Vehicular Crowdsensing for Smart Cities

Tzu-Yang Yu, Xiru Zhu, and Muthucumaru Maheswaran

Abstract As smart vehicles begin to roam the streets, new possibilities will emerge for large-scale data acquisition tasks necessary for proactive smart cities applications. Unlike mobile devices, smart vehicles carry powerful sensors and are highly mobile; they can cover large areas and perform high quality sensing. However due to restricted reward structures and limited bandwidths of cellular and VANETs, not all vehicles can participate equally. Thus, we must find a method for selecting promising participants which can efficiently the required collect sensing information. In this chapter, we present ideas for participant selection under varying conditions from large scale crowdsensing to personalized crowdsensing. We present several algorithms using a common framework.

1 Introduction

Modern vehicles are equipped with increasingly powerful sensors, communication interfaces and computing resources. As such, vehicular crowdsensing is quickly becoming a new paradigm for data collection [10, 14]. Data collected from crowd sensing could become essential for a smart city to provide dynamic and proactive services. Vehicles are recruited as sensing participants for large-scale crowdsensing tasks such as urban sensing or traffic condition monitoring. Compared to conventional mobile crowdsensing, vehicles are ideal platforms to collect, store, compute, and share large amounts of sensor data. The advantages are manifold; for instance, vehicles have greater mobility and cover wider sensing area. Furthermore, the mobility patterns of vehicles are predictable due to the prevalence of navigation systems. Most importantly, the abundance of on-board resources and lack of power constraints enable complex and long-running sensing tasks.

T.-Y. Yu · X. Zhu · M. Maheswaran (✉)
McGill University, Montreal, QC, Canada
e-mail: tzu-yang.yu@mail.mcgill.ca; xiru.zhu@mail.mcgill.ca; maheswar@cs.mcgill.ca

M. Maheswaran, E. Badidi (eds.), *Handbook of Smart Cities*,
https://doi.org/10.1007/978-3-319-97271-8_7

The primary application of modern vehicular crowdsensing research is generalized and large scale monitoring such as environment and traffic monitoring, map updating, public safety, urban sensing and so on [17, 25]. As such, data collected are primarily analyzed in a cloud server and results made available for public use. Such information can be reused by multiple applications. Given its nature, large-scale sensing is dominated by enterprises or governments. We believe that the benefit of crowdsensing paradigm should be available for personal use; tasks tend to be numerous but limited in scale. We define this paradigm as Personalized Vehicular Crowdsensing (PVC) [22]; it focuses on supporting user-specific sensing tasks.

Unlike generalized crowdsensing tasks, user-specific sensing tasks catered towards users' custom requests and are unlikely to be shared with other users. For instance, different sizes of trucks require varying road width for driving and turning. However, due to construction, snow, events or even bad parking, passable roads may no longer be traversable. Hence, it is necessary to look in real time for a wide variety of road width for different size of trucks. Such road width requirement depends on type of truck; the system should allow user to tune the road width parameter as a sensing task.

One related application, Waze, also attempts to leverage vehicular crowdsensing for everyday users [7]. In Waze, participants form part of a community which gathers information such as police location, traffic or roadblocks location. However, unlike our proposal, Waze does not support variegated user inquiries; sensing tasks are predefined by the platform. In addition, Waze requires participants to actively enter information; a participant which sees an accident would need to manually enter such information while driving. In contrast, PVC does not require participants to be actively engaged. The client generate a customized sensing task as a runnable program and submit the program to selected vehicle. Selected vehicle execute the program, sends result back to the requester if the task objective is met.

Applications like Waze that rely on users manually entering sensing results may not be trustworthy if users intentionally submit faulty sensing results for their own benefit. For instance, users who want to have better traffic conditions while driving can report accidents on the road. Thus, vehicles moving in the same direction may be directed to other routes by the system. In contract, our PVC allows the sensing program to implement security policies. The participant running the sensing program must follow the policies; otherwise the task result will be rejected. Thus, a program based sensing task not only can support customized sensing tasks but can also reduce security concerns.

As crowdsensing systems leverage participants for collecting data, an incentive system is necessary to maintain participation. To incentivize more users to participate in a crowdsensing system, the platform should reward participants. Conventional mobile crowdsensing often require complex and fine-grained incentive mechanism; it requires participants use their mobile phones and actively gather sensing data from their local environment. This can lead to significant inconvenience to participants. Besides, some tasks require participants to have specific knowledge before participating [16]. For instance, a task aimed at collecting photos of rare plant species may want to recruit participants with some knowledge of Botany.

Vehicular crowdsensing, on the other hand, does not require human involvement for completing tasks. Indeed, the quality of sensing results depends on the on-board sensors rather than the human factor. The participant only lend the on-board resources to the recruiter, which can be seen as buying computational resources from cloud servers. However, given that vehicles are owned by individuals, there exists significant privacy and security issues compared to cloud servers. Thus, typical payment schemes used in cloud computing cannot be directly applied for vehicular crowdsensing.

Moreover, unlike conventional cloud computing which consists of numerous of static servers, vehicular crowdsensing system comprises a dynamic collection of vehicles (mobile servers). Because of spatial-temporal nature of moving vehicles, efficiently selecting and utilizing vehicular participants' on-board resources is one of the key challenges in building a vehicular crowdsensing service. Although many participant recruitment algorithms has been recently added to the literature, unique characteristics of PVC have introduced several new challenges in designing participant recruitment algorithms.

In this chapter, we focus on providing a discussion on the concept of vehicular crowdsensing for smart cities, with focus on how participant selection algorithm is used under different crowdsensing paradigms and specific application needs. We also explore the challenges of providing recruitment algorithms for PVC as well as propose and evaluate several recruitment algorithms for PVC tasks.

2 Background and Characteristics

Crowdsensing refers to the outsourcing of data collection to users. This means that participants contribute to a shared pool of information with mobile sensing devices. Here, sensing devices often refer to smart phones or increasingly refer to vehicles with powerful sensors. We distinguish between Vehicular and Mobile crowdsensing given each has distinct characteristics. For instance, the mobility pattern of vehicles is predictable due to the prevalence of navigation systems. Such predicted path is often utilized for efficient participant selection for Vehicular crowdsensing tasks.

2.1 Two Different Crowdsensing Paradigms

In the literature, crowdsensing can be categorized into two type of paradigms: public crowdsensing [14] and personalized crowdsensing [22]. The primary application of public crowdsensing research is generalized and large-scale monitoring such as environment, traffic monitoring, map updating, public safety, noise pollution assessment, or urban sensing. The aggregated data is often shared to the public and can be reused by multiple applications. As such, sensed data is collected and

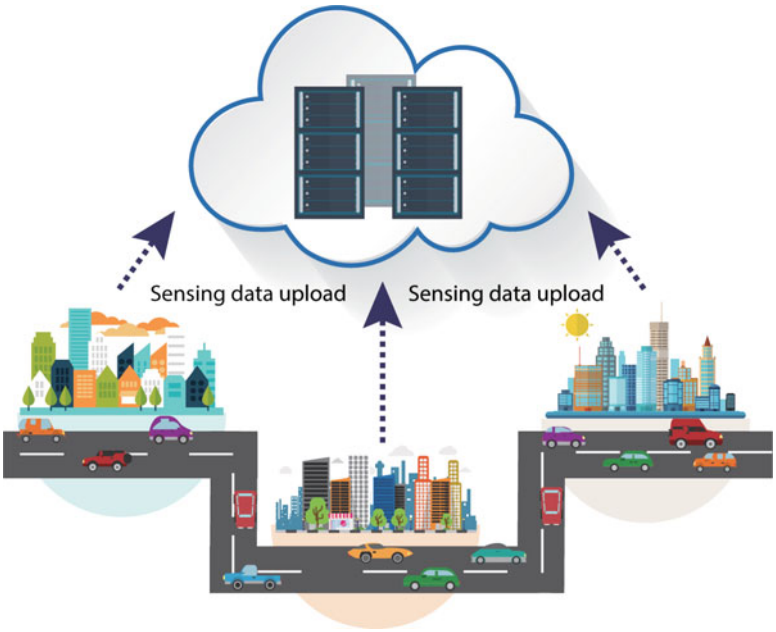

Fig. 1 Conventional public vehicular crowdsensing

analyzed within the cloud server itself and then made available for public use. Given its scale, such large scale sensing are dominated by enterprises or governments.

In terms of architecture for public sensing, the server schedules optimal set of vehicular participants to cover the sensing region and collects sensing data from selected participants as shown in Fig. 1. The information sensed is often fixed, such as air quality or noise level; hence, a server would continuously request the same type of information.

In contrast, personalized crowdsensing focuses on supporting user defined sensing tasks. These tasks cater towards a specific user's requests and hence are unlikely to be requested by other users. Thus, data can also be locally processed on the sensing device; the server only need to forward the results. For instance, different size of trucks need varying road width for driving and turning. However, due to construction, snow, or events, it's possible roads width change and are no longer traversable. Hence, it is necessary for the trucker to get real time road width information ahead of his path by hiring participants. These participants are hired to perform the specific user defined sensing task of looking at road width and need to follow specific spatio-temporal constraints from the recruiter. Thus the sensing result is unlikely to be shared with other users.

The Fig. 2 depicts the architecture of a personalized crowdsensing service. As shown in Fig. 2, recruiters send task requests to the server. The server retrieves requests and assigns tasks to appropriate participants. Participants utilizes on

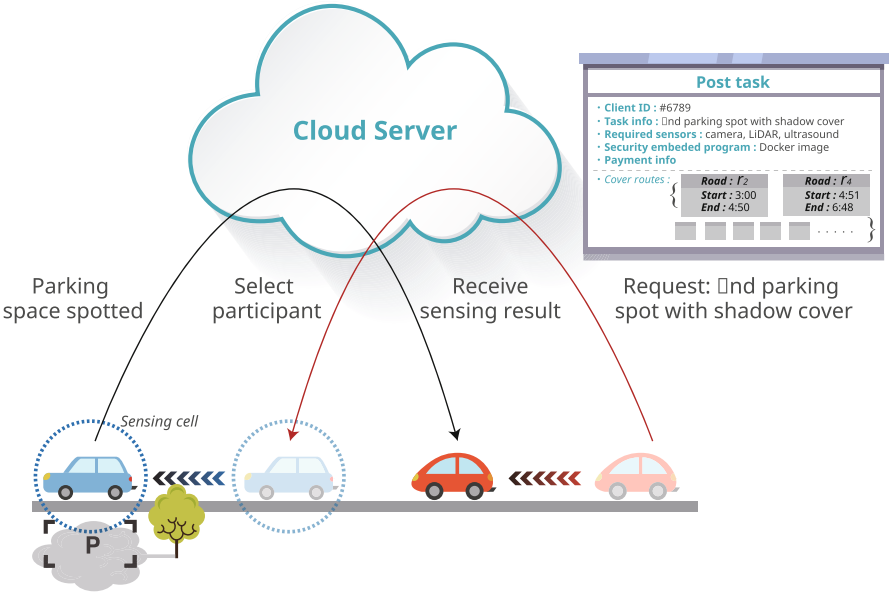

Fig. 2 Personalized vehicular crowdsensing

board resources to process sensing data. When the requested task is successfully completed, participants will send back sensing results to the corresponding recruiter via the cloud server. With personalized sensing, data requests are frequently received from multiple recruiters. The cloud server collects client requests and schedules such requests to participants to fulfill.

2.2 Central Server for Vehicular Crowdsensing

With vehicular crowdsensing, one major problem is the reliability of data received. We would be troubled if a participant sent purposefully misleading data. Hence, central control is necessary to maintain reliable information for the two vehicular crowdsensing paradigms introduced above. Second, there should exist a reward system for which providers of reliable and timely data are appropriately rewarded. Otherwise, it may be difficult to maintain user participation. Third, we seek to reduce redundant data. Unlike fixed sensors, vehicles move around and it's likely that sensing coverage overlap with others. Thus, the central server need to be equipped with data de-duplication mechanisms to avoid overwhelming by gathering all possible data to consumer.

3 Public Vehicular Crowdsensing

3.1 Background

In this section, we focus on participant selection for public vehicular crowdsensing. Public vehicular crowdsensing targets large scale sensing, such as air quality monitoring, traffic monitoring or even public safety. Public Vehicular Crowdsensing's purpose is to gather as much information as possible in the targeted sensing region. Given its scale, not every potential participant can and should be selected. The problems are threefold. First, each participant must be rewarded; this reward is often monetary [23]. For instance, Amazon Mechanical Turk is an example of crowdsourcing with monetary incentive where users perform tasks for small amounts of money [4]. Second, local network infrastructure can be overwhelmed when large number of vehicles continuously collect and send data. Vehicular crowdsensing requires significant overheads; Current vehicular communication technology have limitations; vehicular ad-hoc networks do not have a central server to control bandwidth and suffer from reliability issues [2]. In contrast, cellular networks are more reliable but current cellular network data usage is already growing exponentially; This risks severe network congestion in the future [1]. Third, significant sensing overlap exist between vehicles; this means selecting all vehicles would results in large amount of redundant data. For instance, sensing air quality using every vehicles in traffic would result in processing large quantity of information; selecting only a few would have been sufficient. Processing large quantity of data requires large data centers which adds costs. Thus, public vehicular crowdsensing participant selection attempts to select participants which can maximize coverage while also limiting costs.

3.2 System Model

Public vehicular crowdsensing is composed of a set of participant vehicles with wireless transceiver and sensor devices. Typically, participants are either smart cars or buses; both can carry onboard computing resources and sensing ability. We also assume we cannot control participants movements; participants do not actively participate in sensing tasks. Selected participants only gather data passively along their route as they pass locations of interest.

To communicate with the cloud or other vehicles, current vehicular crowdsensing relies on vehicular ad hoc networks (VANET) and LTE. VANET utilize IEEE 802.11p, Wireless Access for the Vehicular Environment as standard (WAVE). They form temporary network of vehicular clusters and road side units (RSUs) [5]. Vehicles can communicate with neighbors in the immediate cluster directly but require RSU's assistance to send data beyond the local cluster. However, VANET suffer from reliability problems [5]. To fix reliability issues, some works for VANET include LTE [18].

To store accumulated data, cloud servers are necessary. Then, applications could request for sensing information from the cloud servers. We assume the cloud servers know the participants' current location; participants could beacon that information from time to time. Many works in the literature assume knowledge of participants' predicted route; this can be information from a GPS for instance. However, some work in the literature only assume knowledge of past routing information. At set time intervals, the cloud servers can schedule a subset of available participants to collect sensing data.

When a vehicle participates in crowdsensing, it must be rewarded. Designing an incentive model which can consistently attract participant is a integral part of crowdsensing. Important aspects to be considered are costs, sensing quality, sensing coverage. Not all incentive models result in monetary gain [23]. Waze is an vehicular crowdsensing application, where users share sensing information on the platform; users are both participants and consumer of sensing information. For monetary incentive system, there exists reverse auction where participants bid for sensing tasks [11, 24]. To prevent low quality sensing, rewards based on data quality are often considered [15]. Other systems rely on game theory such as Stackelberg game. Here, a consumer proposes an offer to all participants to consider. This repeats for multiple iteration until sensing needs are met [20].

Besides minimizing cost, maximizing the amount of area covered and amount of area covered N times over the targeted region is paramount. Overall, there are two properties for coverage to consider; spatial coverage and temporal coverage. Spatial coverage simply seeks to maximize amount of area covered over a time window. The weakness of this approach is that areas covered by sensors can be unbalanced; some areas may never be covered while other areas may be always covered. Thus, popular roads will always have sensing participants on it while side roads suffer in terms of coverage frequency. However, this approach tend to work well with sparse distribution of participants [10]. In contrast, temporal coverage is about maximizing the number of regions covered at least N times. This would result in more even distribution for collected data but may reduce the overall quantity of data collected. Furthermore, coverage may not be as complete. There's little incentive to gather more data from an area already covered even though some new sensing information could be gained. A third approach exists; by combining both spatial and temporal coverage properties, we can obtain a hybrid approach [14]. In this approach, each vehicle can only gather a capped amount of information at a specific area.

3.3 Definitions and Assumptions

We define notations to be used when describing participant selection algorithms. We define a sensing region as $R = \{r_1, r_2, \ldots, r_m\}$, composed of smaller areas. The exact definition and size of each area r_i differs with each approach. For instance, in [10], each area consist of graph nodes whereas in [21] they consist of 1 meter by 1 meter square areas. We define the set of vehicles as $V = \{v_1, v_2, \ldots, v_n\}$. We

Table 1 List of notation for public vehicular crowdsensing

Notation	Description
T	Sensing time window
V	Set of all vehicles
P	Set of vehicular participants selected for sensing
R	Set of areas for sensing
t_i	Time i within time window T
v_i	Vehicle i in V
p_i	Participant i in P
r_i	Area i in R
r_{v_i,t_j}	Location of vehicle v_i at time t_j
$C(v_i)$	Function which returns the cost of participant v_i
B	The total budget of sensing task

assume each v_i has sensors and is willing to collect information. We define vehicles selected as participants as $P = \{p_1, p_2, \ldots, p_m\}$, where $P \subseteq V$. We define the time window as $T = \{t_1, t_2, \ldots, t_q\}$, where each t_j is a time unit. Let r_{v_i,t_j} be the location of vehicle v_i at time t_j, where $r_{v_i,t_j} \in R \cup \emptyset$. We define C as a function which when given a vehicle returns the cost of recruiting such vehicle. We define B as the budget constraint which limits the number of participants we can select. Finally, we define $coverage$ as a function which takes in a participant set and sensing region and returns the coverage measure of selecting such participant set (Table 1).

3.4 Problem Statement

Public vehicular participant selection problem can be expressed as integer linear programming problem. We assume that the coverage function is provided. This problem has been shown to be NP hard; this can proved by reducing the problem to a set cover problem [9, 10, 21].

Input Values Sensing region R, vehicle set V, cost function C, Budget B, coverage function $coverage$

Objective Function Find a set of participants P best fit as sensing participants.

Maximize $coverage(P, R, \ldots), P \subseteq V$

Subject to $\sum_{p_i \in P} C(p_i) <= B$

3.5 Participant Selection Algorithms

In this section, we will explore a variety of algorithms proposed for public vehicular crowdsensing participant selection. Many works in the literature have covered

mobile crowdsensing but have only recently started covering participant recruitment for vehicular crowdsensing.

Hamid et al. first proposed the idea of utilizing vehicular trajectory to best select vehicular participants [8]. In this paper, each participant has a reputation based on its past sensing results; participant commitment and quality of information provided. Participant commitment is the likelihood a participant will follow its provided trajectory. Quality of information is the quality of previous sensing collection by the participant. Hence the reputation of a participant v_i is given by the following equation where α and β are weights and p and q are participant commitment and quality of information respectively.

$$reputation(p_k) = \alpha * p_{p_k} + \beta * q_{p_k} \tag{1}$$

The reward for each participant, C_{p_k} is based upon a fixed cost plus a variable cost based on coverage distance d_{p_k} and reputation.

$$C(p_k) = cost_{fixed} + cost_{variable} * d_{p_k} * reputation(p_k) \tag{2}$$

The problem is formulated as two step integer linear programming, one for maximizing the number of regions covered and the second for minimizing cost of participants which achieves maximum coverage. In the first step, Hamid et al. maximizes coverage by maximizing the average number of regions covered while remaining within budget B.

$$hamid\_coverage(P, R) = \sum_{t_j \in T} | \bigcup_{p_i \in P} r_{p_i,t_j} | \tag{3}$$

In the second step, the distance covered by selected vehicles is minimized while maintaining the same level of coverage in step one. This will minimize the total cost.

$$hamid\_step_2(P, R) = \sum_{p_i \in P} p_i * d_{p_i} \tag{4}$$

To solve the above equations, Hamid et al. simply utilized a integer programming solver, Gurobi 5.1. Integer linear programming is NP-hard but there are methods for approximations. However, even with approximations, the runtime of the method is bound to be a higher order polynomial. Thus, Hamid et al. method of solving vehicular participant takes too long for large-scale public sensing.

Han et al. presents two participant recruitment algorithms based on predicted trajectory [9]. Note that the cost, $c(v_i)$ for selecting v_i is assumed to be 1 where $\forall v_i \in V$. The cost of selecting a vehicle is the exact same as selecting any other vehicle. The budget constraint B is the number of participant selected. The $coverage$ function measure temporal coverage; it computes number of areas covered by sensors at least once.

$$han\_coverage(P, R) = |\bigcup_{t_j \in T} \bigcup_{p_i \in P} r_{p_i,t_j}| \tag{5}$$

Finally, to evaluate the algorithms, Han et al. utilized a dataset consisting of real GPS trace of 20,000 shanghai taxi from 08:42–09:42. The first algorithm, referred as the offline algorithm, assumes full knowledge of all vehicles and their trajectory within the time window T. It finds the vehicle which adds the most coverage and selects it as part of the solution. The algorithm then iterates until B vehicles have been selected. The algorithm's time complexity is $O(B * |V| * \log(|V|))$. In contrast, the second algorithm, referred as the online algorithm, assume no prior knowledge of a vehicle before it joins the crowdsensing system. Since the system wouldn't know of a vehicle and its trajectory before it comes into range, this may be a more realistic assumption. The algorithm decides whether to select a vehicle v_i when it joins the system by comparing the gain in temporal coverage of adding v_i with a dynamic threshold. The dynamic threshold is computed based on the number of participants already selected. The online algorithm's time complexity is $O(B * |V|)$.

Similarly, He et al. also proposed two participant recruitment algorithms based vehicular trajectory [10]. Both algorithms assume full knowledge of vehicles and their trajectory within time window T. The crowdsensing cost C is generated according to a normal distribution. To evaluate the solution, traffic trace dataset was obtained from TAPAS-Cologne [19], a 24 h generated vehicular trace of the city of Cologne in Germany simulated using SUMO [13]. The first algorithm consist of a greedy approximation which maximizes spatial coverage and is meant for small number of sparsely deployed participants. Here, He et al. define spatial coverage as simply the number of areas covered over a time period T.

$$he\_coverage\_1(P, R) = \sum_{t_j \in T} |\bigcup_{p_i \in P} r_{p_i,t_j}| \tag{6}$$

Similar to the offline algorithm by Han et al. this algorithm adds the most cost effective participant and iterates until the budget constraint is met. Cost effectiveness is defined as the difference in spatial coverage divided by its cost. The algorithm has a time complexity of $O(|V|^2|T|)$ Since this algorithm does not take into account temporal coverage, this may result in unbalanced data distribution but maximizes the amount of information gathered. This is beneficial with a small number of participants sparsely deployed. In addition, this algorithm is a $(|T| + 1)$ approximation algorithm; quality suffers when the time window is long. In contrast, the second algorithm proposed is a genetic algorithm meant for large number of densely deployed participants. It utilizes the minimum covered time for all areas as coverage.

$$he\_coverage\_2(P, R) = min(\bigcup_{r_k \in R} he\_coverage\_1(P, r_k)) \tag{7}$$

The genetic algorithm encodes vehicle selection outcome as a binary string. Thus, with 5 possible participants, if we only select the first participant, the encoding would be "10,000". Initially, a large number of solutions are randomly generated. At each generation, the coverage, based on temporal coverage is computed and only a top percent of the population survives. In addition, mutation and crossover operations occur as well to mimic the evolutionary processes. Crossover combines two solutions to obtain a hybrid of the two. Mutation randomly changes part of the selection outcome. Finally, solutions which violate the cost constraint are trimmed to fit. The algorithm runs until the time limit is reached or until the theoretical upper bound is reached. The main advantage of this algorithm is that it can be capped in terms of runtime which allows selection for large number of participations; on the downside they may result in weaker solutions.

Due to polynomial time complexity from many proposed algorithms, Yi et al. proposes a linear time algorithm for participant selection based on submodular property of the problem [21]. This would benefit large-scale vehicular crowdsensing where scheduling time may be tight. Coverage is defined as the maximum number of areas covered over T.

$$yi\_coverage(P, R) = \sum_{t_j \in T} |\bigcup_{p_i \in P} r_{p_i,t_j}| \tag{8}$$

The utility of a recruiter is defined as the number of coverage minus the costs of participants recruited where λ is a weight parameter. The objective is to maximize the utility of the recruiter. Note that there is no hard cap for number of participants; λ can be adjusted to serve as a soft cap. The utility function is proved to be submodular

$$utility(P, R) = yi\_coverage(P, R) - \lambda * \sum_{p_i \in P} C(p_i) \tag{9}$$

The algorithm relies on a forward and reverse greedy algorithm operating at the same time. Thus, the algorithm begins with an empty set and also a set of all participants. At each iteration, we consider whether to add a participant v_i by adding v_i to one of the empty set and removing v_i from the full set. The change in utility by adding and removing a vehicle is utilized to compute a probability of whether to include the participant as part of the solution. The runtime complexity of the algorithm is $O(|T| * |R| * |V|)$ and thus is linear. It achieves an approximation ratio of $1/2$. To evaluate the algorithm, Shanghai Taxi dataset was utilized, covering the trajectory of 4316 taxis. Furthermore, a synthetic dataset was generated using Gauss-Markov mobility model. Results show that the algorithm only does slightly worse than a polynomial greedy algorithm but took considerably less time to run.

In contrast to generalized public crowdsensing approaches, Gao et al. proposes an air monitoring vehicular crowdsensing system using buses [6]. Unique to Mosaic's approach is that unlike passenger cars, buses have fixed routes. Thus the first algorithm selects entire bus routes based on achievable coverage. To expand on the first algorithm, the second algorithm selects individual buses. Furthermore,

instead of seeking to cover an entire area, Mosaic proposes Points of Interests (POI) which are high priority sensing locations. These high priority area could be schools, hospital or other public spaces. The priority between POI are considered to be equal. The sensing region R is split into 100 by 100 m areas; this may feel somewhat rough in terms of precision but is acceptable for air quality monitoring. Each area has an importance value, δ, attached, based on its distance to the nearest POI. Hence, an area close to a POI has higher δ compared to an area far from any POIs. Since air quality can be inferred from nearby measurements, coverage can be defined as the number of route passing within or near the area.

$$local(r_k, R, POI) = \begin{cases} \delta(r_k, R, POI) & > 2\, routes\ passing\ thru\ r_k \\ 0.75 * \delta(r_k, R, POI) & 1\ or\ 2\, routes\ passing\ thru\ r_k \\ 0.5 * \delta(r_k, R, POI) & >= 1\, routes\ 1\, areas\ away \\ 0.25 * \delta(r_k, R, POI) & >= 1\, routes\ 2\, areas\ away \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$coverage(V, R, POI) = \sum_{r_i \in R} local(r_i, V, POI) \tag{11}$$

The first algorithm proposed to find the best bus route is similar to the greedy algorithms mentioned in previous papers. It finds the route which adds the most coverage and adds it as part of the solution; this is iterated until the *Budget* is reached. However, since this algorithm does not consider the temporal dimension of coverage, a second algorithm for bus selection is proposed. In this second algorithm, coverage equation is slightly modified to include percent of time coverage. Similarly, the algorithm attempts to select the bus with highest coverage and adds it as part of the solution. It iterates until the budget B is reached. The algorithm's time complexity is $O(B * |V| * |R|^{0.5})$ and takes about 10 s to run with 1415 buses. To evaluate Mosaic, data was collected from a $2.9 \times 3.1\,km^2$ city area in China. It consisted of 282 bus routes and 1415 buses schedule with 72 POI consisting of school and hospital locations. Mosaic proposed a crowdsensing system which works well for air quality monitoring but would suffer when crowdsensing for other type of sensing data. Here, temporal coverage is less emphasized; air quality changes does not occur as suddenly. This paper does not consider participants selection for general sensing tasks.

The participant recruitment algorithms proposed above only considered a single type of sensing data. However, sensors on vehicles are heterogeneous, furthermore quality of sensors may differ. Hence, Liu et al. proposed a heterogeneous participant selection algorithm [14]. Unlike other works mentioned, Liu et al. utilize a time continous markov chain mobility model rather than assume to know a participant's trajectory. Thus, given a vehicle's position, a vehicle has an average stay duration and a likelihood of transitioning to another area. The longer a vehicle stays in an area, the more data it gathers. The cost of selection is 1 for each participant and the Budget is the maximum number of vehicle which can be selected. Thus the

coverage is simply how much information all participant can gather from all sensors it possess. The objective is to maximize the total coverage while remaining within the $Budget$.

The algorithm for selection is once more a greedy algorithm which adds the best participant at each iteration and repeats until $Budget$ has been reached. This is O($|Budget| * |V|$) in terms of time complexity since it only seeks to find the maximum rather than sorting. To evaluate this algorithm, GPS trace of T-Drive trajectory dataset are used. This contain the trajectories of 10,357 taxis and about 15 million data points.

One weakness of current vehicular recruitment strategy is that once selected, a participant must continue sensing for a fixed period of time. However, inefficiency exists; the participant selected may be only truly cost effective for part of the time window T selected. Furthermore, trajectory provided by participants tend to be error prone in reality. Thus, Hu et al. have proposed a variable duration participant recruitment with uncertain trajectory [12].

To deal with uncertainty, Hu et al. proposes a probabilistic method for estimating location of a vehicle given a trajectory. This probability can be obtained from historical data. Let $prob(r_k, t_j, v_k)$ be the likelihood of vehicle v_k to be at region r_k at time t_j. Thus, the position vehicle i, r_{v_i,t_j} instead of returning a single area r_k, returns a R sized matrix of probability. Note that $|r_{v_i,t_j}| = 1$.

$$r_{v_i,t_j} = \bigcup_{r_k \in R} prob(r_k | v_i, t_j) \tag{12}$$

Furthermore, the solution for variable duration participant recruitment is a $|V|$ by $|T|$ matrix denoting whether v_i is recruited at t_i instead of a set of selected vehicles.

$$select_{v_i} = \begin{bmatrix} v_{11} & \delta_{12} & \delta_{13} & \dots & \delta_{1q} \\ v_{21} & \delta_{22} & \delta_{23} & \dots & \delta_{2q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{n1} & \delta_{n2} & \delta_{n3} & \dots & \delta_{nq} \end{bmatrix}$$

$$select_{v_i,r_j,t_k} = \begin{cases} 1 & Selected \\ 0 & Otherwise \end{cases} \tag{13}$$

In addition, the cost for selecting participants is defined by a $|V|$ by $|T|$ matrix, where a cost is associated for each time point for each vehicle. Thus, cost function for a participant v_i is defined as follows.

$$C(v_i) = \sum_{t_j \in T} C_{v_i,t_j} * solution_{v_i,v_j} \tag{14}$$

Finally, Hu et al. defines spatial coverage as a function of the vehicle trajectory probability and the required number of vehicles for sensing at area r_k, $required_{r_k}$. Thus, the number vehicles selected at r_k at time t_j must be greater or equal than $required_{r_k}$ or else coverage is defined as 0. Let P_{r_k} be the set of participants selected which covers region r_k.

$$SC(r_i, t_j, V) = \prod_{p_k \in P_{r_i}} prob(r_i, t_j, p_k) \tag{15}$$

$$SC_{case}(r_i, t_j, V) = \begin{cases} SC(select, r_i, t_j) & \sum_{v_k \in V} select_{r_i, t_j, v_k} > required_{r_k} \\ 0 & otherwise \end{cases} \tag{16}$$

$$coverage(select, R) = \sum_{r_i \in R} \sum_{t_j \in |T|} SC_{case}(select, r_i, t_j, V) \tag{17}$$

Note that this coverage function will penalize for overlap as the coverage probability is multiplied for every vehicle covering an area. The objective of the algorithm is to select a set of vehicles at different time points which maximizes the coverage function while staying within budget B. Given that variable duration participant selection problem is a subset of standard vehicular participant selection, the problem is NP-hard. Thus, the first step is a pruning algorithm to remove non-viable participants while the second step is a greedy algorithm to select the vehicle and time for sensing.

The pruning step computes a Pearson correlation matrix to find vehicles with significant trajectory overlap. Vehicles with significant overlap are grouped together. The vehicle with lowest average costs within the group will be preferred given they cover similar areas at similar times. As such, the algorithm can prune to only $|V'|$ participants. The step has a time complexity is $O(|V'||V|)$.

In the second step, vehicles are still selected at specific times. The algorithm simply seeks to iteratively recruit the best participant v_k for each area r_k at each time t_j. To find such best vehicle, Hu et al. define the following SC efficiency measure to rank participants.

$$efficient_{v_i, r_j, t_k} = SC(select_{t_k}, R) - SC(select_{t_k - 1})/C_{v_i, t_j} \tag{18}$$

This measures the marginal increase in spatial coverage for a single vehicle at time step t_j. Thus, the time complexity is $O(|V'|^2|T|^2)$.

To evaluate their algorithms, Hu et al. utilized a trace dataset from taxis in Rome over an area of 64 km^2 [3]. The duration of the dataset is over 30 days. Thus, the model proposed attempts to select participants at each t_i during a given time window. This improves coverage metrics but suffers from higher computational costs. Furthermore, the proposed model can only obtain greater coverage with the same workload by selecting more vehicles and switching off vehicles when they are no longer useful for sensing. This has a few problems; first reward system

could include a fixed cost as part of the recruitment; recruiting more vehicles would increase costs [8]. Furthermore, sensing data collected may result in highly fragmented data from multiple sources; for instance a single time window T covered can have at most T vehicles covering each area. This is a problem because of lack of continuity and varying sensor quality. Compared to single continuous data, having multiple fragments of data increases the level of noise and reduce sensing quality.

4 Personalized Vehicular Crowdsensing

Public vehicular crowdsensing is more appropriate for large scale sensing such as environment and traffic monitoring, public safety, and urban sensing. Beyond public sensing, the benefit of crowdsensing can be available to support more personal tasks. However, unlike conventional large scale sensing tasks, the unique characteristics of personalized vehicular crowdsensing have introduced several new challenges in the design of participant recruitment algorithms. These include:

1. Personalized crowdsensing which targets every-day users have tighter budget constraints compared to the public crowdsensing supported by large enterprises or governments.
2. Requests by users are diverse; for instance, finding parking space, checking a favorite restaurant, etc. Evidently, it would be meaningless to employ large scale sensing; the sensing region in personalized vehicular crowdsensing is both location and time specific.
3. Personalized crowdsensing tasks are time sensitive compared with large scale sensing tasks. For instance in finding parking slot scenario, the spot can be taken before the client arrives. Thus the system needs to guarantee timeliness. On the other hand, we do not need to cover all sensing region at all time; we can reduce the number of sensing participants accordingly.
4. Because clients can submit requests as needed, multiple requests can be made by a single client. Since sensing data is processed locally, overloaded participants may fail to process all requests. Thus, on-line load balancing is necessary when selecting participants.

Traditional vehicular crowdsensing participant recruitment seeks to cover an entire area over a time window. Therefore traditional approaches cannot be applied to the Personalized Vehicular Crowdsensing Participant Recruitment problem (PVC-PR) because here we only consider a particular location at a specific time. For instance, suppose a recruiter is interested in finding a parking slot near his destination. Let client's route go through three unique regions $R = \{r_1, r_2, r_3\}$ with each region requiring 10 min to traverse. In such tasks, we do not need to know the parking status at region r_3 while we are still in region r_1. This is because the parking space might be taken away before we arrive at r_3 which requires a total 20 min driving time. Thus, traditional participant recruitment algorithms could suffer from over recruitment, which is very inefficient for PVC tasks. We only need

to maintain partial coverage at indicated locations instead of covering all locations all the time.

In the following sections, we propose and evaluate several algorithms specifically for PVC tasks. The main objective of these algorithms is to recruit the minimum set of vehicular participants which can complete the PVC tasks. We also ensure proper load balancing among all the participants to reduce the chance of task failures.

4.1 System Model and Assumptions

Similar to conventional crowdsensing systems, PVC is comprised of cloud servers and massive number of smart vehicles, and the vehicles are equipped with sensors and communication devices such as Wi-Fi and Cellular interfaces. When a vehicle begins its journey, general information such as unique vehicle ID and predicted route from navigation system are uploaded and stored in the cloud server. Vehicles need to reload predicted trajectories again if any changes occur to their planned routes. Recruiters can make queries on sensing data of interest to the server. The query needs to specify the sensing target and minimum distance from the recruiter. For instance, the recruiter is interested in finding parking space at least 1 km ahead but no longer than 5 km away. We consider such monitoring area as Monitoring Window (MW). Given the recruiter's planned route as the sensing route, the cloud server need to select a set of proper vehicular participants to complete such task. The participant selection decision is based on the provided monitoring area. As shown in Fig. 2, the monitoring area changes dynamically based on projected trajectory of the recruiter. Thus monitoring coverage should follow recruiters' movements.

4.2 Definitions

Before formally presenting the problem, we describe some definitions and notations. We consider the area of interest to be divided into a number of small road segments $R = \{r_1, r_2 \ldots r_m\}$. Each road segment has a single traffic direction. Roads which have opposite traffic directions are considered as two different road segments. The area also contains a set of vehicles V, and let $C = \{c_1, c_2 \ldots c_i\}$ be the set of clients and $P = \{p_1, p_2 \ldots p_j\} \subseteq V$ be the set of participants.

Client's sensing route and participant's projected route is composed of sequence of road segments as shown in Fig. 3. Each road segment contains a time stamp specifying arrival time of such vehicle. Let R_c be the set of route segments in client's query c and let R_p be the set of predicted future road segments of a given participant p. The time stamp of a given road segment can be derived from the following functions:

Fig. 3 Future routes and sensing routes

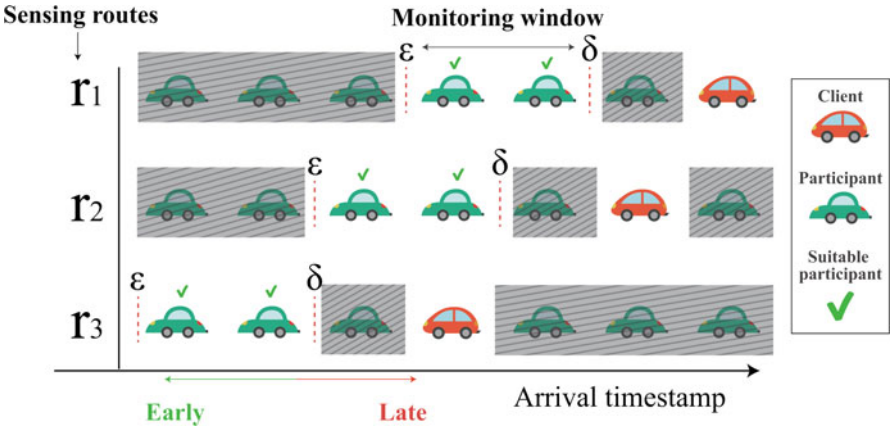

Fig. 4 Monitoring window (MW): a window for selecting useful sensing participants. For instance, a client needs a participant at least 1 min drive away $\delta = 60$ s, but cannot be more than 5 min drive away $\varepsilon = 300$ s

$$\mathscr{T}(r_k, R) = \begin{cases} \mathbb{R}^+ & r_k \in R \\ -MAX\_INT & \text{otherwise} \end{cases}$$

In order to select valid and suitable participants for a specific client, we use the following definitions:

Definition 1 (Common road segments) Given a client's query route R_c and participant's future trajectory R_p, the set of common road segments is defined as

$$R_{c,p}^{com} = R_c \cap R_p, \; c \in C \wedge p \in P \tag{19}$$

Definition 2 (Data timeliness) An important criteria to consider for participant selection is that data must arrive in a timely fashion to the client. In other word, data that arrives too early or too late is considered worthless to the client. Thus, participants need to be in a specific sensing window in order to provide useful sensing data. We refer to such window as monitoring window. Figure 4 shows how MW helps in selecting useful participants. Let δ and ε be the lower bound and upper bound for the window, respectively. The value of a participant at route r_k is represented as follow:

$$\Gamma(R_c, R_p, r_k) = \begin{cases} 1 & \delta \leq \mathscr{T}(r_k, R_c) - \mathscr{T}(r_k, R_p) \leq \varepsilon \\ & , r_k \in R_{c,p}^{com} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

Definition 3 (Query route coverage (QRC)) We define query route coverage as the total number of road segments a set of selected participants can cover for a single client under the associated timeliness constraint. Given a client, $c \in C$, and a participant, $p \in P$, a single participant coverage for a single client can be defined as:

$$single\_cover(c, p) = \left\{\forall r_k \in R_p | \Gamma(R_c, R_p, r_k) = 1\right\} \tag{21}$$

Thus, given a set of participants $S \subseteq P$, the function QRC can be defined as follows:

$$QRC(S) = \left| \bigcup_{p' \in S} single\_cover(c, p') \right| \tag{22}$$

Definition 4 (Participant maximum load (PML)) Participants will be assigned sensing tasks by different clients along its journey. Because of the spatio-temporal nature of the PVC tasks, certain regions may not have sufficient member of participants at specific spatio-temporal locations. Hence, some participants in popular sensing regions may be overloaded. Figure 5 shows an example of participant workload with 3 clients. In the figure, the participant p services 3 clients between timestamp t_1 and t_2 in region 2.

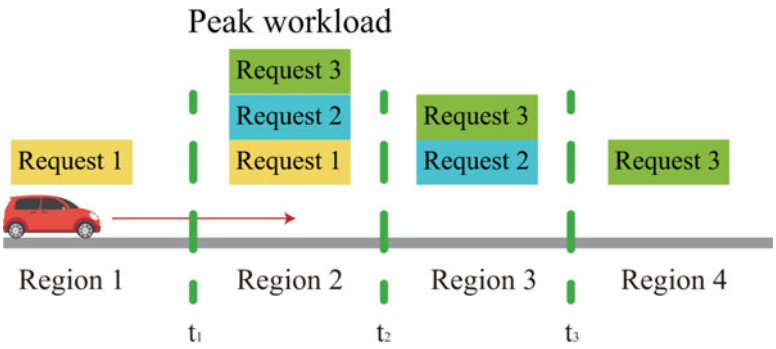

Fig. 5 Sample workload for a single participant with 3 clients; depending on clients location of interest, the workload is spatial-temporally assigned

Table 2 List of notation for personalized vehicular crowdsensing

Notation	Description
R	Set of road segments
C	Set of clients
P	Set of vehicular participants
R_c	The query route segments of the client $c \in C$
R_p	The predicted future road segments of the participant $p \in P$
$\mathscr{T}(r, R)$	Function to get the timestamp of a road segment r, given a set of road segments
$R_{c,p}^{com}$	The set of common road segments, obtained from $R_c \cap R_p$.
δ	Lower bound of monitoring window (MW)
ε	Upper bound of MW
$\Gamma(R_c, R_p, r)$	Value function return 1 if the timeliness constraints are satisfied and 0 otherwise
$single\_cover(c, p)$	The function return set of road segments such that $\forall r\ \Gamma(R_c, R_p, r) = 1$
ℓoad_p^{max}	The maximum workload the participant p will have for entire journey
β_c	Maximal number of vehicles that the client c can hire for doing the sensing task
cap_p	Soft threshold of maximum number of tasks that p is able to serve simultaneously

4.3 Problem Formulation

The PVC participant recruitment problem can be formulated as a two stage optimization. In the first stage, we seek to find the set of participants which can maximize requested coverage. That is, given a single client c and set of participants $P = \{p_1, p_2 \ldots p_j\} \subseteq V$. The objective function is defined as:

Objective function of the first stage recruit subset of vehicles $S' \subseteq P$ maximizing coverage subject to a budget constraint β_c.

$$\left\{ S' \in \arg\max_{S} \ QRC(S) \,\middle|\, |S'| < \beta_c \right\}$$

where β_c is the maximal number of vehicles that the client c can hire for doing the sensing task (Table 2).

Objective function of the second stage Since the first stage may return several solutions which achieves maximum coverage, we need to make sure the solution can achieve global load balance to guarantee quality of service. Thus, the objective of the second stage focuses on recruiting a set of participants from the first stage which reduces the workload among all vehicular participants given a stream of requests $\mathbf{c} = \{c_1, c_2 \ldots.\}$. The formulation of this stage is defined as below.

$$Minimize \sum_{p' \in P} \frac{\ell oad_{p'}^{max}}{cap_{p'}}$$

Where $\ell oad_{p'}^{max}$ signify the maximum workload participant p' will be service for, and $cap_{p'}$ is a soft threshold of maximum number of tasks that p' is willing to serve simultaneously.

4.4 Algorithm Design

In this section, we present our online participant selection algorithm for each incoming request. To ensure our solution can maximize coverage for requested sensing routes as well as spread global load balance, we use the following score function for selecting decision.

Definition 5 (Workload score function) To select proper participants, our workload score function which considers participant's current maximum workload, ℓoad_p^{max} is defined as follows:

$$Score(\ell oad_p^{max}) \leftarrow \frac{cap_p}{\ell oad_p^{max}}, \ \forall\, \ell oad_p^{max}, \ and \ \forall\, cap_p \in \mathbb{N}^+ \tag{23}$$

Where ℓoad_p^{max} indicates the maximum workload the participant p will have for the entire journey, and cap_p is a soft threshold of maximum number of tasks that p is willing to serve simultaneously.

The pseudocode detailed in Algorithm 1 shows how our score function is used for recruiting participants while considering load balance. In the algorithm, a participant is selected in each round, where the size of round S is capped by the size of the query routes $|R_c|$. The vehicle selection decision is based on the vehicle's weight w. The weight of the participant p' is calculated by its workload score times its route coverage for the clients query route,

$w \leftarrow Score(\ell oad_{p'}^{max}) \times single\_cover(c, p')$.

We select the vehicle which has the maximum weight w^{max} as shown in Algorithm 1 from line 7 to line 14.

The above algorithm assume availability of perfect predictions of candidate vehicles future route. Such assumption is not realistic given high potential of prediction errors. We evaluate predicted trip error based on different traffic levels using the TAPAS Cologne simulated vehicle trace. We found that as the length of a trip increases, the error in predicted locations of participant vehicles will increased (see Fig. 6). In the light traffic, prediction error increase slowly as the length in time of trips increase, whereas prediction error grows quickly in heavy traffic. We solve this issue by proposing a windowed based scheduling approach where we only schedule participants for set time windows instead of whole trips. We set the initial

Algorithm 1 Weight based Greedy algorithm (WBG)

Input c, P
Output $Participants$

$Participants \leftarrow \emptyset$
$S \leftarrow R_c$
while S is not empty **do**
  $p^{best} \leftarrow \emptyset$
  $R^{cover} \leftarrow 0$
  $w^{max} \leftarrow INT\_MIN$
  for $p\prime \in P$ **do**
    $w' \leftarrow Score(\ell oad_{p'}^{max}) \times |single\_cover(c, p')|$
    if $w' > w^{max}$ **then**
      $w^{max} = w'$
      $p^{best} \leftarrow p'$
      $R^{cover} \leftarrow \hat{R}$
    end if
  end for

  if p^{best} is empty **then**
    No valid participant, break the loop
  end if

  $S \leftarrow S \setminus R^{cover}$
  $update\_workload(p^{best}, R^{cover})$
  $Participants \leftarrow Participants \cup \{p^{best}\}$
end while

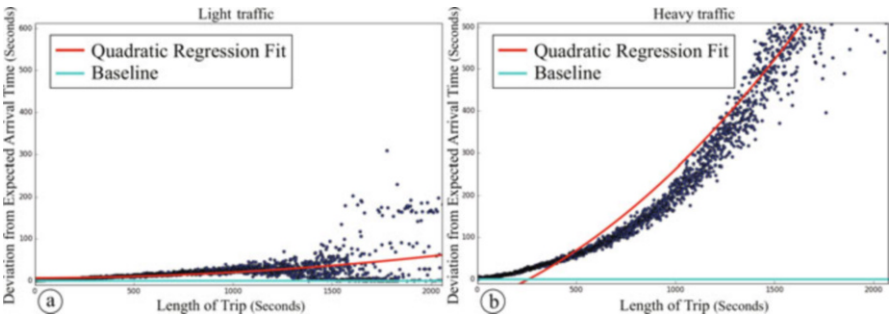

Fig. 6 The y axis measures the difference in time from a very light traffic expected arrival time. We can see deviation from arrival time increases at different rate based on traffic

scheduling window to 300 s for each sensing task, since errors for predicted arrival times within trip length of 300 s are acceptable in both light and heavy traffic as shown in Fig. 6. However, a static window size suffers from too frequent scheduling, resulting computation resources waste and increased delays. For instance, in light traffic, a 1200 s may be better choice for the scheduling window instead of 300 s. Thus, we proposed Dynamic Window Based (DWB) solution.

Algorithm 2 Dynamic window based scheduling (DWB)

1: $\mathscr{P} \leftarrow$ Get participant candidates
2: $\mathscr{C} \leftarrow$ Subscribe to the request buffer
3: **for each** $c \in \mathscr{C}$ **do**
4: $\left\{R_{c'} \in R_c \mid \forall_{r_k}\ \mathscr{T}(r_k, R_c) < current\_time + SW_c\right\}$ ▷ SW is calculated based on Eq. 24
5: $S \leftarrow WBG(R_{c'},\ \mathscr{P})$ ▷ The Weight Based Greedy Algorithm (see Algorithm 1)
6: $Expected\_cover_c \leftarrow |QRC(S,\ c)|$
7: Notify the client c and each $p \in S$
8: **end for**

As shown in Algorithm 2, DWB only schedule participants for serving a client's sensing task within a specific scheduling window SW. The size of SW is calculate as the following:

$$SW_c \leftarrow \begin{cases} cover\_rate < 1 & \text{MAX} \begin{cases} SW_c \times cover\_rate_c \\ SW_{min} \end{cases} \\ \text{otherwise} & SW_c \times \theta \ \text{ where } \theta \in \mathbb{R}^+ \end{cases} \tag{24}$$

θ is a positive multiplier controls how fast SW_c is grow. Note that to predict the next SW size, we need to know the coverage performance of the previous SW. In such case, we assume that the client calculated the coverage rate before submit the next rescheduling request, where the coverage rate is calculated as the follow:

$$cover\_rate_c \leftarrow \frac{Actual\_cover_c}{Expected\_cover_c} \tag{25}$$

Each client has its own scheduling window due to various of driving behavior. The scheduler continuously adjust the size of SW until the corresponding sensing routes are fully scheduled. The flow of DWB scheduling is shown in Fig. 7.

4.5 Experiment Setup

To evaluate the performance of our proposed algorithm, we utilized the TAPAS Cologne dataset, one of the largest traffic simulation dataset. It consists of data simulated over a 24 h period for the city of Cologne, Germany. It covers over 1000 square kilometers[19]. To reduce simulation time, we restricted ourselves to a data subset which consist of 7200 time steps from 6 AM to 8 AM. During this period about 34,000 unique vehicles were part of the simulation. To obtain the trace of vehicle positions as predicted path, we utilized the SUMO traffic simulator [13]. Figure 8 shows the workflow of our experiment. For simplicity, we assume all the PVC clients are drivers, that is, we randomly select a subset of vehicles as clients from the 34,000 unique vehicles. We utilize a uniform random distribution to

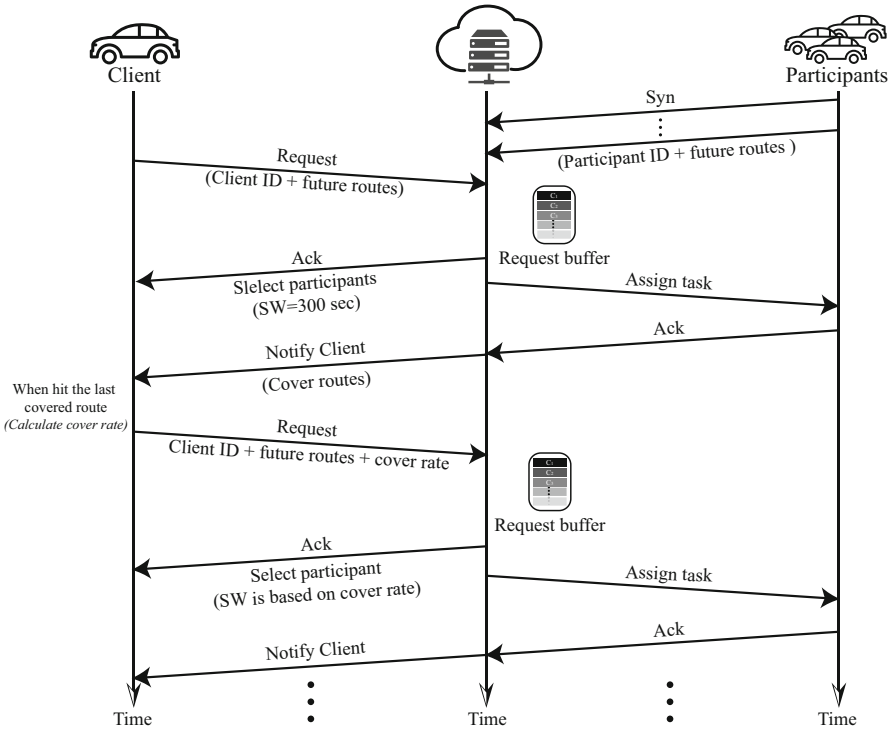

Fig. 7 Dynamic window based scheduling workflow

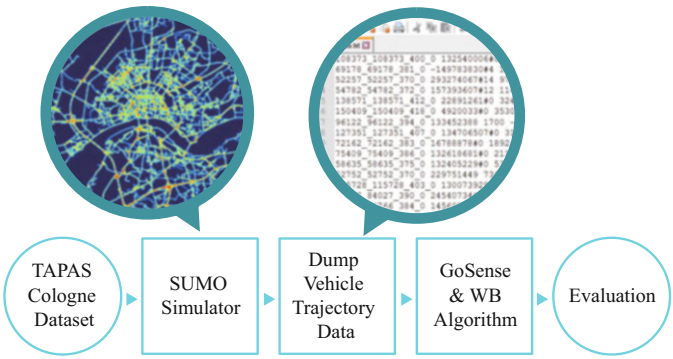

Fig. 8 Experiment flow

select 30,000 clients whereby each client's sensing route range from 1 to 125 road segments. When clients begin their journey, client send their future sensing routes and sensing objectives to the server. The system then schedule participants to cover clients' sensing routes in a first-in-first-serve manner. A vehicular client cannot be the participant for its own task but can be a participant for other tasks. For parameter

setting, we set the budget to be constrained by $\beta_c = 20$, and we set the soft parallel task constraint for each vehicle to be $cap = 10$. We evaluate the performance of the proposed algorithm by comparing against the PR algorithm in GoSense [22]. The simulation programs were written in Java and utilized only a single thread. When evaluating the run time of the algorithms, each algorithm was run alone. The machine utilized has i5-4590 k 3.30 GHZ and 16 GB of RAM.

4.6 Main Results

In this subsection, we compare the performance of GoSense and our proposed weight based algorithm. We utilize the number of participants required for completing a task and maximum peak workload among all participants as metrics to evaluate the algorithms. The peak workload is defined as follows:

$$Peak\ workload \leftarrow \arg\max_{p' \in P} \ell oad_{p'}^{max}$$

As shown in Fig. 9, WBG outperforms GoSense in terms of load balance. We also evaluated how data timeliness constraints δ and ε influenced the performance. The size of the window is defined as:

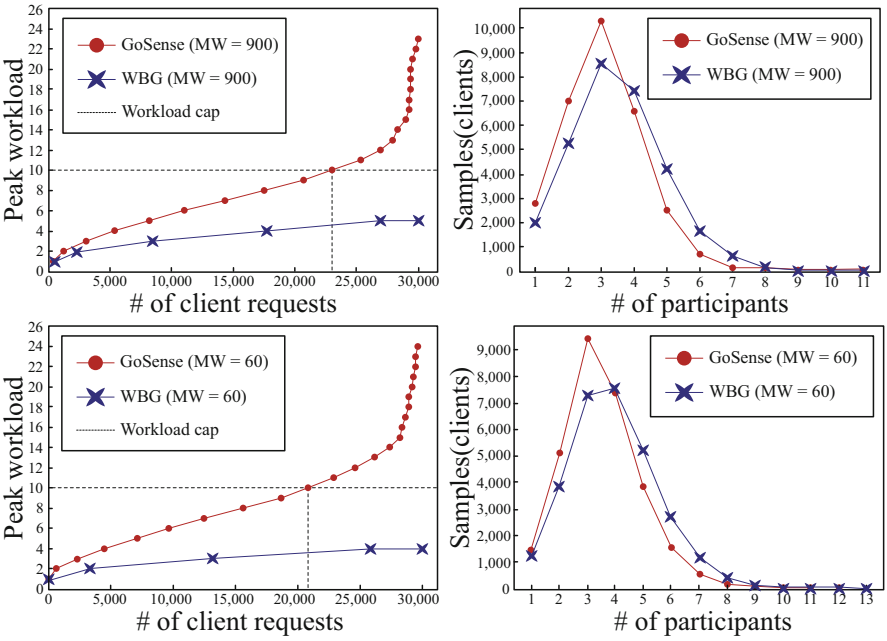

Fig. 9 Comparison of GoSense and WBG algorithm performance. Note: MW←monitoring window (second)

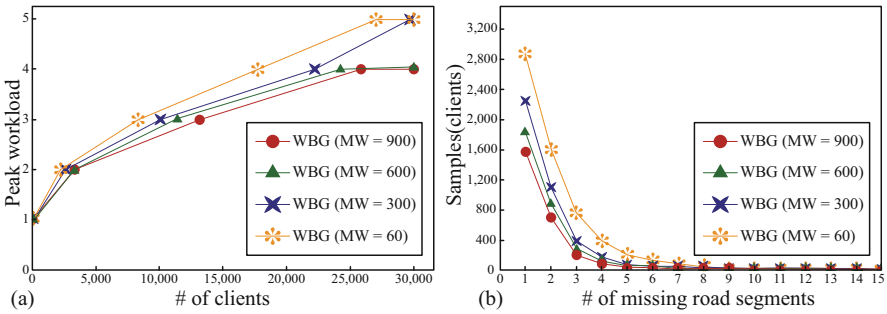

Fig. 10 Simulation results under different monitoring window (MW): second

$$MW \leftarrow \varepsilon - \delta$$

Given different sizes of MW, WBG still outperformed GoSense. We also observe that GoSense requires slightly less participants but also causes significantly heavier peak workload in comparison to WBG. Thus, we can trade off increase in participants for better load balancing.

Figure 10 depicts how MW influence the performance. As we can see in Fig. 10a, wider MW will result in smaller peak workload. That is, because wider MW will yield more potential participants for selection as shown in Fig. 3c; the workload can be more easily spread between participants. We also found that the overall number participants decreased when the size of monitoring window is decreased as shown in Fig. 9. This is because when the size of MW is small there is high chance that the region does not contains the suitable participants which result in uncovered region. As depicted in Fig. 10b, we can see that as MW narrows, the number of missing route segment covered increases. This is because the time interval which a participant is considered valid is reduced. Thus at every time step, less participants are considered.

4.7 The Execution Time

Next, we consider the execution time of both algorithms under the same conditions. We explore run time under different number of participants and different road length. Here, number of participants refers to the vehicles which can be selected for PVC tasks. Hence, a road length of 50 would require coverage in 50 different road segments. In general, we can see that as the number of participants or road length increases, run time increases linearly. This matches well with our complexity of $O(S \times R \times R_p)$. As we can see from Fig. 11a, b, when considering the number of participants, the run time of WBG is only slightly above GoSense. In contrast, the

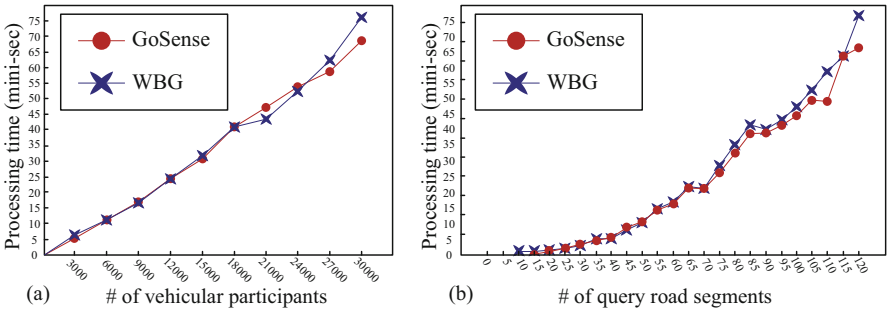

Fig. 11 Running time under different number of participants and length of request route segments

number of query routes seems to show no impact between either algorithm. Hence, we have shown that utilizing WBG does not result in a heavier overhead compared to GoSense.

4.8 Window Based Scheduling (Static vs. Dynamic)

In this section, we explore the improved WBG algorithm using window based scheduling. When experimenting in this section, we utilized a simple position prediction algorithm using average route speed. Hence, significant errors exists when predicting for a longer time periods. Figure 12 shows the performance in terms of route segments missing rate in Cumulative Distribution Function (CDF) graph format. Here, route segment missing rate is defined as $route\ missing\ rate_c \Leftarrow 1 - cover\_rate_c$. As we can see, scheduling for whole route leads to the worst performance. Using 80% of vehicles as comparison point, scheduling for whole results in 20% and 40% route missing rate compared to 5% and 20% for static window scheduling and 10% and 20% for dynamic window scheduling. Overall, static window scheduling performs best for missing route rate. There is a trade off; static window scheduling requires fairly frequent scheduling and increase computational overhead. Thus, if we care about reducing the number of scheduling, dynamic window scheduling reduces the scheduling count as seen in Fig. 13. Furthermore, the performance of dynamic window scheduling remains within reach of static scheduling with more frequency scheduling. Thus, we have shown dynamic window scheduling can be a competitive option when considering heavy overhead of scheduling for all vehicles.

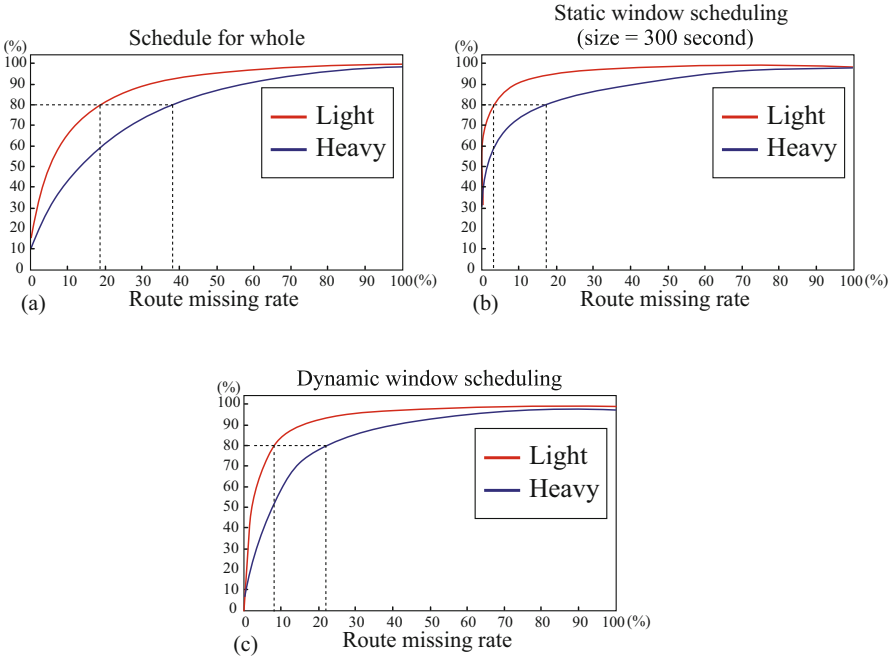

Fig. 12 Results show in Cumulative Distribution Function (CDF) graph under random monitoring window. The random number is generated using the same seed. The route missing rate = number of uncover route segments R_c^{miss} divided by the total query route segments R_c. For calculating SW, we set $\theta = 2$ and $SW_{min} = 300$ for our dynamic window scheduling

5 Future Works

Previous works have proposed various methods and metrics for selecting sensing participants when the trajectory has some level of certainty. However, such sensing are considered opportunistic; no solution proposed so far seeks to actively suggest new routes to the participants to minimize sensing coverage. For instance, if a participant's current trajectory collects information that is already known; that participant would make a small overall contribution. However, by taking a detour, the participant could greatly increase the contribution to the sensing task. The advantages are unmistakable; by rerouting participants, areas without possibility of sensing in the old model can be sensed. Overall the sensing quality can be improved. However, significant challenges lie ahead. First, a reward system must be designed to incentivize users to tolerate rerouting while minimizing costs. Unlike participants selected to sense opportunistically, rerouting consumes a participant's time and requires direct participation. Second, a new route generation algorithm must be devised to suggest routes which increase overall sensing coverage to potential participants while minimizing overlap and costs. Finally, a complete vehicular

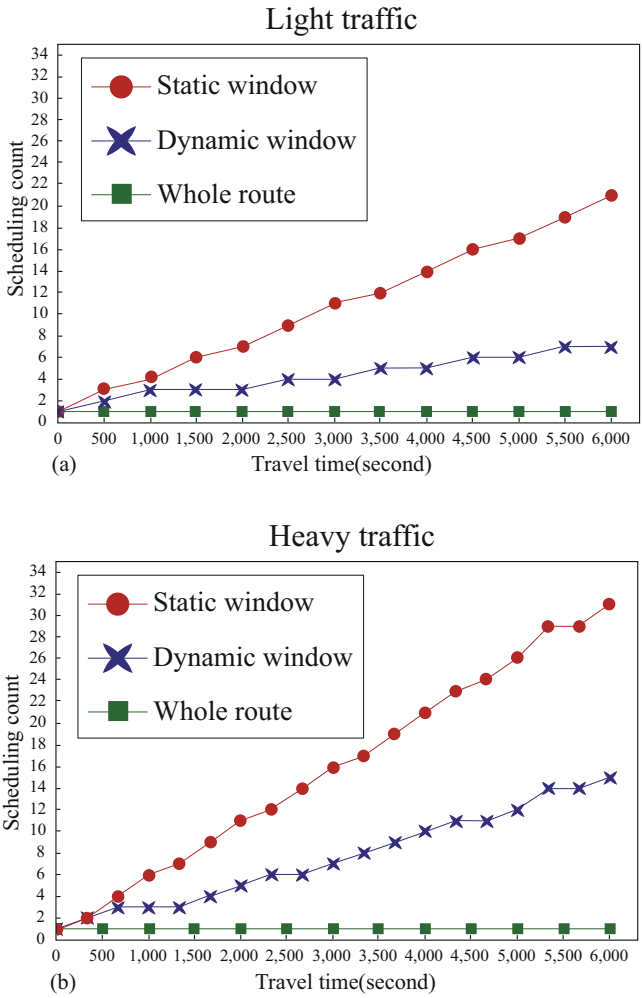

Fig. 13 Scheduling count difference under light and heavy traffic

crowdsensing system should combine both opportunistic sensing and participatory sensing. How to seamlessly incorporate both types of sensing will prove to be an intriguing challenge.

6 Conclusion

Vehicular crowdsensing is certain to play a major role in sensing data collection for proactive services in smart cities. Because vehicles are highly mobile, each vehicle can quickly gather sensing information over large regions. However, challenges

remain; current network infrastructure and vehicular networking technologies cannot support large-scale vehicular crowdsensing due to its bandwidth requirements. Luckily, it is unnecessary to select all vehicles as participants; sensor coverage intersects and produce duplicated data. Thus, we can minimize bandwidth use by selecting the best participants from the available ones. The selection criterion can be diverse; this includes spatio-temporal coverage, distance to point of interest, or even combined coverage of heterogeneous sensors. Thus, the selection problem consist of maximizing these measures while minimizing or staying within a budget limit. In personalized vehicular crowdsensing, load balance criteria is also included because of local processing needs. However, work still remain in the area of participatory vehicular crowdsensing; actively suggesting routes to participants for sensing can significantly improve sensing coverage.

References

1. Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper, Mar 2017.
2. S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392, 2014.
3. L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from https://crawdad.org/roma/taxi/20140717, July 2014.
4. M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.
5. C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan. A comparative survey of vanet clustering techniques. *IEEE Communications Surveys & Tutorials*, 19(1):657–681, 2017.
6. Y. Gao, W. Dong, K. Guo, X. Liu, Y. Chen, X. Liu, J. Bu, and C. Chen. Mosaic: A low-cost mobile sensing system for urban air quality monitoring. In *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pages 1–9. IEEE, 2016.
7. Google. Waze mobile, 2017. https://www.waze.com/.
8. S. A. Hamid, H. Abouzeid, H. S. Hassanein, and G. Takahara. Optimal recruitment of smart vehicles for reputation-aware public sensing. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 3160–3165. IEEE, 2014.
9. K. Han, C. Chen, Q. Zhao, and X. Guan. Trajectory-based node selection scheme in vehicular crowdsensing. In *Communications in China (ICCC), 2015 IEEE/CIC International Conference on*, pages 1–6. IEEE, 2015.
10. Z. He, J. Cao, and X. Liu. High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2542–2550. IEEE, 2015.
11. C. Hu, M. Xiao, L. Huang, and G. Gao. Truthful incentive mechanism for vehicle-based nondeterministic crowdsensing. In *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*, pages 1–10. IEEE, 2016.
12. M. Hu, Z. Zhong, Y. Niu, and M. Ni. Duration-variable participant recruitment for urban crowdsourcing with indeterministic trajectories. *IEEE Transactions on Vehicular Technology*, 2017.
13. D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

14. Y. Liu, J. Niu, and X. Liu. Comprehensive tempo-spatial data collection in crowd sensing using a heterogeneous sensing vehicle selection method. *Personal and Ubiquitous Computing*, 20(3):397–411, 2016.
15. D. Peng, F. Wu, and G. Chen. Pay as how well you do: A quality based incentive mechanism for crowdsensing. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 177–186. ACM, 2015.
16. S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *International Conference on Pervasive Computing*, pages 138–155. Springer, 2010.
17. L. Shao, C. Wang, Z. Li, and C. Jiang. Traffic condition estimation using vehicular crowdsensing data. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, Dec 2015.
18. S. Ucar, S. C. Ergen, and O. Ozkasap. Vmasc: Vehicular multi-hop algorithm for stable clustering in vehicular ad hoc networks. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 2381–2386. IEEE, 2013.
19. S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075, 2014.
20. M. Wu, D. Ye, S. Tang, and R. Yu. Collaborative vehicle sensing in bus networks: A stackelberg game approach. In *Communications in China (ICCC), 2016 IEEE/CIC International Conference on*, pages 1–6. IEEE, 2016.
21. K. Yi, R. Du, L. Liu, Q. Chen, and K. Gao. Fast participant recruitment algorithm for large-scale vehicle-based mobile crowd sensing. *Pervasive and Mobile Computing*, 2017.
22. T. Y. Yu, X. Zhu, and H. Chen. Gosense: Efficient vehicle selection for user defined vehicular crowdsensing. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5, June 2017.
23. X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016.
24. X. Zhang, Z. Yang, Z. Zhou, H. Cai, L. Chen, and X. Li. Free market of crowdsourcing: Incentive mechanism design for mobile sensing. *IEEE transactions on parallel and distributed systems*, 25(12):3190–3200, 2014.
25. D. Zhao, H. Ma, L. Liu, and X.-Y. Li. Opportunistic coverage for urban vehicular sensing. *Computer Communications*, 60:71–85, 2015.