35]. In our model, UAVs are not employed to facilitate communication with users on the ground. Consequently, we do not account for aspects related to UAV communication capacity in our model. More specifically, we envision the UAVs as mobile sensors roaming in specific areas and exploiting on-board sensing units, such as temperature, air-quality sensors, on-board cameras etc. This aspect, allows us to avoid air-to-ground communication with user's devices. Furthermore, since our objective is the positioning of the UAV base station, we omit considerations related to UAV flight. Furthermore, as our goal is the positioning of the UAV base station, we disregard aspects related to the number of UAV and of their flight (such as UAV speed or wind [36]) and we consider instead parameters like UAV range and autonomy. Please refer to [34, 35] for details about performance of UAVs.

The goal of this work is twofold. On the one hand, we introduce the coverage metric which measures the probability that a location is reached by users of the CrowdSensing platform (even after a detour), hence this metric expresses the probability that a given place is covered from the point of view of sensing.On the other hand, we exploit such metric to identify those locations scarcely visited by the MCS users. More specifically, the meaning of the coverage is measuring the probability that a location will be visited by at least one user, and in turn, to collect data from nearby of such location. The higher the coverage of a location, the more likely we can expect to retrieve data from it. Differently, the lower a location is covered, the smaller is the probability of having data from such location. In order to overcome this last case, we exploit the coverage map for a set of locations of interest, to identify strategic location for the deployment of k UAV base stations. The identified locations are selected to deploy a set of base stations from which a flock of UAV can take off and retrieve sensing data. It is worth to notice that we consider a reduced UAV flying model.At this stage, we are not interested to simulate realistic flying conditions such as the speed, wind direction, effect of turbulence and its impact on the battery depletion. Rather, we aim at quantifying the potentialities of the proposed coverage metric for the purpose of optimizing the location for a set of UAV base stations.

As a representative example, we report in Figure 1 our reference architecture. The red box shows the selected base station (BS) from which 2 UAVs fly towards un-covered locations (represented as purple circles). Data collected

from the 2 UAVs are then transferred to the MCS back-end. The operating radius of the UAV is fixed to d_r. We also show a detour example in with the path followed by an end-users change from a detour from $A \rightarrow B$, to $A \xrightarrow{L_k} B$.

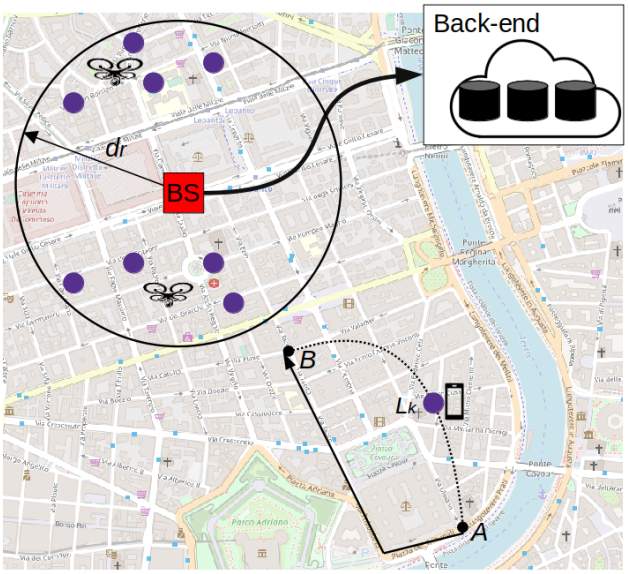


Fig. 1. The reference MCS architecture showing the use of a UAVs and a detoured path.

Note that in this work we disregard the problems that may arise due to failures of end-devices providing information (included location information) to the CrowdSensing platform, which, from our point of view, may result in an incomplete coverage map. In fact, this is a more general problem concerning the quality of information in the CrowdSensing platform as a whole, and, as such, we expect to be addressed by using state of the art solutions by the platform itself, not only to obtain collect and up-to-date location information from the end devices, but for all collected data in general. We refer the readers interested in the reliability of CrowdSensing platforms and in the quality of information they produce to the following works [37–39].

3.1. The Probabilistic Model

We present the probabilistic model for measuring the coverage, some key components lead our model: 1) exploit a coverage map to *drive* the deployment of an UAVs station 2) enabling the possibility of deploying multiple UAV stations at the same time 3) consider some features of UAVs, such as the operating radius, un-admissible areas etc. The adopted notation is reported in Table 1.

We define $L = \{l_h : h \in [1, H]\}$ the set of locations of interests, with $|L| = H$ and $U = \{u_k : k \in [1, K]\}$, $|U| = K$ the set of end-users joining a Mobile CrowdSensing data collection campaign. Every user $u_i \in U$ moves along a set T_i of J_i trajectories, that is, $T_i = \{t_{i,j} : j \in [1, J_i]\}$, where each trajectory is a finite and ordered sequence of way-points. Given a location of interest l_h and a trajectory $t_{i,j}$, it is possible to measure the physical distance between each way-point in the trajectory and l_h, e.g with the geodetic distance. We hence define the distance $t^h_{i,j}$ between trajectory $t_{i,j}$ and location of interest l_h as the minimum of the distances between the way-points of $t_{i,j}$ and l_h.

We define *coverage* c_h of a location l_h the probability that any of the users in U accepts detouring towards l_h. With the term detouring, we refer to a user that deliberately accepts altering its original trajectory in order to pass close to a location of interest. Following an intuitive observation, the probability of detouring increases as users get closer to the target location, while such probability decreases as users are far from such location. Note that, as our concept of coverage relates to sensing, how much the user should come close to the location of interest depends on the sensing coverage of sensors required to perform the task at hand. As in this paper we are not interested to any

Table 1
The adopted notation.

l_h	**location h**
$T_{i,j}$	j-th trajectory of user i
L	the set of locations
U	the set of users
X_h^i	random variable for modelling detour
$Y^{i,j,h}$	random variable for modelling detour toward l_h
W^h	random variable modelling the coverage
Z	set of not admissible areas
c_h	coverage of location l_h
τ	coverage threshold
d_r	UAV distance radius
δ	distance measure

specific sensing technology, without loss of generality we assume that a location is covered if the user goes to that precise location.

We define the random variable $X_i^h : \Omega \to \mathbb{R}^+$, with Ω the set of events in the form: user $u_i \in U$ accepts detouring towards $l_h \in L$ *up to* distance $\bar{t} \in \mathbb{R}^+$ from any of her trajectories. As a representative example, $X_i^h = 500$ is the event that user u_i accepts a detour from distances bound in: $[0, 500]$ meters from location l_h. Events in Ω are continuous in $\mathbb{R}^+$, with $f_{X_i^h}$ the probability density function (see Section 5 for an instance of such distribution) and $F_{X_i^h}$ the cumulative distribution function.

We now define a random variable for each of user u_i's trajectories, the random variable is defined as follows:

$$Y^{i,j,h} = \begin{cases} 1 & u_i \text{ detours from } t_{i,j} \text{ to location } l_h \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$\forall t_{i,j} \in T_i$, the family of random variables $\{Y^{i,j,h}\}$ are defined by two opposite events: detouring towards l_h or not. If a user accepts a detour up distance $\bar{t}$ from location l_h with probability p (as defined with the Equation 1), then the user will also accept a detour at distance $t \leqslant \bar{t}$ from l_h, as she/he is closer to the target location. Therefore, we define the probability $P(Y^{i,j,h} = 1)$ of detouring at distance $\bar{t}$ from l_h as:

$$P(Y^{i,j,h} = 1) = 1 - F_{X_i^h}(t) = \int_{t_{i,j}^h}^{\infty} f_{X_i^h}(t)dt \tag{2}$$

We now generalize the random variable defined in Equation 1, with the random variable $K^{i,h}$ modelling two events: the user u_i detours towards l_h and its opposite event:

$$K^{i,h} = \begin{cases} 1 & \text{user } u_i \text{ detours to } l_h \text{ from any of trajectories } T_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

thus, assuming that events $Y^{i,j,h}$ are all independent. In particular, we assume that the same user decides to detour towards l_h independently from the fact that she/he already detoured to another location at any distance.

the probability $P(K^{i,h} = 1)$ is given by:

$$P(K^{i,h}) = 1 - \prod_{\forall t_{i,j} \in T_i} (1 - P(Y^{i,j,h})) \tag{4}$$

More specifically, we compute the probability of *not* detouring from *every* trajectories $t_{i,j} \in T_i$ and we compute the opposite probability, namely the probability of detouring from any of the trajectories $t_{i,j} \in T_i$.

Finally, we define the coverage probability for a location. We define the random variable W^h modelling two opposite events: at least one user detours towards l_h and its opposite event:

$$W^h = \begin{cases} 1 & \text{location } l_h \text{ is covered by at least a user in } U \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

thus, assuming that the events $K^{i,h}$ are all independent, the probability $P(W^h = 1)$ is given by:

$$P(W^h) = 1 - \prod_{\forall u_i \in U} (1 - P(K^{i,h})) \tag{6}$$

obtained by the opposite probability that none of the users in U detour towards l_h. This last equation implements our coverage model used on the our experimental analysis.

3.2. The Problem Statement

We now describe the UAV deployment problem, whose goal is the optimization of the position of a set of k base stations for UAVs. We assume that the locations of interest are within a plane region delimited by a bounding box B. Moreover, we consider a simple UAV flying model according to which:

- UAVs have a fixed operating radius referred to as d_r meters from the base station;
- UAVs fly at a fixed altitude from the ground enabling the data collection (e.g. sampling environmental metrics);
- UAV to BS communications follow LOS model, hence the UAV to BS channel power gain can be considered as a free-space path loss model.

Moreover, the proposed model has been designed by considering the following assumption:

- The detour probability varies with a truncated exponential distribution determining the likelihood of accepting/rejecting a detour toward a location of interest;
- If at least one user visits a location, then we assume such location becomes covered. In particular, we assume that it is possible to collect sensing data from every visited location;
- We do not consider the impact of atmospheric agents to the UAV deployment.

Let $L = \{l_h = (x, y) : h \in [1, H]\}$ be the set of locations of interest in B, where (x, y) identify to the longitude and latitude of the location according to a reference system (e.g. EPSG 4326). For each location $l_h = (x, y)$ let c_h be its coverage value, corresponding to the coverage probability of location (x, y) computed according to the model proposed in 3.1 and formalized with Equation 6. We also assume that within B, there exist locations that are not admissible as UAV station(s). Ideally, the admissible areas are those free from obstacles, prohibitions or restrictions. To this purpose we define the set Z of not admissible locations as the union of $\hat{z}$ elliptic regions Z_i (with $i \in [1, \hat{z}]$), each of center z_i and radius ρ_i, where $Z_i = \{(x, y) : \delta(z_i, (x, y)) < \rho_i\}$.

Our optimization problem aims at finding the best location (x, y) for a base station so that:

- the select base station lies in B but not in any of the not admissible areas;
- the UAVs can reach locations with low coverage probability. This constraint allows to use UAV only for locations scarcely visited by the MCS end-users;
- the distance between the base station and the target locations is always lower than the maximum operative range of a UAV, namely the operation radius d_r.

We formalize the addressed problem as follows:

$$BS(1): \begin{cases} max|T_{x,y}| \\ \text{where} \\ (x,y) \in B \setminus Z, \\ T_{x,y} = \{l_h = (x',y') \in L : \\ \quad c_h \geqslant \tau \vee \\ \quad \delta(l_h,(x,y)) \leqslant d_r\} \end{cases} \tag{7}$$

where τ is a parameter modeling the coverage of a location (if $c_h \geqslant \tau$ we assume that the location l_h is covered); δ is a distance measure between two points (e.g. the geodetic distance), and d_r is the maximum operative range of a UAV, that allows the UAV itself to return to the station after the mission (for example, if the maximum travel distance of an UAV is $500m$, then d_r could be $250m$). The set $T_{x,y}$ identifies the locations of interest already covered, namely those locations with a coverage probability greater than τ, or locations at most distant d_r from the base station in (x,y). Therefore, our problem aims at maximizing the set $T_{x,y}$, by identifying the optimal location for the UAV base station given the two aforementioned constraints. It is worth to notice that more accurate models can be adopted to constraint the maximum travel distance for UAVs, an interesting approach is detailed in [40].

3.3. The Proposed Deployment Algorithm

We now describe our heuristic algorithm solving the problem defined in Equation 7. We introduce the solution for a single UAV station, and then we briefly discuss how this process can be generalized to deploy k UAV stations. We also present in this section the analysis of the computational complexity of the proposed algorithm.

Our strategy is inspired by the Spiral MBS Placement Algorithm [19], however our approach provides additional features not present in [19]:

- we exploit the coverage probability model to determine the initial set of locations suitable for UAVs;
- we optimize the number of required iterations to find the optimal solutions by pruning out *inefficient* locations;
- we model the not admissible areas preventing the possibility of deploying a base station arbitrary;
- we test our strategy with a real-world mobility data set.

The strategy consists of three stages: i) building the coverage map for B; ii) filtering out locations of interest with high coverage value (as already covered by the MCS end-users, hence not suitable for a UAV fly over) and iii) identifying the base station by excluding not admissible areas.

Figure 2 visualizes the pipeline of our strategy. During the first step, we compute for every location in L its coverage value. The figure shows in red and orange colors the locations with high coverage values, while in purple the locations with low coverage values. The second stage of the pipeline removes the locations with coverage greater than τ as they are already covered (or potentiality covered with a detour) by the MCS users. Lastly, during the third stage, we identify the optimal locations of the UAV station, by drawing a circle of center (x,y) and radius d_r such that the circle maximizes the amount of uncovered locations (purple circles). During this stage, the strategy excludes the polygons marked with dashed lines in Figure 2, as they represent the not admissible areas not allowed for any deployment.

The strategy is implemented with Algorithm 1 `StationPositioning` that takes in input the set L of locations of interests (see Section 3.2) and set Z of not admissible locations. The algorithm returns the coordinates $(\bar{x},\bar{y})$ chosen for the UAV station, the set $T_{\bar{x},\bar{y}}$ of locations covered by the station and the radius of the station $r_{\bar{x},\bar{y}}$ that corresponds to the maximum distance the UAV grounded in $(\bar{x},\bar{y})$ is able to reach.

`StationPositioning` first performs some initialization (lines 2-4), in particular it sets the initial position of the station $\bar{x},\bar{y}$ in a not admissible point (line 2) and it defines the set of covered locations $T_{\bar{x},\bar{y}}$ as empty (Line 3). Then it constructs the list C of the location of interest that are not already covered by the mobile CrowdSensing platform, that is, the locations with coverage smaller than τ (line 5). In line 6 the algorithm computes, for each

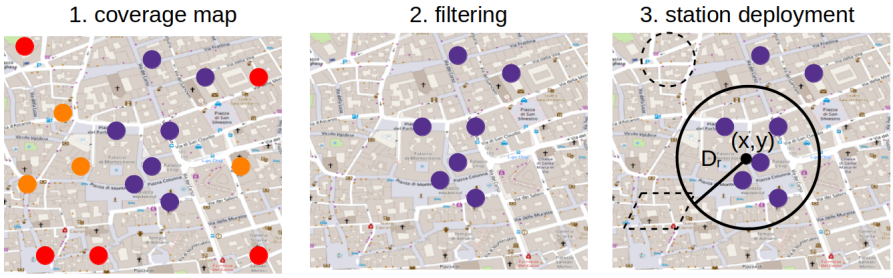


Fig. 2. Step of the UAV station deployment strategy.

location $l_h \in C$ to be covered, its maximum coefficient m_h that measures the number of other locations in C at a distance at most $2*d_r$ from (x,y). More specifically, the maximum coefficient is the maximum number of uncovered locations of interest bound within a circle of radius $2*d_r$ and of center (x,y). The algorithm computes m_h for every location in C and it then sorts the locations in C according to their values of m_h in descending order (line 7). Then, the algorithm iterates over C in order to deploy a station. Line 9 of `StationPositioning` implements an optimization: for a given location $l_h \in C$, if its maximum coefficient m_h is lower or equal than the cardinality of the set of locations of interest already covered by the current position $(\bar{x},\bar{y})$ of the station, then the algorithm already found a better location where to deploy the UAV station and the location l_h is disregarded. Indeed, placing a station in $(\bar{x},\bar{y})$ allows to cover a higher number of points with respect to placing in l_h. If this is the case, then the algorithm interrupts with the current iteration and returns the current solution. Differently, the algorithm invokes the `DeployStation` function that evaluates the current point l_h (line 11). The result of `DeployStation` is a valid solution for the deployment whose return values have the same meaning of those of `StationPositioning`: the coordinates of the UAV station (x',y'), the set of locations covered $T_{x',y'}$ and the radius $r_{x',y'}$. If the location (x',y') for the station is better than the current one (line 12-14) then the solution is updated with the new one (lines 15-17) and the algorithm iterates until case evaluated in line occurs or all the points in C had been considered. The algorithm then ends returning the location $(\bar{x},\bar{y})$ as the chosen location for the station (line 18).

Algorithm 1 StationPositioning

```
1:  function StationPositioning(L,Z)
2:      (x̄, ȳ) ← (∞, ∞)
3:      T_{x̄,ȳ} ← ∅
4:      r_{ȳ,ȳ} ← d_r
5:      C ← list of l_h ∈ L : c_h < τ
6:      Let  m_h = |{l_h' ∈ C : δ(l_h', l_h) ⩽ 2 * d_r}|  ∀l_h ∈ C
7:      C ← sort(C, m_h)
8:      for all l_h ∈ C do
9:          if |T_{x̄,ȳ}| ⩾ m_h then break
10:         [(x', y'), T_{x',y'}, r_{x',y'}] ←
11:             DeployStation(l_h, C \ l_h)
12:         if (|T_{x',y'}| == m_h)∨
13:                 (|T_{x',y'}| > |T_{x̄,ȳ}|)∨
14:                 (|T_{x',y'}| == |T_{x̄,ȳ}| ∧ r_{x̄,ȳ} > r_{x',y'}) then
15:             (x̄, ȳ) ← (x', y')
16:             T_{x̄,ȳ} ← T_{x',y'}
17:             r_{x̄,ȳ} ← r_{x',y'}
18:     return[(x̄, ȳ), T_{x̄,ȳ}, r_{x̄,ȳ}]
```

The `DeployStation` algorithm identifies the optimal deployment by analyzing the current position l_h and the set $\hat{C} = C \setminus l_h$, as reported in Algorithm 2. The goal of `DeployStation` is to find a deployment solution able to cover at least the point l_h and the highest number of points in $\hat{C}$. To this purpose, `DeployStation` performs the following operations:

- checks that the location returned is not included in any of the not admissible areas, as implemented with Algorithm 4 `checkCenter`;
- updates the set of covered locations with Algorithm 3 `updateLocations`;
- finds the smallest circle containing the maximum number of locations in C, by exploiting the Algorithm `1-Center` as implemented in [41].

`DeployStation` initially checks that the point $l = (x, y)$ in input is not included in any of the not admissible areas (line 3). If this is the case, then `DeployStation` interrupts its execution and it skips to the next iteration of `StationPositioning`.

Differently, `DeployStation` invokes Algorithm 3 `updateLocations` which performs the following two operations:

- fills $\bar{C}$ with the locations in $\hat{C}$ at a distance lower or equal than d_r. These locations can *surely* be covered from a station centered in (z, w);
- removes from $\hat{C}$ the locations in $\bar{C}$ and those at distance greater to $2 * d_r$ as locations that *surely* cannot be covered. As a result, the set $\hat{C}$ will contain locations at distance $d_r < d \leqslant 2 * d_r$. These locations *might* be covered by a station centered in (z, w).

Figure 3 shows a graphical representation of the `updateLocations` algorithm. Locations in violet color are those included in $\bar{C}$ as d_r distant from (z, w), locations in orange color are the ones includes in $\hat{C}$ as $d_r \leqslant d < 2 * d_r$ from (z, w). Locations in red color are those ignored as too far from (z, w).

`DeployStation` finds a valid solution by iterating over the items in $\hat{C}$ (line 7) (from the closest to the far point to (z, w)) and it finds the closest circle including the items in $\bar{C}$ (the priority ones) *and* the point l with the `1-Center` algorithm (line 10). If the solution found by `1-Center` is admissible (line 11-13), then `DeployStation` updates the locations with `updateLocations` and it moves to the next item in $\hat{C}$, otherwise `DeployStation` interrupts and it returns the previous solution.

Finally, Algorithm 4 `checkCenter` is in charge of checking if the candidate location found by `DeployStation` is admissible. If not, it re-deploys the station's position. In particular, `checkCenter` computes the intersecting areas of all the disks D_h whose centers are the locations $l_h \in \bar{C}$ and that have of radius d_r, by excluding however the not admissible locations in Z (line 4). If the resulting set is non empty, than any point in this intersection is a valid candidate position for the UAV station. Otherwise it returns the not admissible solution (∞, ∞) (line 5).

The algorithms we described allow the deployment of one UAV station. However, a similar process can be applied for the positioning of k stations. In particular, this generalization consists of executing k iterations of `StationPositioning` and by checking that stations i does not overlap with station $i - 1$. As an example, given $k = 2$ stations we check that the circle of center $(\bar{x}, \bar{y})$ obtained for station 2 does not overlap with the circle for station 2. In other words, once a station is deployed, we consider the corresponding circle of center $(\bar{x}, \bar{y})$ and radius $r_{\bar{x},\bar{y}}$ a new not admissible area.

We now discuss briefly some considerations about the computational complexity of Algorithm 1 `StationPositioning`. Its input consists of a number of locations of interest (set L, of size n), and of set Z (of size z) of not admissible areas. In particular, the algorithm extracts from set L the set C of locations that need to be covered, and set Z consists of a set of constraints in the form $\delta((x, y), (zx_i, zy_i)) \geqslant d_i$, each expressing the fact that the circular region centered in (zx_i, zy_i) and of radius d_i is not admissible as UAV base station. As in any practical application the size of these two sets is limited, the actual implementation of the algorithm is rather fast, as also observed in the simulations. In any case, we consider that the outer cycle of `StationPositioning` (line 8) is repeated n times, and that the complexity of each cycle equals the complexity of `DeployStation`. In turn, `DeployStation` repeats its main cycle (line 8) up to $n - 1$ times (note that as at each cycle the size of C reduces of at least one element), and each cycle invokes `1-Center` (line 10, of linear complexity in n [41]), `UpdateLocations` (line 16, also of linear complexity in n), and `CheckCenter` line 12. Concerning this last function, we observe that its purpose is to find,

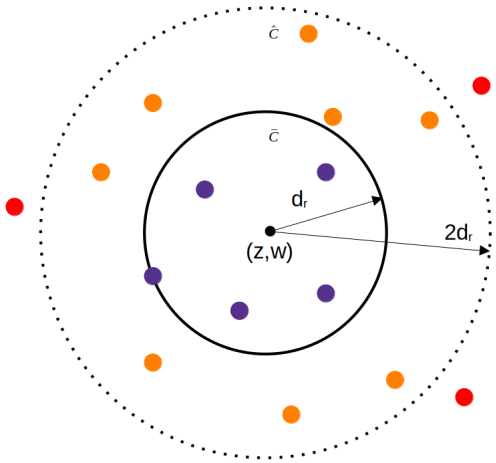


Fig. 3. Graphical representation of sets $\bar{C}$ and $\hat{C}$ implemented with Algorithm 3.

Algorithm 2 DeployStation

1: **function** DEPLOYSTATION($l, \hat{C}$)
2: $\bar{C} \leftarrow l$
3: $(z,w) \leftarrow$ CHECKCENTER$(l, \bar{C})$
4: **if** $(z,w) == (\infty,\infty)$ **then return** $[(z,w), \emptyset, 0]$
5: $[\hat{C}, \bar{C}] \leftarrow$ UPDATELOCATIONS$(\hat{C}, \bar{C}, (z,w))$
6: $r_{z,w} \leftarrow max_{l' \in \bar{C}} \delta(l', (z,w))$
7: **while** $\hat{C} \neq \emptyset$ **do**
8: **Let** $l : \delta(l,(z,w)) \leqslant \delta(l',(z,w)) \forall l' \in \hat{C}$
9: $\hat{C} \leftarrow \hat{C} \setminus l$
10: $[(x,y), r_{x,y}] \leftarrow$ 1-CENTER$(\bar{C} \cup l)$
11: **if** $r_{x,y} \leqslant d_r$ **then**
12: $(x',y') \leftarrow$ CHECKCENTER$((x,y), \bar{C} \cup l)$
13: **if** $(x',y') == (\infty,\infty)$ **then**
14: **return** $[(z,w), \bar{C}, r_{z,w}]$
15: $(z,w) \leftarrow (x',y')$
16: $[\hat{C}, \bar{C}] \leftarrow$ UPDATELOCATIONS$(\hat{C}, \bar{C}, (z,w))$
17: $r_{z,w} \leftarrow max_{l' \in \bar{C}} \delta(l', (z,w))$
18: **else**
19: **return**$[(z,w), \bar{C}, r_{z,w}]$
20: **return**$[(z,w), \bar{C}, r_{z,w}]$

if exists, an admissible solution in the admissible region delimited by the constraints in line 3 and out of the not admissible area Z (line 4), which is defined in terms of other $\hat{z}$ constraints. Hence these constraints are, in the worst case less than $n + \hat{z}$ (but it can be expected that they are much less in general), and they give rise to a non-convex admissible region, delimited by elliptic curves on a 2-dimensional plane. A simple way to find an admissible point can be to test for the admissibility of each single point of intersection between each pair of constraints, which would lead to an upper bound of $O(n+\hat{z})^3$ to its complexity. Hence, an upper bound to the complexity of the algorithm is $O(n^2 * (n+\hat{z})^3)$.

Algorithm 3 updateLocations

function UPDATELOCATIONS($\hat{C}, \bar{C}, (x, y)$)
  $\bar{C} \leftarrow \bar{C} \cup \{l \in \hat{C} : \delta(l, (x, y)) \leqslant d_r\}$
  $\hat{C} \leftarrow \{l \in \hat{C} : d_r < \delta(l, (x, y)) \leqslant 2 * d_r\}$
  return $[\hat{C}, \bar{C}]$

Algorithm 4 checkCenter

function CHECKCENTER($(x, y), \bar{C}$)
  if $(x, y) \notin Z$ **then return** (x, y)
  Let $D_h = \{(x', y') : \delta(l_h, (x', y')) \leqslant d_r\} \ \ \forall l_h \in \bar{C}$
  $I \leftarrow \cap_{l_h \in \bar{C}}(D_h) \setminus Z$
  if $I == \emptyset$ **then return** (∞, ∞)
  take any point (p, q) *in* I
  return (p, q)

4. Experimental Data Set Preparation

Our experiments are based on a real-world experimental mobility data set, namely GeoLife [42, 43]. The data set is an initiative of Microsoft Research Asia, it includes 182 participants, and it spans a very extended time frame: from April 2007 to August 2012. The data set reports the GPS positions of the participants obtained with GPS trackers or, in some cases, with the personal user's devices. Only a limited number of trajectories are labelled by the used transportation mean, e.g. car, bus, metro etc. The data set collects mainly GPS locations from the Beijing area and it is publicly available and widely adopted in the current literature. We show in Figure 4 a graphical representation of the data set, the figure clearly shows the most visited locations centered in the Beijing area.

Fig. 4. Aggregated representation of GeoLife restricted to Beijing area.

In the reaming in this section, we analyze some of the mobility features of GeoLife and we describe our generative method to extend the amount of the users' trajectories, namely the GeoLife+ data set. The mobility features are computed with the scikit-mobility python library [44].

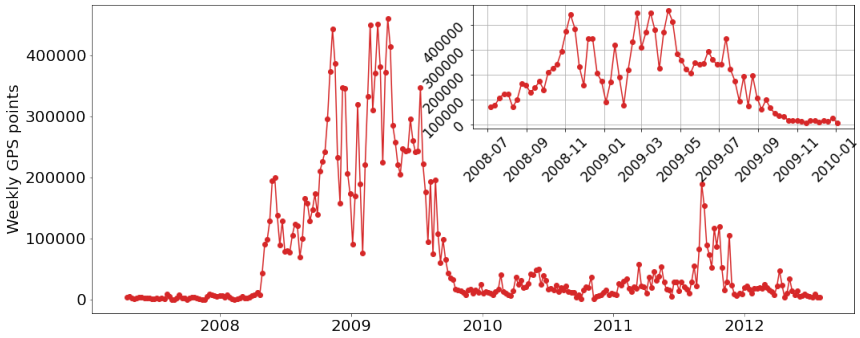


Fig. 5. GeoLife's weekly aggregated GPS points. The inset in the graph shows the aggregated GPS points on a restricted period.

4.1. Analysis of the Mobility Features

The GeoLife data set reports the GPS points in an extended time frame and in a wide geographical area. We firstly cropped the locations to the Beijing city[1], and we also restricted the considered time frame. We report in Figure 5 the time series showing the variation of GPS points aggregated on a weekly basis, from April 2007 to August 2012, the inset in the figure shows the number of GPS points in the sub-period July 2008 to December 2009 with the highest number of GPS points. The original data set comprises 14.696.563 points, 106 users and a total of 11376 trajectories with an average of 107 trajectories per user.

4.2. Data Augmentation Through a Generative Method

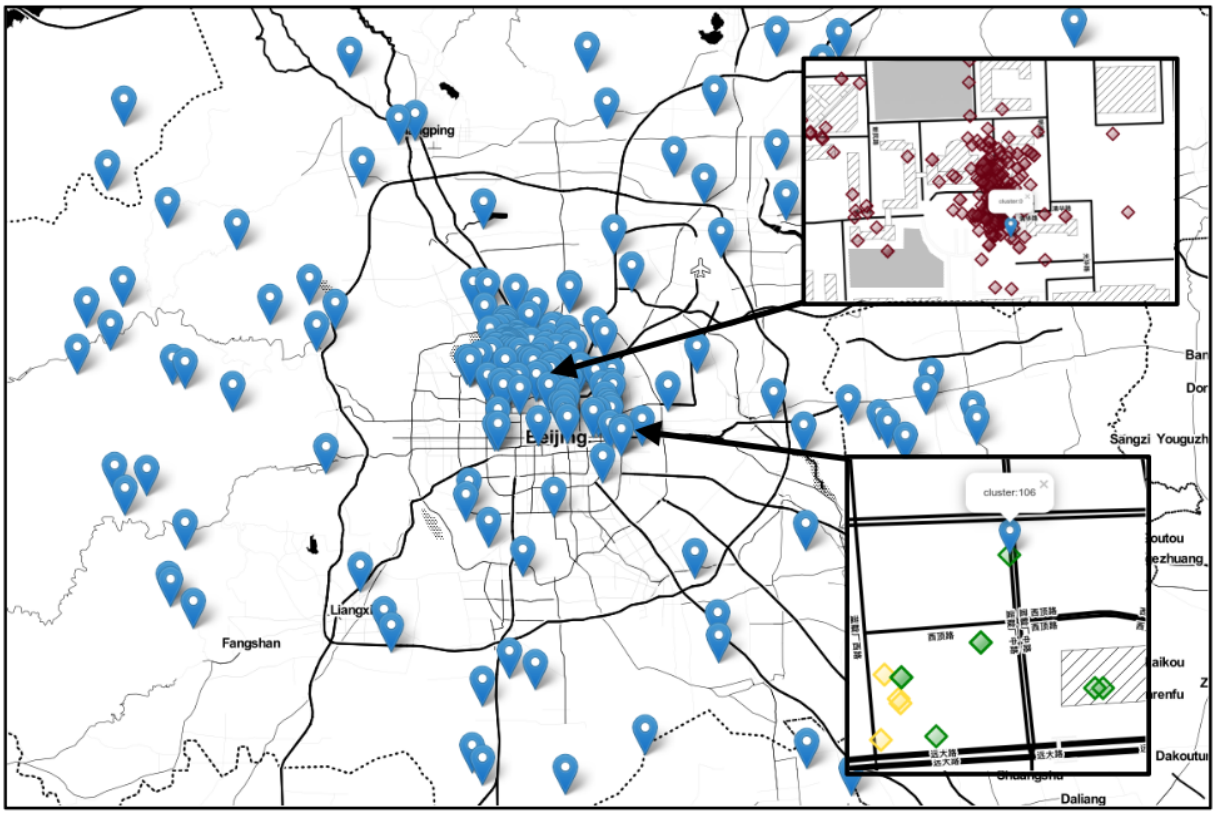
GeoLife provides a valid reference data set for analysing the user mobility. However, for the purpose of this work, we required to further extend the amount of GPS points in order to simulate crowded locations and, in turn, to build a coverage map. To this purpose, we designed a simple generative method to generate users' trajectories. Our generator allows to define the number of users, the type of mobility (pedestrian, by car, by bike), and the number of trajectories each user will follow. Moreover, the source and destination of each trajectory is not based on a stochastic process, rather it follows these steps:

- identifying the users' stop places of the original GeoLife data set:
- clustering the stop places;
- computing the cluster's medoids and extending with extra points.

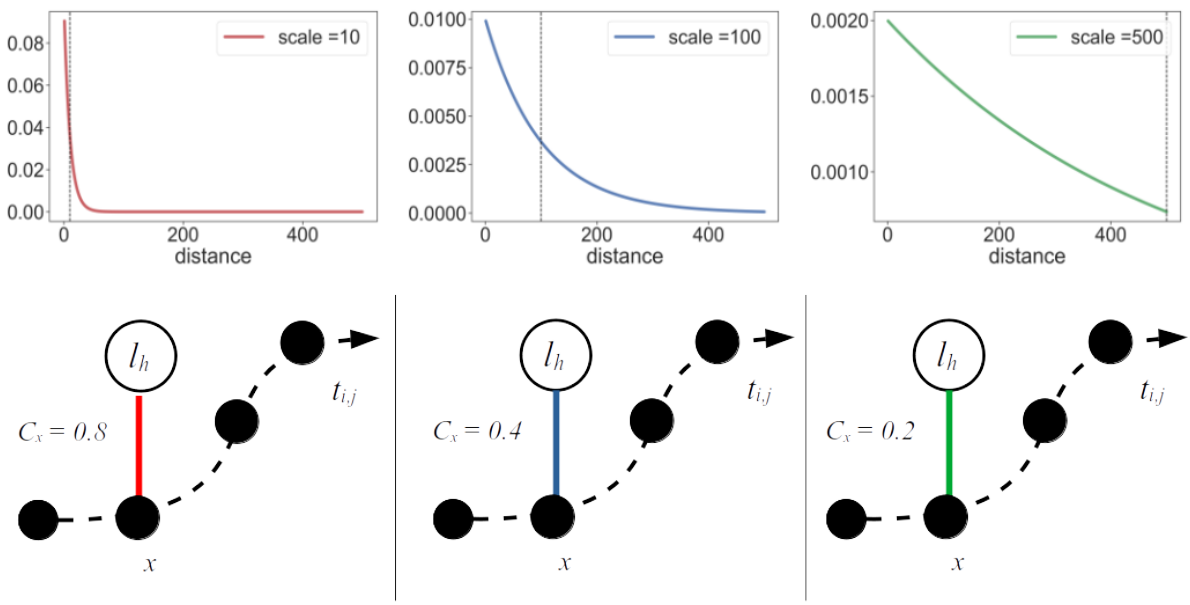
The first step consists in detecting the places where users stop for a while. To this purpose, we detect the stop places with the method described in [45] and implemented by [44]. We identify stop places where users stop for 60 minutes in a range of 100 meters. The second step consists of clustering the stop places, so that to group contiguous stop places. The clustering is implemented with the DBSCAN [46] algorithm configured with a minimum set of 4 points in a radius of $\epsilon = 200$ meters. The last step consists of finding the cluster's medoid, namely the closest stop place to the cluster's centroid. This last step allows to obtain a valid representative stop place for each of the clusters without incurring in invalid locations. Figure 6a reports the medoids obtained from the clustered stop places. The blue markers show the medoids, and the two insets in Figure 6a show the stop places (as colored diamonds) and the corresponding cluster medoids.

The set of medoids M we previously computed, represents our preferential sources and destinations of the augmented user's trajectories. The strategy we implemented generates, for every user, y trajectories in the time period July 2007 - December 2009 from a source $s \in M$ to a destination $d \in M$ with a given speed v (according to the mobility profile, e.g. $v = 1.8m/s$ for pedestrians). The path $s \rightarrow d$ is computed following the shortest path, by using only admissible routes. This last filter prevents a trajectory to cross rivers, buildings or traversing locations

[1] `min lat = 39.54, max lat = 40.3, min lon = 115.75, max lon = 117.13`, EPSG 4326 reference system



(a) Medoids identified by clustering the stop-places (Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org).



(b) Effect of the scale parameter to the coverage value.

Fig. 6. Medoids and effect of the scale parameter.

Table 2
Setting of the Generative Model (*sp* stands for stop place).

sp duration	60’
sp radius	100*m*
sp eps	200*m*
min *sp* cluster	4
interpolation distance	200*m*
augmented users	100
user’s trajectories	50
mobility profile	pedestrians

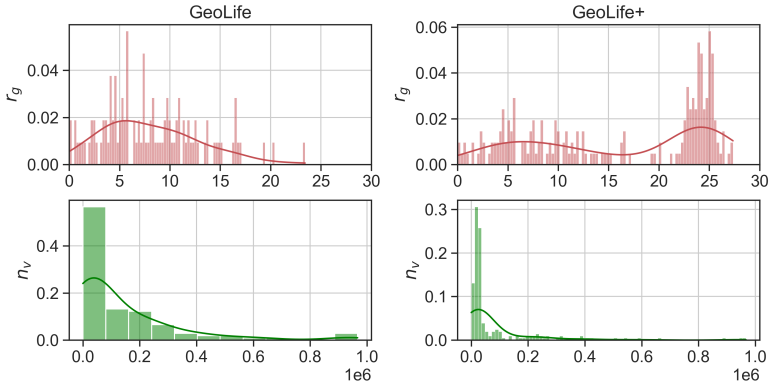


Fig. 7. Comparison of the distribution of the radius of gyration (r_g) and the number of visits (n_v) for GeoLife and GeoLife+.

not connected to a valid urban path. Moreover, the shortest path is computed according to the mobility profile. More specifically, pedestrians are free to use walking routes, while bikes can also ride streets or bike lanes. Each augmented trajectory is in the form of an ordered sequence of GPS points from s to d. We re-sampled the GPS points for every trajectory so that to obtain GPS points evenly distributed from s to the d. In particular, we use a linear interpolation generating a GPS point every 200 meters. It is worth to notice that the settings we used for the generative method are selected according to the objectives of this work. Our goal is not to characterize/predict the urban mobility, rather to test the coverage model and the UAV deployment strategy with a dense mobility data set. Table 2 summarizes the generative model settings.

The GeoLife+ data set is obtained by generating 100 users each with 50 extra trajectories roaming in the area bounded in Figure 4. Table 3 compares some of the mobility features of GeoLife and GeoLife+ data sets. We also analyze two mobility features, namely the distribution of the radius of gyration r_g and the number of visits n_v. The radius of gyration characterizes the typical distance travelled by a set of users, the higher the radius, the far distant users travel. The number of visits defines the number of locations visited by the users. Concerning the distribution of r_g it is possible to observe a shift of the typical distance travelled. With the GeoLife data set, the distribution peaks around 5 *km*, while with GeoLife+ it shifts to 25 *km* with a median of 7.47 *km* for GeoLife and 18.18 *km* for GeoLife+. Concerning the number of visits n_v, the distribution tends shrinking in GeoLife+. The median number of visited locations of GeoLife’s users is of 56.406 while for GeoLife+’s users is of 24.524, the reduction is due to the increase of users with the GeoLife+ data set.

5. Analysis of the Coverage Model and Evaluation

In this section, we detail how we execute the proposed coverage model on a real-world dataset. Section 5.1 describes how we set-up the model, Section 5.2 reports detailed information about the experimental dataset. In

Table 3
Comparison between GeoLife and GeoLife+ data sets

Parameter	GeoLife	GeoLife+
#points	14.696.563	17.042.546
#users	106	206
#trajectories	11376	16370
#avg traj	107	79
r_g (50th)	7.47	18.18
n_v (50th)	56.406	24.524

particular, we describe the time periods used and the obtained coverage map. Finally, in Section 5.3 we discuss the impact of the mobility dataset to the computation of the coverage model.

5.1. Set-up of the Coverage Model

The proposed UAV deployment algorithm is based on the coverage model described in Section 3.1. We analyze in this section how we configure and apply the coverage model to the GeoLife+ data set.

We briefly recall from Section 3.1 the random variable X_i^h modelling the events: user $u_i \in U$ accepts a detour towards location $l_h \in L$ up to a distance $t_{i,j}^h \in \mathbb{R}^+$ (we recall that $t_{i,j}^h$ is defined as the minimum distance between the trajectory $t_{i,j}$ and the location l_h). Following an intuitive observation, the closer a user to a location, the higher the probability she/he is willing to accept a detour towards such location. Such observation is well reflected with an exponential distribution, whose probability density function is in the form:

$$f_{X_i^h}(t) = \lambda * \exp(-\lambda * t) \tag{8}$$

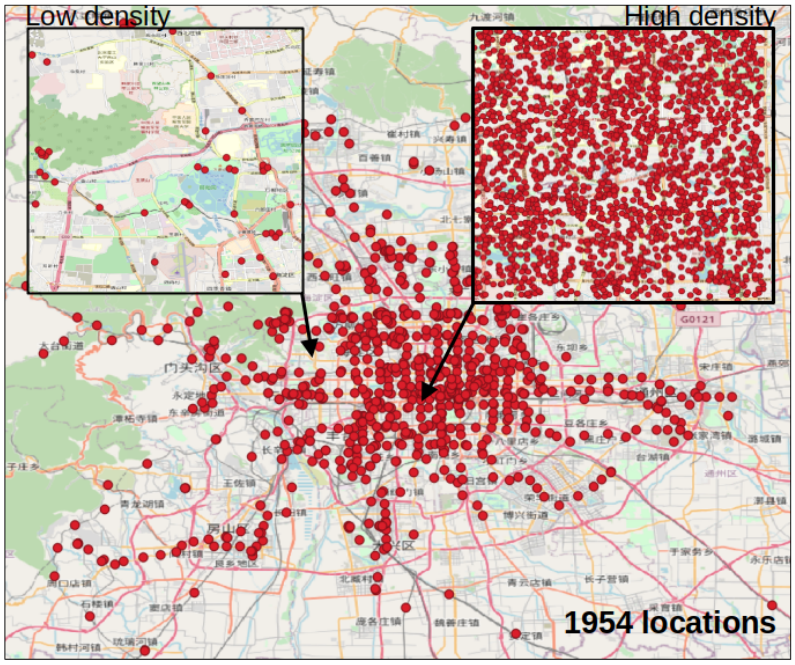
The distribution's scale is $1/\lambda$ and $t \in [0, \mathbb{R}^+]$. The scale parameter plays a significant role in determining the spread of the exponential distribution. In our specific situation, this parameter represents the distance at which users are willing to tolerate a deviation from their preferred route. To clarify, when the scale is smaller, it implies that a detour is acceptable to users only if it's close to their intended location. Conversely, when the scale is larger, it indicates that users are more open to accepting detours even when they are a considerable distance away from their desired destination. We implemented Equation 8 as a python process using the scipy library. In particular, we set $1/\lambda = 100$.

We report in Figure 6b, three representative examples of an exponential distribution set with scale values in the range $[10, 100, 500]$ meters. Setting the exponential with a small value of scale (e.g. 10 meters) does not prevent for a user to accept a detour from 500 meters. Rather, the probability of such event will be smaller than that of an exponential distribution configured with a higher scale value.

The coverage value at distance $t_{i,j}^h$ to l_h decreases as the scale increases, hence the detour probability of a user following trajectory $t_{i,j}$ varies with the parameter $1/\lambda$. As a result, the coverage probability defined in Equation 6 can be re-written in closed form as:

$$P(W^h) = 1 - \prod_{\forall u_i \in U} \Big(\prod_{\forall t_{i,j} \in T_i} (1 - \exp(-\lambda * t_{i,j}^h)) \Big) \tag{9}$$

The set of locations of interest, L (see Section 3.1) is obtained by extracting points of interest from the Beijing area that, generally, are visited by the GeoLife+ users. To this purpose, we extract from OpenStreetMap a collection of places labelled with the following tags: `square`, `monument`, `mall`, `station` and `bus`. We also extend such places with a set of random points in the city center. As a result, we obtain 1954 locations of interest and we compute, for each of them, the coverage probability with Equation 6. The selected locations and the corresponding coverage map are reported in Figures 8a and 8b, respectively. The two figures show two insets: the high density (HD) and the low density (LD) regions. These are defined as bounding box of 8.5 $km \times 11.1$ km. The HD region



(a) The selected locations of interest.

(b) The coverage map computed over 1954 locations on the Beijing area with the GeoLife+ data set.

Fig. 8. The coverage probability model in action (Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org).

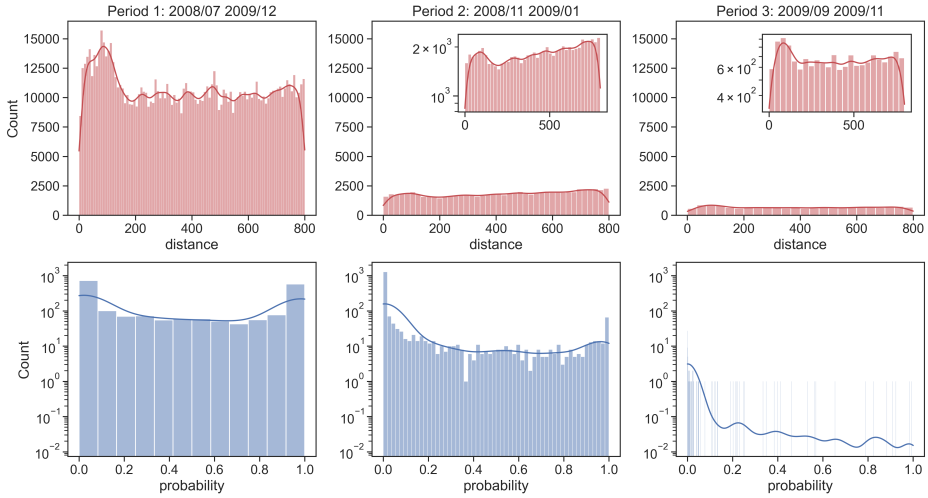


Fig. 9. Distribution of the distances and of the coverage probability.

includes $|L| = 1121$ locations of interests, while the LD one includes $|L| = 44$ locations. Recalling the objective function of our UAV deployment strategy (see Equation 7), we set the exponential scale value to $1/\lambda = 10$.

5.2. Experimental Dataset preparation

We compute the coverage probability for each of the locations by selecting different time periods of the GeoLife+ data set (refer to Figure 5 for a visual representation of the amount of GPS points):

- Period 1: 18 months (July 2008 to December 2009), 17.042.546 GPS points;
- Period 2: 3 months with high mobility (November 2008 to January 2009), 4.469.833 GPS points;
- Period 3: 3 months with low mobility (September 2009 to November 2009), 814.527 GPS points.

Moreover, in order to speed up the computation process, we filter out trajectory's points too far the locations of interest. To this end, we set a detour radius of 800 *m*.

We now analyze the distribution of the obtained distances from the locations of interest and the distribution of the coverage probability for 3 time periods, as shown in Figure 9. Concerning the distribution of the distances, the histograms report the minimum Haversine distance between each location of interest and each of the user's trajectories. As expected, the distances are bound in the interval [0 - 800] *m*, as the detour radius is set to 800 *m*. Moreover, the number of GPS points for Period 1 is far higher that that of Periods 2 and 3. We observe that the distance distribution for Periods 1 and 3 peak to smaller distance values with respect to Period 2, as shown with the two insets in Figure 9 plotted on a log-y scale.

Concerning the coverage probability, we report the log-y distribution for the 3 periods. Such values are obtained by executing the coverage model defined by Equation 6. The obtained probabilities for Periods 1 and 2 show the following trend: the majority of the locations are not covered (coverage probability = 0) or they are fully covered (coverage probability = 1) by user's trajectories. This trend is clearly shown by the $[0, 1]$ bins in Figure 9. Period 3 is characterized by a lower number of GPS points with respect to the other periods (814.527), and this affects the distribution of distances as well all the coverage probability of the locations of interests. From Figure 9 we observe that locations with Period 3 are scarcely covered, with very low probability values.

5.3. Impact of the Mobility to the Coverage Model

The coverage model proposed in Section 3.1 is strictly affected by the mobility dataset used for the experiments, as the coverage probability of a set of locations of interest also depends on the past mobility traces. More specifically, three aspects of the human mobility might affect the results of the coverage model:

Fig. 10. Evolution of the coverage map on Period 2 and Period 3.

- the considered time period;
- the demographic of the dataset;
- the existence of events strongly modifying the mobility patterns.

Concerning the time period, it determines high/low covered regions, e.g. regions visited during summer time might differ significantly with respect to regions visited during the winter season. Similarly, but on a different time scale, the coverage probability of some regions might differ according to the daily hours, e.g. the regions visited during working hours differ with respect to off-time hours. The duration of the dataset used to compute the coverage, strictly depends on the objectives of the CrowdSensing monitoring campaign. More specifically, if the goal is to collect data on the long period, without stringent temporal constraints, then it is reasonable to consider a dataset spanning for a long duration so that to obtain a stable coverage map of the considered regions. Differently, if the goal is to retrieve data from regions rapidly changing, then the duration of the dataset used for building the coverage map should also reproduce the mobility for a short period. In order to clarify this aspect, we consider a scenario in which our goal is to collect data from a set of regions on a long period, e.g. 3 months. We compare in Figure 10 two coverage maps obtained by considering Period 2 and Period 3 described in Section 5.2 (November 2008 to January 2009 and September 2099 to November 2009). In the case depicted in the figure it is seen that at a micro scale there are clearly differences in the coverage, although at a macro scale there are (macro) areas whose coverage remains stable. and September 2008 to November 2009). The maps show some differences, but as a general trend the regions with high coverage values remain stable. In addition to the time period, also the demographic of the dataset represents a distinguishing feature which affects the coverage computation. As a significant example, a mobility dataset based on students from the high schools tend to visit different regions with respect to employees daily commuting to work. Finally, if the dataset captures social events (such as concerts, gatherings, or crowds) then the resulting coverage values will be also affected by these events situation modifying the existing mobility patterns.

6. UAV Deployment Strategy: Settings and Results

We analyze the performance of `StationPositioning` by varying three main settings and by comparing the results against three benchmark algorithms. In particular, we modify:

- the GeoLife+ time periods;
- the geographical regions;
- the percentage of not admissible areas;

For what concerns the periods and the regions, Section 5 reports the settings we applied. Concerning the not admissible areas, we progressively reduce the allowed deployment areas of the HD region with a number of circles, we refer to them as the not admissible areas. The number of circles is obtained as a percentage of the total area of the HD region. In particular, the total HD area is of 94.885.022,25 m^2 and we exclude: 5% (default), 10%, 30% and 60%. The not admissible areas are represented as non-overlapping circles of radius 1 km. It is worth to notice that the proposed algorithm (see Algorithm 4) does not require a circular shape for the not admissible areas. Differently, our implementation models such areas as generic polygons.

Table 4
Experimental settings.

Period 1	18 months, 17.042.546 GPS points
Period 2	3 months, 4.469.833 GPS points
Period 3	3 months, 814.527 GPS points
Exponential scale	$1/\lambda = 10$
Detour radius	800 m
Geographical regions	HD (1121), LD (44)
τ coverage threshold	0.85
not admissible areas	[5%,10%,30%,60%]
Radius of not admissible areas	1km
d_r UAV radius	500m
k stations	[1,3,5]

We also assume to deploy a commercial UAV drone flying in circle of radius $d_r = 500m$ from the ground station. We test the deployment for $k = [1, 3, 5]$ UAV stations. Furthermore, recalling the objective function of our UAV deployment strategy (see Equation 7), we set the coverage threshold $\tau = 0.85$. The τ value allows to select only those locations scarcely visited by the MCS users, therefore they represent the target locations of a flock of UAV.

Concerning the benchmark algorithms, we simulate the deployment of 3 other algorithms: Random, DBSCAN, KMeans deployment algorithms, as reported in sub-section 6.1. All the experimental settings are reported in Table 4

6.1. Metrics and Benchmark Algorithms

We study the performance of our deployment strategy by measuring the ratio of the covered locations with respect to the total number of the target locations. More specifically, the ratio measures the amount of locations with coverage probability $p \leqslant \tau$ that can be reached by UAV grounded at k stations. Therefore, the higher the ratio the higher the number of location covered by UAV. We assess the ratio by comparing it to three benchmark algorithms: *random*, *DBSCAN* [46] and *KMeans* [47]. The choice of these benchmark algorithms is based on the principle of utilizing established and widely-recognized clustering techniques. In particular, we details in the next their distinguishing features:

- The Random algorithm serves as a initial benchmark, enabling a basic performance comparison between `StationPositioning` and a stochastic approach. While we acknowledge that Random deployment may lead to the placement of inefficient stations, it remains a valid and plausible approach.
- DBSCAN employs a robust unsupervised clustering strategy. We view it as a valuable and viable alternative for addressing the problem in our study, as it has demonstrated effectiveness in numerous other application scenarios.
- KMeans also utilizes an unsupervised clustering strategy. It partitions data into k clusters by iteratively optimizing cluster centroids, assigning each data point to its nearest centroid. K-Means is well-suited for large datasets but requires the a priori specification of the number of clusters (k).

The Random deployment consists of randomly picking a center (x, y) in a region (HD or LD) and by measuring the ratio of coverable locations with respect to the target locations. In order to obtain stable results, we ran the Random positioning for $R_0 = 10.000$ iterations, to guarantee a statistical confidence of about 95%.

The DBSCAN algorithm deploys a UAV station by clustering the target locations. Clusters are obtained by setting the maximum distance between two samples $eps = d_r$ computed with the Haversine distance and by excluding noisy samples. Then, we pick the top $k \in [1, 3, 5]$ clusters, according to the amount of samples they comprise. Finally, we identify the corresponding centroids as the location for deploying the UAV stations.

Finally, the KMeans algorithm clusters the target locations according to the well-known unsupervised strategy. We simply configure KMeans so that to detect $k \in [1, 3, 5]$ clusters and, similarly to DBSCAN, we identify the cluster's centroids as the location where to deploy the UAV stations.

In order to obtain a fair performance comparison with respect to the proposed algorithm, we extended the Random, DBSCAN and KMeans standard implementations with an optimization that avoids deploying stations overlapping with not admissible areas. Such optimization mimics our Algorithm 4 `checkCenter`.

All the algorithms have been implemented with a custom Python-based simulator. The simulator is logically organized in three main blocks:

- Configuration settings: we can set a list of settings to apply during a simulation run. Examples of such settings include: the start/end period, scale value and detour radius of the exponential distribution, list of locations of interest;
- Coverage analysis: the simulator computes the coverage map by using the list of locations of interest defined with the settings. The coverage map provides a coverage score for every location of interest, as graphically shown in Figure 8b. The coverage map is obtained by using Equation 9.
- Stations Positioning: the simulator takes as input the coverage map and it returns the location of k distinct UAV stations. Each station is placed by executing Algorithm 2.

6.2. Numerical Results

We now analyze the experimental results obtained with the settings previously described. Figure 11 summarizes the overall results for Period 1, from July 2008 to December 2009. The figure reports the results both for the HD and LD regions. We observe that the `StationPositioning` algorithm always obtains higher ratio values than the benchmark algorithms for $k \in [1, 3, 5]$. However, we note that such values vary remarkably between the HD and LD regions. From the figure, we observe that the algorithms tested with the HD region always return lower ratio values than the LD ones. In fact, the HD region comprises a higher number of locations of interest with respect to the LD (1121 vs 44 locations). As a result, the amount of *coverable* locations in the HD region with any value of k will always result lower than the LD region.

`StationPositioning` deploys up to $k = 5$ stations in the HD region covering approximately 10% of all the target locations (1121). Moreover, the software optimization described at line 6 of Algorithm 1 reduces the number of iterations down to 80%. Differently, the benchmark algorithm configured with $k = 5$ obtain lower ratio values. In particular, Random covers 3%, DBSCSAN 1.7% and KMeans 1%. The inefficiency of DBSCAN and KMeans depends from the fact that such algorithms deploy the stations by only considering the cluster's centroids of the top k clusters. In turn, the identified centroids might be located too far from the target locations, hence not reachable by any UAV. Differently, `StationPositioning` considers each of the target locations and it progressively shifts the center of the station so that to maximize the ratio, as described with the procedure in Algorithm `DeployStation`.

We show in Figure 12a a visual representation of $k = 5$ stations deployed with the 4 algorithm. The heatamp shows the coverage probability of the target locations in the HD region: the yellow areas denote set of locations with high-coverage probability, while the green areas indicate locations with low probability values. Except the Random algorithm whose deployment if fully randomized, the remaining algorithms aim to deploy stations in the green areas. The grey circle in Figure 12a shows the not admissible area (it covers the 5% of the total area), while the red marks report the UAV stations. It is worth to notice that the KMeans algorithm can deploy up to 4 out of the 5 stations. In particular, KMeasn detects $k = 5$ clusters, it checks if the cluster's centroids do not overlap with the not admissible areas and, eventually, it deploys the stations. In our experiment, 1 out of 5 centroids overlaps with the not admissible area, hence only 4 stations are actually deployed.

We then analyse the impact of the selected time periods. To this purpose, we test the 4 algorithms with Period 2 and Period 3 spanning respectively from November 2008 to January 2009, and from September 2009 to December 2009. As described in Section 5, the 2 periods cover a 3-month-length time window of GeoLife+. The 2 sub-sets are characterized by a lower number of GPS points with respect to Period 1, leading to a different distribution of the coverage probability, as shown in Figure 9. When we filter out locations of interest with a coverage probability lower or equal than τ, we obtain from periods 2 and 3 a high number of low-covered locations. The final result is

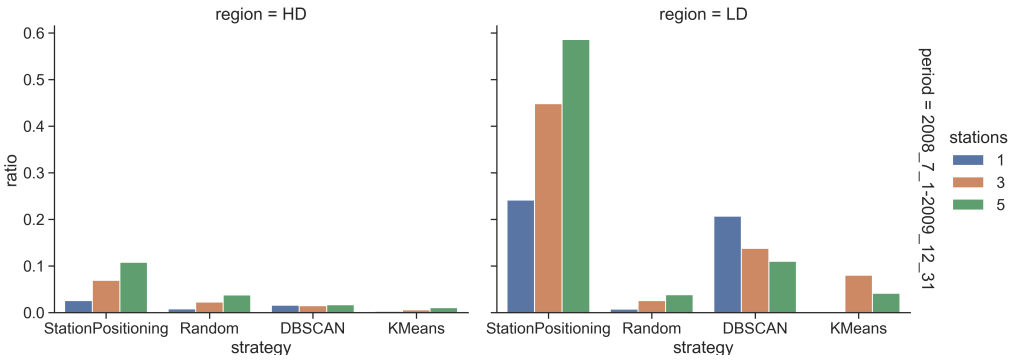
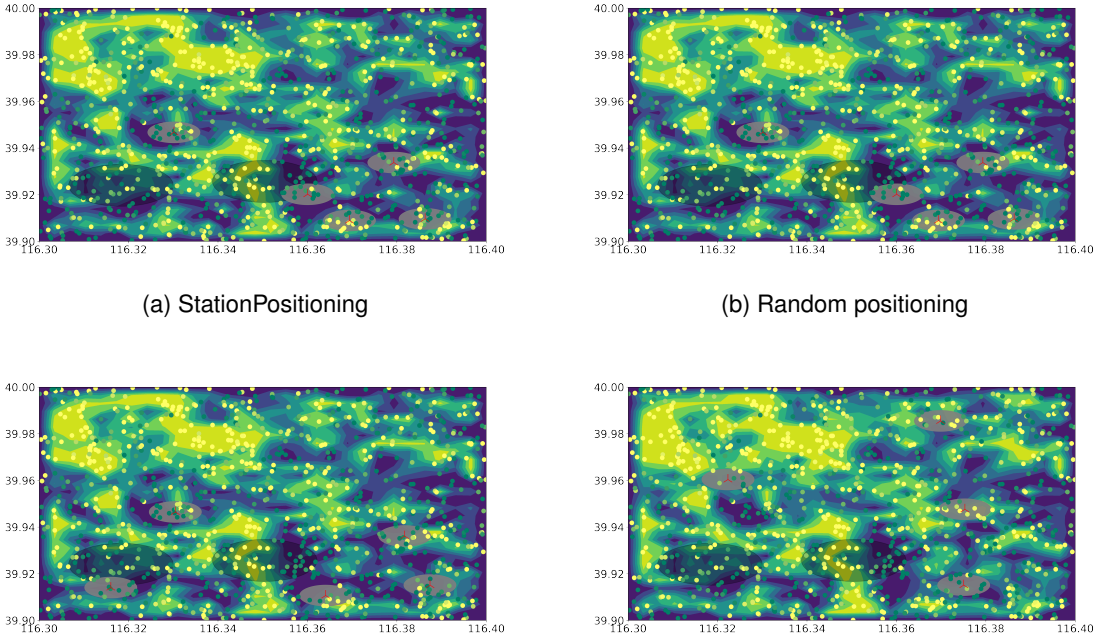


Fig. 11. Performance of the 4 deployment strategies with period 1 in 2 regions (HD, LD).



(a) StationPositioning

(b) Random positioning

(c) DBSCAN positioning

(d) KMeans positioning

Fig. 12. Results of station deployment with 4 algorithms.

that the deployment algorithms maximize the amount of low-covered locations in an always more-dense scenario, but the ratio value cannot be further increased for Periods 2 and 3 as the algorithms already find the $k = 5$ optimal spots for the UAV stations. We show in Figure 13 the obtained results that are comparable to performance of Period 1.

We finally focus on the impact of the not admissible areas to the performance of `StationPositioning`. In particular, we analyze the results by varying the percentage of the areas where it is not possible to deploy any station. Our goal is to stress `StationPositioning` and to verify if it re-deploys stations to near-optimal locations. We briefly recall the settings adopted for the selection of the not admissible areas:

– circular shape of radius 1 *km*;

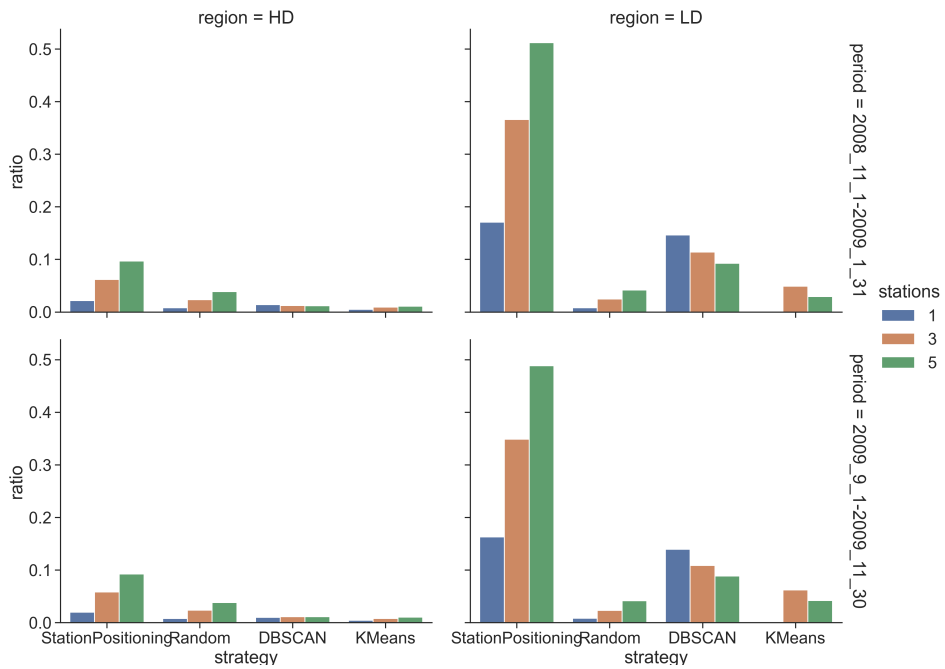


Fig. 13. Performance of the 4 deployment strategies with periods 2, 3 and 2 regions (HD, LD).

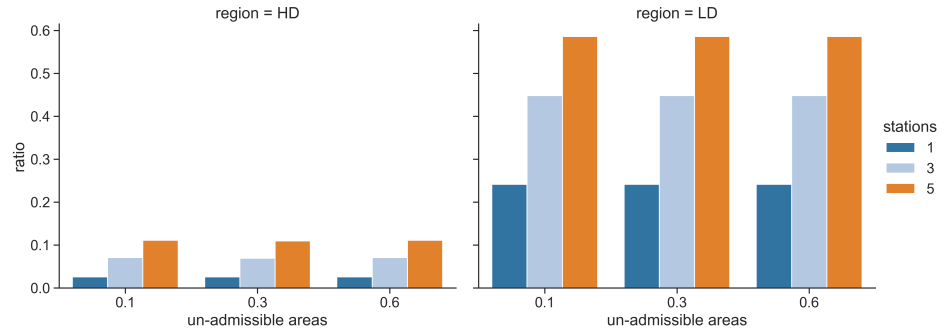


Fig. 14. Performance of `StationPositioning` with an increasing number of not admissible areas.

– the number of not admissible areas is 5% (default setting), 10%, 30% and 60% of the total area of the HD region.

Figure 14 shows the obtained results for Period 1. Differently from what initially expected, the performance of `StationPositioning` do not degrade with the increasing percentage. We observe a stable trend of the ratio, suggesting us that the algorithm does not violate the not admissible area constraints and it can still find high-performing spots for the UAV stations. The obtained results are motivated by 2 optimizations:

– Algorithm 4 `checkCenter`: this procedure verifies if the candidate station located in (x, y) overlaps with the intersection of all the not admissible area and, eventually, it recomputes a valid alternative;
– Algorithm 2 `DeployStation`: once the stations (x, y) is selected, the algorithm draws a minimum circle of radius d_r and the algorithm shifts its final position so that to include the highest number of locations with low coverage value.

In summary, from the obtained results we can infer the following considerations:

– As compared to an exhaustive approach, the StationPositioning algorithm reduces of around 80% the number iterations required to converge to the optimal solution;
– At an equal number of stations deployed, our approach StationPositioning outperforms *conventional* approaches based on KMeans or DBSCAN to position the UAV base station, in terms of coverage of target locations, as shown in Figure 11;
– Differently than DBSCAN or KMeans-based approaches, the StationPositioning algorithm takes into account also areas that are not admissible for the deployment of the UAV base station.

7. Discussion

Beyond the technical aspects, the advantages of our methodology consist of the use of UAVs that complement the coverage given by users of a CrowdSensing platform. Concerning the time of execution of the algorithm, we have evaluated its complexity, which is polynomial in the number of locations to cover and in the number of non-admissible areas. However, in practical settings, the placement of a UAV base station requires a physical deployment and it is not expected to be moved frequently, hence also the algorithm will be executed only sporadically. For this reason the algorithm execution time, which is polynomial anyway, is less relevant than its ability to find an optimal location for the deployment

At the same time however, there are limitations concerning the need to know in advance the system parameters, in particular those areas not suitable to deploy an UAV station, and possible further limitations due to the specific regulations for the flight of UAVs in some areas.

The strategy we propose in this work is based on a data-driven approach. In particular, we analyze the mobility of a representative data set, we implement the coverage model, we spot low-covered locations and, finally, we implement an optimizer to locate the UAV stations. The current limitations of the proposed model can be summarized as follows:

– The accuracy of the coverage map is influenced by the number of users participating in a CrowdSensing data collection campaign. Specifically, as the number of users and tracked trajectories increases, the resulting coverage map becomes more accurate.
– As the CrowdSensing paradigm necessitates user contributions to data collection from their own devices (e.g., smartphones), this responsibility may accelerate battery depletion, potentially diminishing user participation in CrowdSensing initiatives.

In the following we summarize some factors influencing the proposed data coverage model:

– user's mobility: the coverage map resulting from our model is correlated with GPS traces followed by users. The way people move, directly influences the resulting coverage map;
– detour probability: in the proposed model, we adopt a truncated exponential distribution modelling the probability that a user accepts detouring toward a location at a given distance. Therefore, such distribution affects the computation of the coverage map, according to principle that: the closer the location to a user, the higher the detour probability toward such location;
– detour radius: the UAV's travelling distance affects the performance of StationPositioning. In particular, the radius directly determines the width of the circle used to bound the coverage area of an UAV.

We consider that such pipeline can be improved with further lines of investigation. On the one hand, we are interested in adopting a predictive analysis to anticipate the coverage probability of the target locations. Predictability can be obtained with several state-of-the-art methodologies such as the use of regressors or the use of RNN (Recurrent Neural Network). Such tools can be trained to predict the coverage probability as time progresses. In particular, LSTM and GRU layers are powerful tools that might increase the performance of a mobile CrowdSensing campaign, as they have the potentiality of detecting changes of the coverage along with the time. On the other hand, the Reinforcement Learning paradigm can also provide a boost of the performance. In this last case, the idea is to implement a try-and-error approach through which deploy stations according to the coverage value (as done in this work), but to also measure the reward. More specifically, the larger is the number of locations that are covered with

a specific deployment, the higher is the reward assigned to the stations. Similarly, the smaller is the number of locations that are covered from a specific deployment, the lower is the reward assigned. Such a *simple* pattern allows to explore the space of the possible deployments and, eventually, to reach quickly the optimum. Concerning the UAV deployment problem, geometric approaches can also be adopted to partition the regions in different sub-areas and assign to each of them a UAV fleet grounded on a specific base station, as also discussed in [48]. We also consider a promising approach to further refine the coverage model proposed in this work. We followed an approach according to which the coverage value for a region is obtained by considering the detour probability toward such region. A reward could also be considered in our model to modify the willingness to accept/decline such detour. Under this respect, a coverage model also considering the attitude of users might further refine the coverage computation.

Concerning the optimization strategy to locate the positions of the UAV stations, our approach is essentially based on a greedy approach in which we place one station after the other without rolling back on the previous decisions. This method demonstrates scalability concerning the quantity of GPS points within trajectories. We tested its effectiveness across various time spans, ranging from 3 to 18 months, and with a varying number of GPS points, spanning from 814K to 17 million points. The rationale behind this approach lies in the selection of UAV stations based on a probabilistic coverage map. This map is constructed considering the anticipated coverage of future points and, consequently, contains inherently imprecise information. However, we are also investigating other optimization approaches that operate on all stations at once. This however would imply not only a change in the optimization algorithm, but also the understanding of how to deal with the deterministic optimization algorithm over imprecise input data, in absence of ground truth associated to the mobility traces. This would also require defining suitable metrics to assess the solutions obtained in this way, and to evaluate the actual cost/benefits of more complex optimization strategies in this framework. A final consideration refers to the UAV flying model. It is worth to mention that in this work we adopted a simple approach as also done in [23]. Future lines of investigations can also consider adopting realistic flying dynamics, considering not only wind direction and speed but also air traffic and maximum payload for a commercial UAV.

8. Conclusions

In this work we have addressed hybrid architectures combining mobile CrowdSensing and unmanned aerial vehicles (UAV). Despite most of the previous works, that have considered the two technologies alone, or that have considered UAV as a replacement for mobile CrowdSensing, we believe that mobile CrowdSensing and UAV are actually to complementary technologies that may provide advantages when used in combination. The adoption of UAVs for Mobile CrowdSensing applications presents some implications along three dimension: 1) safety: UAVs can be employed to collect data but some safety measures require to be considered as: managing un-admissible flying areas, keep line of sight between the UAV and the base station, manage the flying altitude of UAV in urban areas; 2) privacy: UAVs adoption require to avoid overflying sensitive areas e.g. military areas or naturalistic areas subject to specific requirements; 3) ethic: we consider it as an important aspect whose implications can be consider only when the technology will be largely diffused.

We tested our deployment algorithm with the GeoLife+ data set, and we analyzed the performance against 3 benchmark algorithm and several experimental settings. We tested the robustness of our algorithm in re-deploying UAV stations in regions with an increasing number of not admissible areas. Our experimental results show stable performance, suggesting us a promising integration of UAV with the mobile CrowdSensing paradigm.

In particular, mobile CrowdSensing is pervasive and may involve a very large number of sensing elements carried daily by their (human) owners, but, at the same time, it depends on the actual human mobility that is not homogeneous and, as a consequence, results in a non-uniform coverage of the space. In order to mitigate this fact, mobile CrowdSensing platforms include the possibility of injecting specific tasks to their participants (e.g. requesting to move to a specific location to sense/take data from there), however this possibility depends on the actual availability of the participants to deviate from their trajectories and it usually requires strong incentives. On the other hand, UAV are controllable and may be directed to cover areas and location that are not sufficiently covered by the mobile CrowdSensing platform, however, they are subject to constraints (such as the autonomy in the flight) that limit their operative range. We thus expect an hybrid architecture in which the mobile CrowdSensing platform is

complemented by a number of UAV grounded in a number of stations deployed in a region of interest, from where the UAV can initiate missions to provide sensing coverage to specific locations within an operative range around each station. In this context, the selection of the location of each station plays a strategic role (deploying a station in an area already well covered by the CrowdSensing is not useful). On the other hand, the nature of CrowdSensing that depends on human mobility may change over time the coverage of the monitored region. For this reason we proposed a probabilistic model for the coverage of a region provided by CrowdSensing, and we then exploit the resulting coverage map as a base for an algorithm that identifies the position of a number of stations so to maximize the overall coverage. At this stage we do not consider a UAV flying model, as our primary goal is to evaluate the benefits of a UAV base station deployed according to the output of the proposed coverage model. We are also working to augment the dataset with synthetic, realistic traces so to assess the scalability of our approach also on the number of users, and beyond the scalability on the number of GPS points. Another interesting future extension concerns the introduction of constraints on the number of UAV in the deployment model.

References

[1] K. Abualsaud, T.M. Elfouly, T. Khattab, E. Yaacoub, L.S. Ismail, M.H. Ahmed and M. Guizani, A Survey on Mobile Crowd-Sensing and Its Applications in the IoT Era, *IEEE Access* **7** (2019), 3855–3881. doi:10.1109/ACCESS.2018.2885918.

[2] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich and P. Bouvry, A Survey on Mobile Crowdsensing Systems: Challenges, Solutions, and Opportunities, *IEEE Communications Surveys Tutorials* **21**(3) (2019), 2419–2465. doi:10.1109/COMST.2019.2914030.

[3] M. Girolami, D. Belli, S. Chessa and L. Foschini, How Mobility and Sociality Reshape the Context: A Decade of Experience in Mobile CrowdSensing, *Sensors* **21**(19) (2021). doi:10.3390/s21196397. https://www.mdpi.com/1424-8220/21/19/6397.

[4] L.G. Jaimes, I.J. Vergara-Laurens and A. Raij, A Survey of Incentive Techniques for Mobile Crowd Sensing, *IEEE Internet of Things Journal* **2**(5) (2015), 370–380. doi:10.1109/JIOT.2015.2409151.

[5] H. Jin, L. Su, D. Chen, H. Guo, K. Nahrstedt and J. Xu, Thanos: Incentive Mechanism with Quality Awareness for Mobile Crowd Sensing, *IEEE Transactions on Mobile Computing* **18**(8) (2019), 1951–1964. doi:10.1109/TMC.2018.2868106.

[6] J. Xu, Z. Rao, L. Xu, D. Yang and T. Li, Incentive Mechanism for Multiple Cooperative Tasks with Compatible Users in Mobile Crowd Sensing via Online Communities, *IEEE Transactions on Mobile Computing* **19**(7) (2020), 1618–1633. doi:10.1109/TMC.2019.2911512.

[7] B. Guo, Z. Wang, Z. Yu, Y. Wang, N.Y. Yen, R. Huang and X. Zhou, Mobile Crowd Sensing and Computing: The review of an emerging human-powered sensing paradigm, *ACM Computing Surveys* **48**(1) (2015).

[8] J. Chen, J. Li, S. He, Y. Sun and H. Chen, Energy-Efficient Coverage Based on Probabilistic Sensing Model in Wireless Sensor Networks, *IEEE Communications Letters* **14**(9) (2010), 833–835. doi:10.1109/LCOMM.2010.080210.100770.

[9] M.R. Senouci, A. Mellouk, L. Oukhellou and A. Aissani, An Evidence-Based Sensor Coverage Model, *IEEE Communications Letters* **16**(9) (2012), 1462–1465. doi:10.1109/LCOMM.2012.070512.120999.

[10] X. Yang, Y. Wen, D. Yuan, M. Zhang, H. Zhao and Y. Meng, Coverage Degree-Coverage Model in Wireless Visual Sensor Networks, *IEEE Wireless Communications Letters* **8**(3) (2019), 817–820. doi:10.1109/LWC.2019.2894667.

[11] V. Akbarzadeh, C. Gagne, M. Parizeau, M. Argany and M.A. Mostafavi, Probabilistic Sensing Model for Sensor Placement Optimization Based on Line-of-Sight Coverage, *IEEE Transactions on Instrumentation and Measurement* **62**(2) (2013), 293–303. doi:10.1109/TIM.2012.2214952.

[12] L. Li, D. Shi, X. Zhang, R. Hou, H. Yue, H. Li and M. Pan, Privacy Preserving Participant Recruitment for Coverage Maximization in Location Aware Mobile Crowdsensing, *IEEE Transactions on Mobile Computing* (2021), 1–1. doi:10.1109/TMC.2021.3050147.

[13] J. Duan, J. Lu, W. Jiang and S. Yang, Incentivizing fairness-aware task allocation in mobile crowdsensing with sweep coverage and stability control, *Applied Soft Computing Journal* **97** (2020). doi:10.1016/j.asoc.2020.106781.

[14] M. Girolami, S. Chessa, G. Adami, M. Dragone and L. Foschini, Sensing Interpolation Strategies for a Mobile Crowdsensing Platform, in: *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2017, pp. 102–108. doi:10.1109/MobileCloud.2017.8.

[15] D.G. Reina, H. Tawfik and S.L. Toral, Multi-subpopulation evolutionary algorithms for coverage deployment of UAV-networks, *Ad Hoc Networks* **68** (2018), 16–32, Advances in Wireless Communication and Networking for Cooperating Autonomous Systems. doi:https://doi.org/10.1016/j.adhoc.2017.09.005.

[16] C. Zhang, L. Zhang, L. Zhu, T. Zhangb, Z. Xiao and X.-G. Xia, 3D Deployment of Multiple UAV-Mounted Base Stations for UAV Communications, *IEEE Transactions on Communications* (2021), 1–1. doi:10.1109/TCOMM.2021.3049387.

[17] E. Arribas, V. Mancuso and V. Cholvi, Coverage Optimization with a Dynamic Network of Drone Relays, *IEEE Transactions on Mobile Computing* **19**(10) (2020), 2278–2298. doi:10.1109/TMC.2019.2927335.

[18] X. Zhang and L. Duan, Fast Deployment of UAV Networks for Optimal Wireless Coverage, *IEEE Transactions on Mobile Computing* **18**(3) (2019), 588–601. doi:10.1109/TMC.2018.2840143.

[19] J. Lyu, Y. Zeng, R. Zhang and T.J. Lim, Placement Optimization of UAV-Mounted Mobile Base Stations, *IEEE Communications Letters* **21**(3) (2017), 604–607. doi:10.1109/LCOMM.2016.2633248.

[20] X. Zhong, Y. Guo, N. Li and S. Li, Deployment Optimization of UAV Relays for Collecting Data From Sensors: A Potential Game Approach, *IEEE Access* **7** (2019), 182962–182973. doi:10.1109/ACCESS.2019.2960314.
[21] X. Tao and A.S. Hafid, Trajectory Design in UAV-Aided Mobile Crowdsensing: A Deep Reinforcement Learning Approach, in: *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6. doi:10.1109/ICC42927.2021.9500579.
[22] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez and M. Guizani, When Mobile Crowd Sensing Meets UAV: Energy-Efficient Task Assignment and Route Planning, *IEEE Transactions on Communications* **66**(11) (2018), 5526–5538. doi:10.1109/TCOMM.2018.2857461.
[23] H. Gao, J. Feng, Y. Xiao, B. Zhang and W. Wang, A UAV-assisted Multi-task Allocation Method for Mobile Crowd Sensing, *IEEE Transactions on Mobile Computing* (2022), 1–1. doi:10.1109/TMC.2022.3147871.
[24] C.H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. Wu and K.K. Leung, Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning, *IEEE Transactions on Mobile Computing* **20**(1) (2021), 130–146. doi:10.1109/TMC.2019.2938509.
[25] E. Barka, C.A. Kerrache, N. Lagraa and A. Lakas, Behavior-aware UAV-assisted crowd sensing technique for urban vehicular environments, 2018, pp. 1–7. doi:10.1109/CCNC.2018.8319174.
[26] C. Tao, K. Zhu, B. Chen and Y. Zhao, UAV-Assisted Ground Signal Map Construction based on 3-D Spatial Correlation, 2020. doi:10.1109/GLOBECOM42002.2020.9322507.
[27] D. Belli, S. Chessa, L. Foschini and M. Girolami, A Probabilistic Model for the Deployment of Human-Enabled Edge Computing in Massive Sensing Scenarios, *IEEE Internet of Things Journal* **7**(3) (2020), 2421–2431. doi:10.1109/JIOT.2019.2957835.
[28] Y. Kang, S. Liu, H. Zhang, Z. Han, S. Osher and H.V. Poor, Task Selection and Collision-Free Route Planning for Mobile Crowd Sensing Using Multi-Population Mean-Field Games, *IEEE Transactions on Green Communications and Networking* (2021), 1–1. doi:10.1109/TGCN.2021.3086001.
[29] J. Chen and J. Yang, Maximizing Coverage Quality with Budget Constrained in Mobile Crowd-Sensing Network for Environmental Monitoring Applications, *Sensors* **19**(10) (2019). doi:10.3390/s19102399. https://www.mdpi.com/1424-8220/19/10/2399.
[30] T.N. Nguyen and S. Zeadally, Mobile Crowd-Sensing Applications: Data Redundancies, Challenges, and Solutions, *ACM Trans. Internet Technol.* **22**(2) (2021). doi:10.1145/3431502.
[31] R. Zhou, Z. Li and C. Wu, A Truthful Online Mechanism for Location-Aware Tasks in Mobile Crowd Sensing, *IEEE Transactions on Mobile Computing* **17**(8) (2018), 1737–1749. doi:10.1109/TMC.2017.2777481.
[32] M. Yu, H. Lin and J. Hu, A Data Trustworthiness Enhanced Reputation Mechanism for Mobile Crowd Sensing, in: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2017, pp. 743–747. doi:10.1109/iThings-GreenCom-CPSCom-SmartData.2017.114.
[33] H. Jin, L. Su, H. Xiao and K. Nahrstedt, Incentive Mechanism for Privacy-Aware Data Aggregation in Mobile Crowd Sensing Systems, *IEEE/ACM Transactions on Networking* **26**(5) (2018), 2019–2032. doi:10.1109/TNET.2018.2840098.
[34] L. Gupta, R. Jain and G. Vaszkun, Survey of Important Issues in UAV Communication Networks, *IEEE Communications Surveys & Tutorials* **18**(2) (2016), 1123–1152. doi:10.1109/COMST.2015.2495297.
[35] A.A. Khuwaja, Y. Chen, N. Zhao, M.-S. Alouini and P. Dobbins, A Survey of Channel Modeling for UAV Communications, *IEEE Communications Surveys & Tutorials* **20**(4) (2018), 2804–2821. doi:10.1109/COMST.2018.2856587.
[36] A. Cho, J. Kim, S. Lee and C. Kee, Wind Estimation and Airspeed Calibration using a UAV with a Single-Antenna GPS Receiver and Pitot Tube, *IEEE Transactions on Aerospace and Electronic Systems* **47**(1) (2011), 109–117. doi:10.1109/TAES.2011.5705663.
[37] F. Restuccia, N. Ghosh, S. Bhattacharjee, S.K. Das and T. Melodia, Quality of Information in Mobile Crowdsensing: Survey and Research Challenges, *ACM Trans. Sen. Netw.* **13**(4) (2017). doi:10.1145/3139256.
[38] C. Zhang, L. Zhu, C. Xu, X. Liu and K. Sharif, Reliable and Privacy-Preserving Truth Discovery for Mobile Crowdsensing Systems, *IEEE Transactions on Dependable and Secure Computing* **18**(3) (2021), 1245–1260. doi:10.1109/TDSC.2019.2919517.
[39] Y. Liu, Z. Zheng, F. Wu, X. Gao and G. Chen, Fault tolerant mechanism design for time coverage in crowdsensing system, in: *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, 2017, pp. 1–8. doi:10.1109/PCCC.2017.8280460.
[40] K.I. Kilic and L. Mostarda, Heuristic Drone Pathfinding Over Optimized Charging Station Grid, *IEEE Access* **9** (2021), 164070–164089. doi:10.1109/ACCESS.2021.3134459.
[41] R. Chandrasekaran, The weighted euclidean 1-center problem, *Operations Research Letters* **1**(3) (1982), 111–112. doi:https://doi.org/10.1016/0167-6377(82)90009-8.
[42] Y. Zheng, L. Zhang, X. Xie and W.-Y. Ma, Mining Interesting Locations and Travel Sequences from GPS Trajectories, in: *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 791–800–. ISBN 9781605584874. doi:10.1145/1526709.1526816.
[43] Y. Zheng, Q. Li, Y. Chen, X. Xie and W.-Y. Ma, Understanding Mobility Based on GPS Data, in: *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 312–321–. ISBN 9781605581361. doi:10.1145/1409635.1409677.
[44] L. Pappalardo, F. Simini, G. Barlacchi and R. Pellungrini, scikit-mobility: a Python library for the analysis, generation and risk assessment of mobility data, 2019.
[45] R. Hariharan and K. Toyama, Project Lachesis: Parsing and Modeling Location Histories, in: *Geographic Information Science*, M.J. Egenhofer, C. Freksa and H.J. Miller, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 106–124. ISBN 978-3-540-30231-5.
[46] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, AAAI Press, 1996, pp. 226–231–.
[47] J.A. Hartigan and M.A. Wong, A k-means clustering algorithm, *JSTOR: Applied Statistics* **28**(1) (1979), 100–108.

[48] K.I. Kilic, O. Gemikonakli and L. Mostarda, Voronoi Tesselation-Based Load-Balanced Multi-Objective Priority-Based Heuristic Optimisation for Multi-Cell Region Coverage with UAVs, *Int. J. Web Grid Serv.* **17**(2) (2021), 152–178–. doi:10.1504/ijwgs.2021.114574.