Online Participatory Sensing in Double Auction Environment with Location Information

Jaya Mukhopadhyay, Vikash Kumar Singh, Sajal Mukhopadhyay, Anita Pal

Abstract—As mobile devices have been ubiquitous, participatory sensing emerges as a powerful tool to solve many contemporary real life problems. Here, we contemplate the participatory sensing in online double auction environment by considering the location information of the participating agents. In this paper we propose a truthful mechanism in this setting and the mechanism also satisfies the other economic properties such as budget balance and individual rationality.

Index Terms—Participatory sensing, location information, online double auction.

I. INTRODUCTION

PARTICIPATORY SENSING [8][13] is a distributed problem solving model in which the common people (may not be professionals) of indefinite size carrying smart devices (such as Tablets, smart watches, smartphones, etc.) may be engaged to accomplish the tasks or sub-tasks. Examples include measuring the level of smoke and toxic gases present in the environment of certain industrial areas [20][17], keeping track of automobiles traffic condition in highly populated urban areas [12][10], monitoring the state of the roads (e.g., potholes, bumps, etc.) by attaching sensors to cars [18], and several directions in healthcare and physical fitness [21]. In a typical participatory sensing model, there exists three different participating community namely; (a) task requester(s), (b) platform (or third party), and (c) task executer(s). The working of participatory sensing system is initiated by the task requester(s) who submit their sensory task(s) to the platform that is/are to be accomplished by the common people incorporated with smart devices. Once the third party (or platform) receives the task(s), he/she (henceforth he) outsource the task(s) or subtask(s) to the group of common people carrying smart devices. In the participatory sensing terminology, these common people carrying smart devices are termed as task executers. In order to complete the assigned task(s), the task executers have to utilize their owned smart device resources (such as battery, GPS system, etc.). Now, the obvious question that may arise is: how to motivate the task executers to accomplish the projected task(s) voluntarily by utilizing their resources. Moreover, it is to be noted that each of the task requesters and the task executers (synonymously called agents) are strategic. In general, strategic means that, the agents chooses their strategies so as to maximize a well defined individualistic utility.

Fig. 1. System model

Answering to above posed question, for motivating the large group of task executers for voluntary participation, one can think of the solution to incentivize the participating task executers by some means, once the task(s) is/are completed. In this paper, we study a single task execution problem (STEP); where there are multiple task requesters having a single common task, that is to be accomplished by the multiple task executers in an online environment. By online environment, we mean that the agents arrives in the system and departs from the system on a regular basis. The proposed model is shown in Fig. 1. The novelty that is introduced in this is to develop a game theoretic approach to model the STEP. As their are multiple task executers and multiple task requesters, this give rise to a double auction framework. In our model the location information of the agents are considered so as to cover a substantial area albeit collecting to much of redundant data. It is to be noted that the task executers location information are tracked implicitly during the supply of the completed tasks to the third party. By doing so it is guaranteed that the task executers can’t gain by lying their actual location. The location aware participatory sensing was first introduced in [11]. However location aware participatory sensing in online double auction environment was not addressed in [11]. In this paper we have addressed the location aware participatory sensing in online DA. To avoid collecting redundant data, clustering concept is implemented before running the auction in each round.

The main contributions of this paper are:

- In the participatory sensing scenario, we have proposed a framework to study the STEP with multiple task executers
and multiple task requesters in an online environment by utilizing the concept of clustering along-with auction. As their are multiple task requesters and multiple task executors, it is a good choice to model the participatory sensing scenario using double auction.

- We propose a single task execution mechanism (STEM) for STEP motivated by [19]4 that takes into account multiple task executors and multiple task requesters. We design a truthful (or incentive compatible) mechanism for this interesting class of problem.

- We have shown that STEM is bounded above by $O(km^2)$. Moreover, we have also shown that our STEM satisfies the several economic properties such as truthfulness, individual rationality, and Budget balance.

- A substantial amount of simulation is done to compare STEM with the carefully designed benchmark scheme (McAfee’s rule).

- We have proved that in the given online environment with clustering the agents can’t gain by manipulating their valuation, arrival and departure time in a given arrival-departure window.

The remainder of the paper is structured as follows. Section II elucidates the preliminary concepts about participatory sensing. Section III describes our proposed model. The proposed mechanisms is illustrated in section IV. The paper is concluded with the possible future directions in section V.

II. Prior works

Recently, there has been a spate of research work at the border of participatory sensing and in their several applications areas. In this section we discuss the prior works on participatory sensing, taking into account incentives aspects, quality of data or information supplied, privacy of the task executors performing the task, and different set-up participatory sensing in budget constraint environment.

In order to get a nice overview of the participatory sensing the readers may refer [22][8][13][7][5]. Currently, the participatory sensing is one of the open research areas. One obvious question that arise in the participatory sensing environment is: how to motivate the large common people carrying smart devices to participate in the system? To answer this question in a better way, the researchers have provided their immense effort in this direction. In past, for voluntary participation of the task executors several incetivizing schemes are discussed in literature. [21] follows the fixed price payment scheme, where the winning agents are paid a fixed price as their payment. However, the fixed price based incentive scheme may not satisfy the several participating agents because of the amount of effort they make in the data collection process. Moreover, the incentive based schemes has got a special attention from the research community. [16] addresses the incentive scheme under the reverse auction based setting (single buyer and multiple sellers). Several incentive schemes has been introduced in [9][14][25]. In [3][24][23] efforts have been made by the researchers to show the effect of quality of data collected by the agents to the overall system by incorporating the quality of data to the system in some sense. Some initial research has been carried out by [8][23][15][6] to preserve the privacy of the agents so that their private information associated with the data are not revealed publicly. Recently, [11] provides the incentive schemes under the location constraints. in their work they have addressed location aware participatory sensing in one buyer and multiple seller environment. In our model we have explored more general multiple sellers-multiple buyers framework in more challenging location aware participatory sensing in online double auction environment.

III. System model

In this section, considering an online environment we formalize a single task execution problem (STEP) for the participatory sensing scenario. By online environment, we mean that the agents arrives in the auction market and departs from the auction market on a regular basis in a given time horizon T (say a day). Let $B = \{B_1, B_2, ..., B_m\}$ be the set of task requesters and $S = \{S_1, S_2, ..., S_n\}$ be the set of task executors such that $m << n$. The task executors and task requesters are synonymously called agents. Each of the task executor $S_i$ incurs a private cost for performing the available task termed as valuation given as $v_i^r$. The set $v_i^r$ denotes the set of valuations of all the task executors given as $v_i^r = \{v_i^r, v_{i2}^r, ..., v_{in}^r\}$. Similar to the task executors, each of the task requester $B_i$ has some private value for buying the task after its completion and is given as $v_i^e$. The set $v_i^e$ denotes the set of valuations of all task requesters given as $v_i^e = \{v_{i1}^e, v_{i2}^e, ..., v_{in}^e\}$. Each of the task executors and task requesters places their private information in a sealed bid manner. It is to be noted that, due to the strategic nature of the agents, they can mis-report their respective private values. So, it is convenient to represent the cost reported for performing the task by the task executor $S_i$ as $v_i^r$ and the value of task requester $B_i$ for buying the task as $v_i^e$. $v_i^r = v_{i1}^r$ and $v_i^e = v_{i1}^e$ describes the fact that $S_i$ and $B_i$ are not deviating from their true valuations. In this model, there are multiple task requesters (as buyers) and multiple task executors (as sellers). So, this is a perfect setting to model the STEP as an online double auction problem (ODAP). Due to online nature of the STEP, one of the realistic parameters that is perceived in our proposed model is arrival and departure time of the agents. The arrival time of any agent is the time at which he/she (henceforth he) knows about the auction market or the time at which he first become aware of his desire to involve into the auction market after entering into the system. The arrival time of each task executor $S_i$ and each task requester $B_i$ are given as $a_i^r$ and $a_i^e$ respectively. For a task requester, we interpret the departure time as the final time in which he values the task. For a task executor, the departure time is the final time in which he is willing to accept payment. The departure time of each task executor $S_i$ and each task requester $B_i$ are given as $d_i^r$ and $d_i^e$ respectively. The agents may mis-report their respective arrival time or departure time or both within the arrival-departure window in order to gain. It is convenient to represent the arrival time of task executor $S_i$ and task requester $B_i$ as $a_i^r$ and $a_i^e$ respectively. Similarly, more conveniently the departure time of task executor $S_i$ and
task requester $B_i$ is $d_i^c$ and $d_i^f$ respectively. $\alpha_i^c = \alpha_i^f = \alpha_i^c$, $\beta_i^c = \beta_i^f$, and $\gamma_i^c = \gamma_i^f$ describes the fact that $S_i$ and $B_i$ are not misreporting their arrival and departure time. In our proposed model, a day is termed as time horizon $T$. The time horizon $T$ is partitioned into several time slots (not necessarily of same length) given as $T = \{\tau_1, \tau_2, \ldots, \tau_r\}$. For each time slot $\tau_i$, a new set of active task requesters $R \subset B$ and a new set of active task executers $U \subset S$ arrives in the auction market. At each time slot $\tau_i$, considering the newly active task executers $U$, a set of clusters of task executers are formed and is given as: $E = \{E_1, E_2, \ldots, E_k\}$, where $E_j$ is termed as the $j^{th}$ cluster for $\tau_i$ time slot. Over the $T$ time horizon the cluster vector can be given as: $E = \{E^1, E^2, \ldots, E^n\}$. Once the clusters are formed, then for each cluster $E_j$ several independent double auction will be performed. At each time slot $\tau_i \in T$ and from each cluster $E_j$, the request of winning task requesters-task executers are paired. At each time slot $\tau_i$, our proposed mechanism matches one task executor to one task requester in a cluster. More formally, a mechanism $M = (A, P)$, where, $A$ is called an allocation function and $P$ is called a payment function. The allocation function $A$ maps the pair of task executers valuation and task requesters valuation to the possible task executor-task requester pairs. Following the payment function, the payment of each task executor $S_i$ and each task requester $B_i$ is given as $P^{x_i}$ and $P^{y_i}$ respectively. As the task executers and task requesters are strategic in nature, they will try to maximize their utility. The utility of any task executor is defined as the difference between the payment received by the task executor and the true valuation of the task executor. More formally, the utility of $S_i$ is $\phi_i^c = P^{x_i} - \gamma_i^c$, if $S_i$ wins otherwise 0. Similarly, the utility of any task requester is defined as the difference between the true valuation of the task requesters and the payment he pays. More formally, the utility of $B_i$ is $\phi_i^c = \gamma_i^c - P^{y_i}$, if $B_i$ wins otherwise 0.

IV. STEM: PROPOSED MECHANISM

A. Outline of STEM

In order to present the brief idea of the STEM to the readers, the outline of the STEM is discussed before going into the detailed view. The outline of the STEM can be thought of as a three stage process:

- For any auction round $t \in T$ find out the active task executers and task requesters.
- Cluster the active task executers based on k-means clustering technique.
- Run the online double auction separately for each cluster of task executers. Task requesters will be the same for all the clusters.

B. Sketch of the STEM

The three stage STEM can further be studied under four different sections: Main routine, Cluster formation, Payment, and Allocation. First, the sub-part of the proposed mechanism i.e. the Main routine phase is discussed and presented. The Cluster formation phase is addressed next. Next, the crucial part of the proposed mechanism i.e. payment phase motivated by [3] is discussed and presented. Finally, one of the allocation phase is addressed.

1) Main routine: The idea lies behind the construction of Main routine is to handle the system partitioned into different time slots $\tau_i \in T$. The input to the Main routine are the set of task executers at $\tau_i$ time slot i.e. $S_{\tau_i}$, the set of available task requesters at $\tau_i$ time slot i.e. $B_{\tau_i}$, the overall time horizon i.e. $T$, the set of cost of execution of all task executers i.e. $\phi^c$, and the set of value for buying the executed tasks by all the task requesters i.e. $\phi^f$. The output is the set of allocation vector $A$.

In line 2, the several data structures that are utilized in main routine are set to $\phi$. The for loop in line 3 iterates over all the time slots $\tau_i \in T$. In line 4, the active $TE()$ function returns the set of active task executers at time slot $\tau_i$ and is held in $U$ data structure. Whereas, the set of active task requesters at any time slot $\tau_i$ is determined by the function active $TR()$ and is held in $R$ data structure. The for loop in line 6–8 iterates over the set of active task executers $U$ and keeps track of costs of the members in set $U$ in $\phi^c$ data structure. Similarly, the for loop in line 9–11 iterates over the set of active task requesters $R$ and keeps track of values of the members in set $R$ in $\phi^c$ data structure.

Algorithm 1 Main routine($S_{\tau_i}, B_{\tau_i}, T, \phi^c, \phi^f$)

Output: $A \leftarrow \{A_1, A_2, \ldots, A_k\}$

1: begin
2: $U \leftarrow \phi, R \leftarrow \phi, \gamma^c \leftarrow \phi, \gamma^f \leftarrow \phi, \phi^c \leftarrow \phi$
3: for all $\tau_i \in T$ do
4: $U \leftarrow$ active $TE$($S_{\tau_i}, \tau_i$) $\triangleright \forall S_i \in U$, $\hat{\alpha}_i^c \leq \tau_i < \hat{\alpha}_i^c$
5: $R \leftarrow$ active $TR$($B_{\tau_i}, \tau_i$) $\triangleright \forall B_i \in R$, $\hat{\alpha}_i^c \leq \tau_i < \hat{\alpha}_i^c$
6: for each $S_i \in U$ do
7: $\gamma^c \leftarrow \gamma^c \cup S_i \cdot \hat{\alpha}_i^c$ $\triangleright \hat{\alpha}_i^c$ is the valuation component of $S_i$
8: end for
9: for each $B_i \in R$ do
10: $\gamma^f \leftarrow \gamma^f \cup B_i \cdot \hat{\alpha}_i^f$ $\triangleright \hat{\alpha}_i^f$ is the valuation component of $B_i$
11: end for
12: $E_i \leftarrow$ Cluster formation($U, k$)
13: for each $E_i \in E$ do
14: $U_i \leftarrow \hat{S}_{Sort\_ascend}(E_i, S_i \cdot \gamma^c)$ $\triangleright$ Sorting based on $\gamma_i^c$ for all $S_i \in E_i$
15: $R_i \leftarrow \hat{S}_{Sort\_descend}(R, B_i \cdot \gamma^f)$ $\triangleright$ Sorting based on $\gamma_i^f$ for all $B_i \in R$
16: Payment($U_i, R_i$)
17: $U_i^{(j)} \leftarrow U_i^{(j)} \cup U_i$
18: $R_i^{(j)} \leftarrow R_i^{(j)} \cup R_i$
19: $\hat{\alpha}_i^c \leftarrow \hat{\alpha}_i^c \cup \hat{\alpha}_i^c$
20: $\hat{\alpha}_i^f \leftarrow \hat{\alpha}_i^f \cup \hat{\alpha}_i^f$
21: $U_i \leftarrow \phi$
22: end for
23: $\gamma^c \leftarrow \phi, \gamma^f \leftarrow \phi$
24: New task executers and task requesters comes.
25: $S_{\tau_i} \leftarrow \hat{S}_{\{new\ task\ executers\}}$
26: $B_{\tau_i} \leftarrow \hat{S}_{\{new\ task\ requesters\}}$
27: end for
28: return $A$
29: end

In line 12, a call to Cluster formation($U, k$) is made; where
$k$ is the number of clusters to be formed. Once in a particular time slot $\tau$, the cluster set $\mathcal{E}^i$ is formed in line 12, the payment and based on the payment the allocation is determined which is captured by the for loop in line 13-22. In line 14 and 15 the task executers and task requesters are sorted in ascending and descending order respectively. In line 16, a call to payment phase is done. In line 17 and 18 all the active task executers and task requesters at time $\tau_j$ in $j^{th}$ cluster which are not paired are placed in $\mathcal{U}^{(j)}_{\mathcal{E}}$ and in $\mathcal{R}^{(j)}_{\mathcal{E}}$ respectively. The $\mathcal{E}^i$ and $\mathcal{L}^j$ data structures keeps track of all the active task executers and task requesters in a given time slot $\tau$, but not allocated in there respective clusters. In line 21, the $\mathcal{U}_c$ data structure is set to $\phi$. The data structure $\gamma^e$ and $\gamma^r$ are set to $\phi$. Now, the new task executers and new task requesters are arriving in the market for the next time slot as depicted in line 24. In line 25-26 $S_r$ and $B_r$ captures the set of all task executers and task requesters that are going to participate in the next time slot. Line 28 returns the final allocation set $\mathcal{A}$.

2) Cluster formation: The input to the Cluster formation are the set of active task executers at any time slot $\tau$, given as $\mathcal{U}$, and the number of cluster to be formed i.e. $k$. Considering the centroid determination phase, Line 2 initializes the $C$ data structure utilized in the Cluster formation algorithm. The random() function in line 4 randomly picks a point as a centroid from the available point set $X$. The randomly selected centroid is placed in $C$ data structure using line 5.

Algorithm 2 Cluster formation ($\mathcal{U}, k$)

1:  begin  
2:  $C \leftarrow \phi$  \quad k centroid determination  
3: while $|C| \neq k$ do  
4:   $x^i \leftarrow \text{random}(X)$ \quad Picking a random point $X_i \in X$  
5:  $C \leftarrow C \cup \{x^i\}$  
6: end while  
7: repeat \quad k cluster formation  
8:   $\mathcal{E}^i \leftarrow \phi$, $\mathcal{L}^j \leftarrow \phi$  
9:   for each $S_k \in \mathcal{U}$ do  
10:    for each $X_j \in C$ do  
11:      $D' \leftarrow D' \cup \{D(S_k, X_j)\}$ \quad Distance between $S_k$ and $X_j$  
12:    end for  
13:  $j^* \leftarrow \arg\min_j D'$  
14:  $\mathcal{L}^j \leftarrow \mathcal{L}^j \cup \{S_k\}$  
15: end for  
16: $C \leftarrow \phi$  
17: for $j = 1$ to $k$ do  
18:   $\mathcal{E}^i \leftarrow \mathcal{E}^i \cup \{\mathcal{L}^j\}$  
19: end for  
20: for each $\mathcal{E}^i \in \mathcal{E}^i$ do  
21:   $x^i_{\ell} = \frac{1}{|\mathcal{E}^i|} \sum_{x^i \in \mathcal{E}^i} x^i$ \quad $x^i_{\ell}$ is the point $\ell$ i.e. a two dimensional vector in cluster $\mathcal{E}^i$  
22:  $C \leftarrow C \cup X^i_{\ell}$  
23: end for  
24: until change in cluster takes place  
25: return $\mathcal{E}^i$  
26: end

In line 3 the while loop ensures that the loop terminates when the size of $k$-centroid are determined. Line 8 initializes the $\mathcal{E}^i_j \leftarrow \phi$ and $\mathcal{L}^j \leftarrow \phi$. The for loop in line 9 iterates over the active set of task executers $\mathcal{U}$. Line 10-12 determines the closest distance of each $S_k$ to a centroid $X_j$. The $\mathcal{L}^j$, in line 14 keeps track of each $S_k$. In line 16, the $C$ data structure is set to $\phi$. The $\mathcal{E}^i$ data structure keeps track of all the clusters formed in a particular time slot $\tau$, as depicted in line 17 - 19. Line 20 - 23 determines the new centroids. The procedure in line 7-24 will be repeated until the change in the cluster is seen. Line 25 returns the set of cluster in any particular time slot $\tau$.

3) Payment: The input to the payment phase are the set of winning task executers i.e. $\mathcal{U}_e$, and the set of winning task requesters i.e. $\mathcal{R}$. In line 2 the $\mathcal{U}$ and $\mathcal{R}$ data structure are set to $\phi$. The for loop in line 3-15 keeps track of payment of the winning task executers. The check in line 3 confirms that if the task executor belongs to freshly arrived category then the payment is decided by line 5 of the Algorithm 4 otherwise the payment is made using line 7. Now, the check in line 9 is done to guarantee that the payment made to any task executor $S_k$ is greater than its cost for executing the task i.e. satisfying the important economic property individual rationality.

Algorithm 3 Payment ($\mathcal{U}_e, \mathcal{R}$)

1: begin  
2: $\mathcal{U} \leftarrow \phi$, $\mathcal{R} \leftarrow \phi$ \quad Fresh arrival  
3: for each $S_k \in \mathcal{U}_e$ do  
4: if $\hat{\alpha}^e \equiv \tau$, then  
5:   $\chi^e_{\ell} \leftarrow \min_{\tau_j \in \left[c-k, \tau\right]} \{P^e_j(\tau_j)\}$ \quad Still active  
6: else  
7:   $\chi^e_{\ell} \leftarrow \min\{P^e_j(\tau_j)\}$  
8: end if  
9: if $\chi^e_{\ell} \geq \hat{\delta}^e_{\ell}$ then  
10:   $P^e_{\ell} \leftarrow P^e_{\ell} \cup \{\chi^e_{\ell}\}$  
11:   $\mathcal{U} \leftarrow \mathcal{U} \cup \{S_k\}$  
12: else:  
13:   $S_k$ is priced out.  
14: end if  
15: end for  
16: for each $B_i \in \mathcal{R}$ do  
17: if $\tilde{\alpha}^r \equiv \tau$, then  
18:   $\chi^r_{\ell} \leftarrow \max_{\tau_j \in \left[c-k, \tau\right]} \{P^r_j(\tau_j)\}$ \quad Fresh arrival  
19: else  
20:   $\chi^r_{\ell} \leftarrow \max\{P^r_j(\tau_j)\}$  
21: end if  
22: if $\chi^r_{\ell} \leq \tilde{\delta}^r_{\ell}$ then  
23:   $P^r_{\ell} \leftarrow P^r_{\ell} \cup \{\chi^r_{\ell}\}$  
24:   $\mathcal{R} \leftarrow \mathcal{R} \cup \{B_i\}$  
25: else:  
26:   $B_i$ is priced out.  
27: end if  
28: end for  
29: Allocation($\mathcal{U}, \mathcal{R}, \gamma^e, \gamma^r$)  
30: end

The $P_e$ data structure in line 10 keeps track of all the payment of all the winning task executers satisfying the
individual rationality. If the condition in line 9 is not satisfied by the task executers, then the winning task executers are priced out of the market as depicted in line 13. The for loop in line 16-28 keeps track of payment of the winning task requesters. The check in line 17 confirms that if the task requester belongs to freshly arrived category then the payment is decided by line 17 otherwise the payment is made using line 19. Now, the check in line 21 is done to guarantee that the payment made by any task requester \( S_i \) is no more than its value for buying the completed task i.e. satisfying the important economic property individual rationality. The \( P_r \) data structure in line 22 keeps track of all the payment of all the winning task requesters satisfying the individual rationality. If the condition in line 22 is not satisfied then the winning task requester is priced out of the market as depicted in line 26. Finally, a call to the allocation phase is done line 29.

**Payment function**: For determining the payment of each agent the valuation of the first losing task executor and losing task requester is taken into consideration which is given by \( I^*_r = \arg\max \{\rho'_r - \rho'_i < 0\} \) such that \( \gamma'_r = B_i \cdot \delta'_r \) and \( \gamma'_i = S'_i \cdot \delta'_r \). For defining the payment, we further require to fetch the valuation of the task requester and the task executor at the index position \( I^*_r \). The valuation of the task requester at any index position is captured by the bijective function \( \gamma^r : Z \rightarrow \mathbb{R}_{\geq 0} \), whereas the valuation of the task executor at any index position is captured by the bijective function \( \gamma^i : Z \rightarrow \mathbb{R}_{\geq 0} \). Let us further denote the valuation of the task requester at the index position \( I^*_r \) by \( \gamma^r(I^*_r) \) and the valuation of the task executor at \( I^*_r \) by \( \gamma^i(I^*_r) \). For determining the payment of all winning task executers and task requesters we will take the help of the average of the cost of the task executor at \( I^*_r \) and the value of the task requester at \( I^*_r \) given by \( \eta = \frac{\gamma^r(I^*_r) + \gamma^i(I^*_r)}{2} \).

Mathematically, the payment of \( i^{th} \) task executor is given as:

\[
P^r_i(\tau_i) = \begin{cases} 
\eta, & \text{if } \gamma^r(I^*_r) \geq \eta \text{ and } \gamma^i(I^*_r) \leq \eta \\
\gamma^r(I^*_r), & \text{otherwise}
\end{cases}
\]

Similarly, the payment of the \( i^{th} \) task requester is given as:

\[
P^i_i(\tau_i) = \begin{cases} 
\eta, & \text{if } \gamma^r(I^*_r) \leq \eta \text{ and } \gamma^i(I^*_r) \geq \eta \\
\gamma^i(I^*_r), & \text{otherwise}
\end{cases}
\]

In this problem set-up, for any particular time slot \( \tau_i \in \tau \) there might be two types of agents: (a) Freshly arrived agents, (b) Still active agents. For freshly arrived task executers and task requesters the payment is calculated as shown below. More formally, the payment of \( i^{th} \) task requester is given as:

\[
\zeta^r_i(\tau_i) = \begin{cases} 
\min_{\rho'_i \in [\rho'_{i-\kappa}, \tau_i]} [P^r_i(\rho')], & \text{if task requester is freshly arrived} \\
\max[\zeta^r(\tau_{i-1}), P^r_i(\tau_i)], & \text{if task executers are still active}
\end{cases}
\]

Here \( \kappa \) is the maximum permitted gap between the arrival and departure of any arbitrary agent \( i \).

\[
\zeta^i(\tau_i) = \begin{cases} 
\min_{\rho'_i \in [\rho'_{i-\kappa}, \tau_i]} [P^r_i(\rho')], & \text{if task executor is freshly arrived} \\
\min[\zeta^i(\tau_{i-1}), P^r_i(\tau_i)], & \text{if task executers are still active}
\end{cases}
\]

Now, if after \( \tau_i \), time slots if a task requester \( i \) is a winner then the final payment of that task requester will be given by \( P^r_i(\tau_i) = \max[\zeta^r(\tau_{i-1}), P^r_i(\tau_i)] \) and similarly if after \( \tau_i \), time slots if a task executor \( i \) is a winner then the final payment of that task executor will be given by \( P^i_i(\tau_i) = \min[\zeta^i(\tau_{i-1}), P^i_i(\tau_i)] \). (4)

4) **Allocation**: The input to the allocation phase are the \( j^{th} \) cluster in \( \tau_i \) time slot i.e. \( \mathcal{L}_j \), the set of task requester \( \mathcal{R} \), the set of cost of task execution of task executers i.e. \( \gamma^r \), and the set of values of task requesters i.e. \( \gamma^i \). The output is the set of task requester-task executor winning pairs held in \( \mathcal{A}_k \). Line 3 sorts the cluster \( \mathcal{L}_i \) in ascending order based on the elements of \( P_r \) and held in \( \mathcal{U}_c \) data structure. The set of active task requesters are sorted in descending order based on the elements of \( P_r \) and held in \( \mathcal{R}_c \) data structure.

**Algorithm 4 Allocation (\( \mathcal{L}_j, \mathcal{R}, \gamma^r, \gamma^i \))**

1: \( \mathcal{A}_k \leftarrow \phi \)
2: \begin{align*}
3: & \mathcal{U}_c \leftarrow \text{Sort_ascend}(\mathcal{L}_j, S_i \cdot \chi^r_i) \quad \text{→ Sorting based on} \\
4: & \mathcal{R}_c \leftarrow \text{Sort_descend}(\mathcal{R}, B_i \cdot \chi^i_i) \quad \text{→ Sorting based on}
\end{align*}
5: \( I_j \leftarrow \arg\max_{\chi^r_i \in \mathcal{U}_c, \chi^i_i \in \mathcal{R}} (\chi^r_i - \chi^i_i \geq 0) \)
6: for \( k = 1 \) to \( I_j \) do
7: \( \mathcal{U}_c \leftarrow \mathcal{U}_c \setminus \{S_i \in \mathcal{U}_c\} \)
8: \( \mathcal{R}_c \leftarrow \mathcal{R}_c \setminus \{B_i \in \mathcal{R}_c\} \)
9: \( \mathcal{A}_k \leftarrow \mathcal{A}_k \cup (\mathcal{U}_c, \mathcal{R}_c) \)
10: \end for
11: \( \mathcal{U}_c \leftarrow \mathcal{U}_c \setminus \mathcal{U}_c \)
12: \( \mathcal{R}_c \leftarrow \mathcal{R}_c \setminus \mathcal{R}_c \)
13: return \( (\mathcal{A}_k, \mathcal{U}_c, \mathcal{R}_c) \)
14: \end

Line 5 determines the largest index \( i \) that satisfy the condition that \( \chi^r_i - \chi^i_i \geq 0 \). The for loop in line 6-10 iterates over the \( I_j \) winning task executor-task requester pairs. In line 7, \( \mathcal{U}_c \) data structure keeps track of all the winning task executers at a particular time slot \( \tau_i \) in a particular cluster \( \mathcal{L}_j \). The \( \mathcal{R}_c \) data structure keeps track of all the \( I_j \) winning task requesters. The \( \mathcal{A}_k \) data structure in line 9 keeps track of all the winning task executor-task requester pairs. Line 11 and 12 removes the winning task executers and winning task requesters respectively from the auction market. Line 13 returns the allocation set \( \mathcal{A}_k, \mathcal{U}_c, \) and \( \mathcal{R}_c \).

C. Analysis of STEM

The STEM is a four stage mechanism consists of: main routine, cluster formation, allocation, and payment. So, the
running time of STEM will be the sum of the running time of main routine, cluster formation, allocation, and payment. Line 2 of the main routine is bounded above by $O(1)$. The for loop in line 3 executes for $s+1$ times as we have $s$ time slots. For the analysis purpose, WLOG we have $n \geq m$. Line 4 will take $O(n)$ time as there are $m$ task requesters. Line 5 is bounded above by $O(m)$ as there are $m$ task requesters. Line 6-8 is bounded above by $O(n)$. The time taken by line 9-11 in worst case is given by $O(m)$. Considering the cluster formation phase, in a given time horizon it is bounded above by $O(c \times k \times n^2) = O(ckn^2)$; where $c$ is number of iterations for which the change in the clusters are to be calculated. The sorting in line 14 and 15 is bounded above by $O(n \lg n)$ and $O(m \lg m)$ respectively. Now, talking about the payment phase motivated by [1], for $k$ different clusters line 3-28 will take $O(k \times k \times n^2) = O(ckn^2)$; where $k$ is the patience bound. Line 29 in the payment phase calls the allocation phase that will contribute $O(n \lg n) + O(m \lg m)$ in the worst case. So, the overall time complexity is dominated by the payment phase and is given as $O(k \times k \times n^2) = O(ckn^2)$. Line 17-20 in main routine phase is bounded above by $O(n)$. Line 25 and 26 takes $O(n)$ and $O(m)$ time respectively. The overall running time of the STEM is: $O(n) + O(n) + O(n) + O(ckn) + O(n \lg n) + O(m \lg m) + O(ckn^2) + O(n) + O(n) = O(ckn^2)$. The analysis is carried out by considering the case $n \geq m$, similarly the case with $m \geq n$ can be tackled and will result in $O(ckn^2)$.

**Lemma 1.** Agent $i$ can’t gain by misreporting their arrival time or departure time or both.

*Proof.* As the agents can mis-report the arrival time or the departure time, so the proof can be illustrated into two parts considering both the cases separately.

- **Case 1** ($\tilde{a}_i^r \neq a_i^r$): Fix $\tilde{d}_i^r$, $\tau_i$. Let us suppose an agent $i$ reports the arrival time as $\tilde{a}_i^r$ such that $\tilde{a}_i^r \neq a_i^r$ or in more formal sense $\tilde{a}_i^r > a_i^r$. It means that, agent $i$ will be aligned with more number of time slots before winning when reporting $\tilde{a}_i^r$ than in the case when reporting truthfully i.e. $a_i^r$ as shown in Fig. 2. Now, it is seen from the construction of the payment function that the agent $i$ will be paid less than or equal to the payment he/she (henceforth he) is receiving when reporting truthfully.

- **Case 2** ($\tilde{d}_i^r \neq d_i^r$): Fix $a_i^r$, $\tau_i$. Let us suppose an agent $i$ reports the departure time as $\tilde{d}_i^r$ such that $\tilde{d}_i^r \neq d_i^r$ or in more formal sense $\tilde{d}_i^r < d_i^r$. It means that, an agent $i$ will be aligned with more number of time slots before becoming inactive when reporting $\tilde{d}_i^r$ than in the case when reporting truthfully i.e. $d_i^r$ as shown in Fig. 3. Now, it is seen from the construction of the payment function is that the agent $i$ will be paid less or equal to the payment he is paid when reporting truthfully.

**Lemma 2.** Agent $i$ can’t gain by misreporting his/her bid value.

*Proof.* Considering the case of task executers. Fix the time slot $\tau_i \in T$ and the cluster.

**Case 1:**

Let us suppose that the $i^{th}$ winning task executer deviates and reports a bid value $\tilde{v}_i^e > v_i^e$. As the task executer was winning with $v_i^e$, with $\tilde{v}_i^e$ he would continue to win and his utility $\tilde{\varphi}_i^e = \varphi_i^e$. If instead he reports $\tilde{v}_i^e < v_i^e$. Again two cases can happen. He can still win. If he wins his utility, according to the definition will be $\tilde{\varphi}_i^e = \varphi_i^e$. If he loses his utility will be $\tilde{\varphi}_i^e = 0 < \varphi_i^e$. 

**Case 2:**

If the $i^{th}$ task executer was losing with $v_i$ let us see whether he would gain by deviation. If he reports $\tilde{v}_i^e < v_i^e$, he would still lose and his utility $\tilde{\varphi}_i^e = 0 = \varphi_i^e$. If instead he reports $\tilde{v}_i^e > v_i^e$. Two cases can occur. If he still loses his utility $\tilde{\varphi}_i^e = 0 = \varphi_i^e$. But if he wins, then he had to beat some valuation $v_i^e > v_i^e$ and hence $\tilde{v}_i^e > v_i^e$. Now as he wins his utility $\tilde{\varphi}_i^e = P_i^e - v_i^e = v_i^e - v_i^e < 0$. So he would have got a negative utility. Hence no gain is achieved.

Considering the case 1 and case 2 above, it can be concluded that any agent $i$ can’t gain by mis-reporting arrival time or departure time. The proof is carried out by considering the task executers, similar argument can be given for the task requesters. This completes the proof.

**Lemma 3.** STEM is weakly Budget balanced.

*Proof.* Fix the time slot $\tau_i$ and cluster $E_j$. This corresponds to the case when the sum of all the monetary transfers of all the agents type profiles is less than or equal to zero. Now, the construction of our STEM is such that, any task executer and task request is paired up only when $S_i \cdot P_i^r - B_i \cdot P_i^e \geq 0$. It means that, for any task executer-task request pair there exist some surplus. In the similar fashion, in a particular time slot $\tau_i$ and in a particular cluster considering all the agents, $\sum S_i \cdot P_i^r - \sum B_i \cdot P_i^e \geq 0$. Hence, the sum total of payments made to the task executers is at least as high as the sum total of the payments received by the task requesters and their is a surplus. Hence, it is proved that the STEM is budget balanced for a particular time slot $\tau_i$ and for a particular cluster. From
Lemma 4. STEM is individual rational.

Proof. Fix the time slot $\tau_i$ and cluster $\mathcal{C}_j$. Individual rationality means that each agent gains a utility that is no less than he would get without participating in a mechanism. Considering the case of task requester, when the task requester is winning then it is ensured that he has to pay an amount $\mathcal{P}_{\tau_i}$ such that $\mathcal{U}'_i \geq \mathcal{P}_{\tau_i}$. From this inequality it is clear that the winning task requester has to pay amount less than his bid value. So, in this case it can be concluded that $\phi_i = \mathcal{U}'_i - \mathcal{P}_{\tau_i} \geq 0$. Moreover, if the task requester is losing in that case his utility is 0. From our claim it must be true for any time slot $\tau_i, \ldots, \tau_s$ and any cluster. Similar argument can be given for the task executers. This completes the proof. □

V. CONCLUSION AND FUTURE WORKS

An incentive compatible mechanism is proposed in this paper to circumvent the location information in online double auction setting for the participatory sensing. In our future work we will focus on investigating the quality consequence in this environment. Another interesting direction is to find algorithms when the task requesters have some limited budgets.

REFERENCES

[1] S. Aflaki, N. Meratnia, M. Baratchi, and P. Havinga. Evaluation of incentives for body area network-based healthcare systems. In 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNP), pages 515–520. IEEE, April 2013.

[2] S. Anawar and S. Yahya. Empowering health behaviour intervention through computational approach for intrinsic incentives in participatory sensing application. In 3rd International Conference on Research and Innovation in Information Systems (ICRIIS), pages 281–285. IEEE, November 2013.

[3] J. Bhattacharjee, A. Pal, S. Mukhopadhyay, and A. Bharama. Incentive and quality aware participatory sensing system. In 12th International Conference on Dependable, Autonomic and Secure Computing (DASC), pages 382–387. IEEE Computer Society, August 2014.

[4] J. Bredin and D. C. Parkes. Models for truthful online double auctions. In 21st International Conference on Uncertainty in Artificial Intelligence (UAI), pages 50–59. AUAI press, July 2005.

[5] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti. Crowdsourcing with smartphones. IEEE Internet Computing, 16(5):36–44, 2012.

[6] E. D. Cristofaro and C. Soriente. Participatory privacy: enabling privacy in participatory sensing. IEEE Network Magazine, 27(1):32–36, 2013.

[7] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. IEEE Communications Magazine, 49(11):32–39, 2011.

[8] H. Gao and C. H. Liu. A survey of incentive mechanisms for participatory sensing. IEEE Communications Surveys & Tutorials, 17:918–943, 2015.

[9] Y. Gao, Y. Chen, and K. J. R. Liu. On cost-effective incentive mechanisms in microtask crowdsourcing. IEEE Transactions on Computational Intelligence and AI in Games, 7(1):3–15, March 2015.

[10] E. Horvitz, J. Agacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In 21st Conference on Uncertainty in Artificial Intelligence (UAI), pages 275–283. AUAI Press, July 2005.

[11] L. G. Jaimes, V. L. Idaides, and M. A. Labrador. A location-based incentive mechanism for participatory sensing systems with budget. In 16th International Conference on Pervasive Computing and Communications (PerCom), pages 103–108. IEEE, March 2012.

[12] K. Lan and H. Wang. On providing incentives to collect road traffic information. In 9th International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, July 2013.

[13] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. IEEE Communication Magazine, 48(9):140–150, September 2010.

[14] J. S. Lee and B. Hoh. Dynamic pricing incentive for participatory sensing. Elsevier Journal of Pervasive and Mobile Computing, 6(6):693–708, December 2010.

[15] Q. Li and G. Cao. Providing privacy-aware incentives for mobile sensing. In 34th International Conference on Distributed Computing Systems (ICDCS), pages 76–84. IEEE, June-July 2014.

[16] T. Luo, H. P. Tan, and L. Xia. Profit-maximizing incentive for participatory sensing. In 33rd Annual International Conference on Computer Communications (INFOCOM), pages 127–135. IEEE, April-May 2014.

[17] E. Massung, D. Coyle, F. K. Cater, M. Jay, and C. Preist. Using crowdsourcing to support pro-environmental community activism. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 371–380. ACM, April-May 2013.

[18] P. Mohan, V. N. Padmanabhan, and R. Ranjee. Nericell: Rich monitoring of road and traffic conditions. In 6th Conference on Embedded Network Sensor Systems (SenSys), pages 323–336. ACM, November 2008.

[19] J. Mukhopadhyay, A. Pal, S. Mukhopadhyay, and V. K. Singh. Quality adaptive online double auction in participatory sensing. CoRR, abs/1608.04857, 2016.

[20] M. Mun, S. Reddy, K. Hilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In 7th International Conference on Mobile Systems, Applications, and Services (MobiSys), pages 55–68. ACM, June 2009.

[21] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava. Examining micro-payments for participatory sensing data collections. In 12th International Conference on Ubiquitous Computing (UbiComp), pages 33–36. ACM, September 2010.

[22] F. Restuccia, S. K. Das, and J. Payton. Incentive mechanisms for participatory sensing: Survey and research challenges. In Transactions on Sensor Network (TOSN), 12(2):13:1–13:40, April 2016.

[23] I. J. Vergara-Laurens, D. Mendez, and M. A. Labrador. Privacy, quality of information, and energy consumption in participatory sensing systems. In 18th International Conference on Pervasive Computing and Communications (PerCom). IEEE, March 2014.

[24] R. Yu, R. Liu, X. Wang, and J. Cao. Improving data quality with an accumulated reputation model in participatory sensing systems. Sensors, 14(3):5573–5594, 2014.

[25] 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.