A Balanced Dissemination of Time Constraint Tasks in Mobile Crowdsourcing: A Double Auction Perspective

Jaya Mukhopadhyay[1(✉)], Vikash Kumar Singh[2], Sajal Mukhopadhyay[3], and Anita Pal[1]

[1] Department of Mathematics, National Institute of Technology, Durgapur 713209, West Bengal, India
jayabesu@gmail.com, anita.buie@gmail.com

[2] School of Computer Science and Engineering, Vellore Institute of Technology, Amaravati 522237, Andhra Pradesh, India
vikash.singh@vitap.ac.in

[3] Department of Computer Science and Engineering, National Institute of Technology, Durgapur 713209, West Bengal, India
sajal@cse.nitdgp.ac.in

Abstract. Mobile crowdsourcing (MCS) is gaining real attention in recent years as it has found widespread applications such as traffic monitoring, pollution control surveillance, locating endangered species, and many others. This paradigm of research is showing an interesting power of smart devices that are held by intelligent agents (such as human beings). In MCS, the tasks which are outsourced are executed by the task executors (intelligent agents carrying smart devices). In this paper, how overlapping tasks (with a deadline) can be disseminated in slots and leveraged as evenly as possible to the stakeholders (task executors or sellers) is addressed through a scalable scheduling (interval partitioning) and economic mechanism (double auction). It is proved that our mechanism is truthful and also shown via simulation that our proposed mechanism will perform better when the agents are manipulative in nature.

1 Introduction

With unprecedented growth of *smartphone* users, the agents carrying smartphones can potentially serve many things [1,2]. The application ranges from environment (say for example collecting data of pollution condition in some areas), eco-system restoring (collecting information for to-be-extinct species), transportation (acquiring information of road condition) and many more [3,4]. In all cases they collect and send information to the task requester(s) (or buyer(s)). Agents with smartphones when solving such a system, is called mobile crowdsourcing (MCS)[1]. In this model task requester(s) publishes tasks to a platform and then the task executor(s) or seller(s) serves the tasks. The architecture of the MCS system is given in Fig. 1.

[1] In literature, mobile crowdsourcing is also termed as participatory sensing.

L. Barolli et al. (Eds.): 3PGCIC 2020, LNNS 158, pp. 74–85, 2021.
https://doi.org/10.1007/978-3-030-61105-7_8

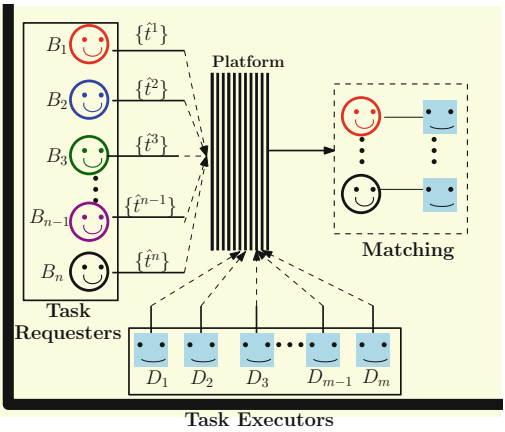


Fig. 1. System model

MCS relies on the fact that the agents provide the data. But the fundamental question is: Why the agents should provide the data? or how they can be motivated? One way to think that in some applications from their social urge they should provide the data. In another case, in many applications, agents may be motivated only when they are given some incentives (ex: money). There are several works devoted to incentivizing agents in MCS [5–8]. In this paper, how the time constrained jobs are to be separated and distributed in a balanced way to the participating sellers through double auction, is addressed.

The main contributions of our paper are:

- First the submitted jobs of the task providers (buyers) are distributed to $|d|$ slots, so that they become non overlapping.
- Second, the jobs of each slot are assigned as evenly as possible, so that sellers are not overburden.
- The third goal is to design a truthful mechanism so that social welfare (in usual economic sense) is maximized.

The rest of the paper is organized as follows. In Sect. 2, preliminaries of the proposed scheme is presented. The system model is discussed in Sect. 3. In Sect. 4, the detailing of the proposed algorithms are presented. In Sect. 5, the simulations are carried out. Conclusions and future works are given in Sect. 6.

2 Related Works

In this section, our emphasis will be to give a brief overview about the prior works done in the field of MCS. Our discussion mainly circumvent around the incentive aspects in MCS, and quality of data provided by the participating agents, etc.

The readers can go through [3,4,9] in order to get an overview of the field. The whole of the MCS field relies mainly on the idea of collecting the data from large group

of interested users (may be common people) having some sensing devices (say smartphone) geographically distributed around the globe. Following the above discussion, the natural question that arise in ones mind is: why the agents should provide the data by placing so much effort (CPU utilization, power consumption, etc.) and more importantly exposing their locations? Answering to the above raised question: In [10] the works have been done in the direction of: how to influence large number of participants to take part in the sensing process along with the evaluation of their provided data? Several incentive schemes have been introduced in [5,6,8]. Dissimilar to above discussed incentive schemes, in [11–13] the incentive compatible mechanisms are introduced for the MCS environment. Some works in MCS environment is devoted to the quality of the data collected by the participating agents [11,14]. However, the drawback that is identified in these proposed schemes is that, in the proposed mechanism the quality of the data reported by the agents to the system are not taken into consideration. In [14,15] efforts have been made in this direction by combining the quality of data provided by the agents with their respective bids.

Some works [16–18] model MCS market as double auction, where task requesters play the role of buyers to buy the sensed data from crowd. In [16], for sensing task allocation in MCS system a truthful double auction mechanism is proposed, that take into consideration the relationship between the number of users that are assigned to do the tasks and utility of task requesters. In [17], a general framework for designing the truthful double auction mechanisms for the dynamic mobile crowdsourcing is proposed. In [18], an approach based on max-min fairness task allocation is discussed. The utilities of the participating agents are maximized, and the trusted data are gathered using incentive mechanisms.

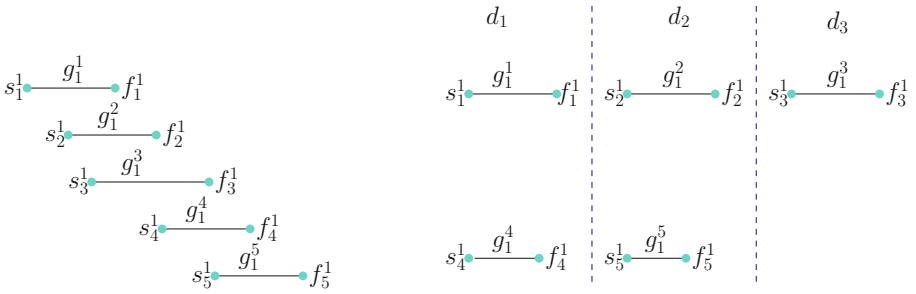
From the above discussed works it can be seen that no work has considered the balanced dissemination of time constrained tasks to the task executors. In this paper, we have investigated this scenario and proposed a double auction based mechanism coupled with scheduling algorithm.

3 System Model

In this MCS model we have a set of participating buyers (task providers) $B = \{B_1, B_2, \ldots, B_n\}$. A set of participating sellers (task executors) $D = \{D_1, D_2, \ldots, D_m\}$. Usually the number of sellers are far more than the number of buyers. Buyers submit a set of jobs $J = \{J_1, J_2, \ldots, J_n\}$, where any $J_i = \{g_j^i\}$ denotes the set of all jobs submitted by the i^{th} buyer. Each job g_i^j has a start time s_j^i and a finish time f_j^i where $f_j^i \geq s_j^i$. However two jobs j and j' may overlap. For the overlapping case of two jobs (j and j', where j being the first job without loss of generality) we have $s_j^i \leq s_{j'}^i \leq f_j^i$ (if the two jobs from the same buyer) or $s_j^i \leq s_{j'}^k \leq f_j^i$ (if two jobs from the different buyers). The first non-trivial goal is to decode the minimum number of slots needed to distribute all the jobs so that no two jobs overlap. To understand the meaning of slot, consider Fig. 2a and Fig. 2b submitted jobs are shown (may be overlapping).

In Fig. 2b non-overlapping jobs are separated in three slots (d_1, d_2, *and* d_3). Slots are the container of non-overlapping jobs. The second goal is to distribute the jobs to the sellers as evenly as possible. For this we use a double auction framework. In

this problem task providers (buyers) and the task executors (sellers) are considered as *strategic* in nature. The buyers are constrained by budget. In this model, the buyers submit the jobs along with the valuation; The valuation of job submitted by arbitrary buyer i is denoted by β_j^i (i^{th} buyer, j^{th} job). β_j^i is the private information of the buyers and denotes the maximum amount the buyer can pay for his task to be executed. This we will further refer as the bid valuation of the buyers. The sellers (task executors) also submit their bid valuation and is denoted by δ_j^i. So, the bid vectors for both buyer and the seller is denoted by $\beta = \{\beta_j^i\}$ and $\delta = \{\delta_j^i\}$. the third goal is to ensure that the agents (strategic) should not misreport their valuation so that the social welfare (in usual economic sense) is maximized.



(a) Overlapping Jobs *(b)* Non-Overlapping Jobs

Fig. 2. Jobs configurations

The utility of any *seller* is defined as the difference between the payment received by the *seller* and the true valuation of the *seller*. More formally, the utility of D_i is:

$$u_i^s = \begin{cases} \sum_j \hat{p}_j^i - \sum_j \delta_j^i, & \text{if } D_i \text{ wins} \\ 0, & \text{Otherwise} \end{cases} \tag{1}$$

Similarly, the utility of any *buyer* is defined as the difference between the true valuation of the *buyer* and the payment he pays. More formally, the utility of B_i is:

$$u_i^b = \begin{cases} \sum_j \beta_j^i - \sum_j p_j^i, & \text{if } B_i \text{ wins} \\ 0, & \text{Otherwise} \end{cases} \tag{2}$$

Further, in Lemma 2 it has been shown that the proposed mechanism is truthful (see Definition below).

Definition 1 (Truthful). Truthful *means that, if the utility relation for the i^{th} buyer is $u_i^b \geq u'^b_i$ holds considering that u_i^b is the utility of buyer i when he is reporting his true bid profile vector β_j^i and u'^b_i is the utility of that buyer when he is reporting any other bid profile vector $\beta'^i_j \neq \beta_j^i$. For each seller i, truthful means $u_i^s \geq u'^s_i$.*

4 Proposed Mechanism

The proposed mechanism namely ***B**alanced **d**istribution **o**f **t**ime bound **t**ask **u**sing **d**ouble **a**uction (BDoTTuDA)* is a two step process: (a) Separating the non-overlapping jobs into d slots, and (b) The set of tasks of each $d_i \in d$ are then auctioned off.

4.1 Sketch of BDoTTuDA

The BDoTTuDA can further be studied under two different sections: *Partitioning and Scheduling*, and *Double auction mechanism*. First the sub-part of the proposed mechanism *i.e. Partitioning and Scheduling* phase is discussed and presented motivated by [19]. The *Double auction mechanism* is presented next motivated by [20].

Algorithm 1. Partitioning and Scheduling (J)

Output: $d \leftarrow \phi$

1: **begin**
2: $x \leftarrow 0$, Heap $\leftarrow \phi$, $j = 1$, $\ell = 0$, count = 0
3: $\hat{S}$ = Sort(J) ▷ Sort jobs based on start time.
4: $Q \leftarrow \hat{S}$ ▷ Maintain a queue for the sorted jobs.
5: $x \leftarrow$ Delete(Q)
6: slot (d_j, x) ▷ Placing x in j^{th} slot.
7: $d_j \leftarrow d_j \cup \{x\}$
8: $d \leftarrow d \cup d_j$
9: *Heap insert* (Heap, ℓ, $(x.f,\ j)$) **/* Inserting element in heap */**
10: **while** $Q \neq \phi$ **do**
11: $x \leftarrow$ Delete(Q)
12: $(\hat{f},\ j') \leftarrow$ *minimum* $(Heap)$ ▷ Returns the minimum element from heap
13: **if** $x \cdot s \geq \hat{f}$ **then**
14: slot ($d_{j'}$, x)
15: $d_{j'} \leftarrow d_{j'} \cup \{x\}$
16: *Heap update* (Heap, 0, $(x.f,\ j')$) **/* Updating heap */**
17: **else**
18: $j = j + 1$
19: slot (d_j, x) ▷ Placing x in j^{th} slot.
20: $d_j \leftarrow d_j \cup \{x\}$
21: $d \leftarrow d \cup d_j$
22: *Heap insert* (Heap, ℓ, $(x.f,\ j)$) **/* Inserting element in heap */**
23: **end if**
24: **end while**
25: **return** d
26: **end**

4.1.1 Partitioning and Scheduling

The input to the Partitioning and Scheduling phase is the set of available jobs given as J. The output the set of slots containing the buyers. Talking about Algorithm 1, Line 3 sorts the jobs based on the given start time. In Line 4, the sorted jobs are maintained

in a Queue data structure. Line 5–8 assign a slot d_1 to the first job in the queue Q. Line 9 initializes the Heap containing the finish time of the currently selected job from Q. The *While* loop in line 10–24 perform the rest of the process of scheduling jobs to particular slot by utilizing several sub-routines such as *Heap insert*, *Heap update*, and *Min heapify*. The *while* loop terminates once all the jobs assigned to respective non-overlapping slots.

Algorithm 2. Double auction mechanism (D, d_i, β, δ)

Output: $\mathscr{A} \leftarrow \phi$, $\hat{p}^s \leftarrow \phi$, $p^b \leftarrow \phi$

1: **begin**
2: **for** each $d_i \in d$ **do**
3: $B' \leftarrow \phi, D' \leftarrow \phi$
4: $D \leftarrow$ Sort_ascending$(D, D.\delta_j^i)$ ▷ Sorting based on $\delta_j^i \in \delta$ for all $D_i \in D$
5: $d \leftarrow$ Sort_descending$(d_i, d_i.\beta_j^i)$ ▷ Sorting based on $\beta_j^i \in \beta$ for all $d_i^j \in d_i$
6: $\kappa \leftarrow argmax_k\{d_i.\beta_j^i - D.\delta_j^i \geq 0\}$
7: **for** $i = 1$ to κ **do**
8: $B' = B' \cup \{d_i^j\}$
9: $D' = D' \cup \{D_i\}$
10: $\mathscr{A}_i \leftarrow (B', D')$
11: **end for**
12: **for** $i = 1$ to κ **do**
13: **if** $\left(\frac{\beta_i^{\kappa+1}+\delta_i^{\kappa+1}}{2} \leq \beta_i^{\kappa}\right)$ and $\left(\frac{\beta_i^{\kappa+1}+\delta_i^{\kappa+1}}{2}\right) \leq \delta_i^{\kappa}$ **then**
14: $p_i^j \leftarrow \beta_i^{\kappa+1}$; $p_i^b \leftarrow p_i^b \cup p_i^j$
15: $\hat{p}_i^j \leftarrow \delta_i^{\kappa+1}$; $p_i^s \leftarrow p_i^s \cup \hat{p}_i^j$
16: **else**
17: $p_i^j \leftarrow \beta_i^{\kappa}$; $p_i^b \leftarrow p_i^b \cup p_i^j$
18: $\hat{p}_i^j \leftarrow \delta_i^{\kappa}$; $p_i^s \leftarrow p_i^s \cup \hat{p}_i^j$
19: **end if**
20: **end for**
21: $\mathscr{A} \leftarrow \mathscr{A} \cup \mathscr{A}_i$
22: $\hat{p}^s \leftarrow \hat{p}^s \cup \hat{p}_i^j$
23: $p^b \leftarrow p^b \cup p_i^j$
24: **end for**
25: **return** $\mathscr{A}$, $\hat{p}^s$, p^b
26: **end**

4.1.2 Double Auction Mechanism

The input to the *Double auction mechanism* are the set of sellers *i.e.* D, and the set of buyers in a slot $d_i \in d$. The output is the set of *buyers-sellers* winning pairs held in $\mathscr{A}_i$ data structure. Line 5 sorts the sellers in ascending order based on the elements of δ_j^i. The set of *buyers* in d_i slot are sorted in descending order based on the elements of β_j^i. Line 6 determines the largest index κ that satisfy the condition that $d_i.\beta_j^i - D.\delta_j^i \geq 0$. The *for* loop in line 7–11 iterates over the κ winning *Buyer-seller* pairs. In line 8 B'

data structure keeps track of all the winning *buyers*. The D' data structure keeps track of all the winning *sellers*. The $\mathscr{A}_i$ data structure in line 10 keeps track of all the winning *seller-buyer* pairs. Line 12–20 determines the payment of winning seller-buyer pairs. Line 25 returns the allocation set $\mathscr{A}$, the seller's payment $\hat{p}^s$, and the buyers payment p^b.

Lemma 1. *The running time of BDoTTuDA is* $O(m \lg m)$.

Proof. The running time of BDoTTuDA will be equal to the sum of the running time of the *Partitioning and Scheduling*, and *Double auction mechanism*. The running time of the *Partitioning and Scheduling* phase is given as $O(n \lg n)$. The *Double auction mechanism* takes $O(m \lg m)$ time. So, the running time of BDoTTuDA is given as $O(n \lg n) + O(m \lg m) = O(m \lg m)$, as the number of sellers are far greater than the number of buyers.

Lemma 2. *BDoTTuDA is Truthful.*

Proof. Let us consider the case of *sellers*. Fix slot $d_i \in d$.

Case 1. Let us say that the i^{th} winning *seller* misreports a bid value as ${\delta'}^i_j > \delta^i_j$. As the *seller* was winning with δ^i_j, with ${\delta'}^i_j$ he would keep on winning and his utility ${u'}^s_i = u^s_i$. If, say, he reports ${\delta'}^i_j < \delta^i_j$. This will give rise to two cases. He can still be in the winning set. If he is in the winning set his utility from the definition will be ${u'}^s_i = u^s_i$. If he is in losing set, then his utility will be ${u'}^s_i = 0 < u^s_i$.

Case 2. If the i^{th} *seller* was in losing set and he reports ${\delta'}^i_j < \delta^i_j$, he would still belong to losing set and his utility ${u'}^s_i = 0 = u^s_i$. If instead he reports ${\delta'}^i_j > \delta^i_j$. This wil give rise to two cases. If he still belong to losing set his utility ${u'}^s_i = 0 = u^s_i$. But if he is in winning set, then he had to beat some valuation ${\delta'}^k_l > \delta^i_j$ and hence $u^s_i > u^s_k$. Now as he is in wining set his utility ${u'}^s_i = \hat{p}^i_j - \delta^i_j = \delta^k_l - \delta^i_j < 0$. So he would have got a negative utility. Hence no gain is achieved.
From the above two cases *i.e.* Case 1 and Case 2, it can be concluded that any seller i can't gain by mis-reporting his bid value. The proof considers the *sellers* case, similar road map could be followed for the *buyers*. This completes the proof.

Lemma 3. *The number of tasks that is assigned to any* j^{th} *slot in expectation is given as* $\frac{n}{k}$*, where k is the number of slots available and n is the number of tasks. In other words, we can say*

$$E[\mathbb{Z}_j] = \frac{n}{k}$$

where, $\mathbb{Z}_j$ *is the random variable measuring the number of tasks assigned to any* j^{th} *slot out of n tasks.*

Proof. Fix a slot j. In this, our goal is to compute the expected number of tasks assigned to any given slot. The indicator random variable $\mathbb{Z}_j$ is used to determine the total number of tasks assigned to j^{th} slot. So, the expected number of tasks assigned to j^{th} slot is given as $E[\mathbb{Z}_j]$. Let us say we have k different slots. Now, when a task is picked up for

allocating a slot, then it can be placed in any of these k different slots. So, any slot ℓ ($1 \leq \ell \leq k$) could be the outcome of the experiment (allocation of slot to the task). It is to be noted that the selection of any such ℓ is equally likely. Therefore, each task t_i can be assigned to any j^{th} slot with probability $\frac{1}{k}$. We define an indicator random variable $\mathbb{Z}_j^i$ associated with the event in which t_i task is assigned to j^{th} slot. Thus,

$$\mathbb{Z}_j^i = I\{t_i \textit{ task is assigned to } j^{th} \textit{ slot}\}$$

$$= \begin{cases} 1, & \textit{if task } t_i \textit{ is assigned to } j^{th} \textit{ slot}, \\ 0, & \textit{otherwise} \end{cases}$$

Taking expectation both side, we get

$$E[\mathbb{Z}_j^i] = E[I\{t_i \textit{ task is assigned to } j^{th} \textit{ slot}\}]$$

As always with the indicator random variable, the expectation is just the probability of the corresponding event [21]:

$$\begin{aligned} E[\mathbb{Z}_j^i] &= 1 \cdot Pr\{\mathbb{Z}_j^i = 1\} + 0 \cdot Pr\{\mathbb{Z}_j^i = 0\} \\ &= 1 \cdot Pr\{\mathbb{Z}_j^i = 1\} \\ &= \frac{1}{k} \end{aligned} \tag{3}$$

Now, let us consider the random variable that is of our interest and is given as $\mathbb{Z}_j = \sum_{i=1}^{n} \mathbb{Z}_j^i$. The expected number of tasks assigned to any j^{th} slot is just the expected value of our indicator random variable $\mathbb{Z}_j$ and is given as $E[\mathbb{Z}_j]$. By taking expectation both side, we get

$$E[\mathbb{Z}_j] = E\left[\sum_{i=1}^{n} \mathbb{Z}_j^i\right] \tag{4}$$

By linearity of expectation, we get

$$E[\mathbb{Z}_j] = \sum_{i=1}^{n} E[\mathbb{Z}_j^i] \tag{5}$$

On substituting the value of $E[\mathbb{Z}_j^i]$ from Eq. 3 to Eq. 5, we get

$$E[\mathbb{Z}_j] = \sum_{i=1}^{n} \frac{1}{k} = \frac{n}{k}$$

So, one can conclude that, in any j^{th} slot on an average $\frac{n}{k}$ tasks will be assigned. Hence proved. □

Observation 1. *If the value of n is 100 and k is 5 then* $E[\mathbb{Z}_i] = \frac{n}{k} = \frac{100}{5} = 20$. *It means that, on an average each slot will be carrying 20 tasks.*

5 Experiments and Results

In this section, the proposed mechanism called BDoTTuDA is compared with the proposed benchmark mechanism (BM) that is vulnerable to manipulation. The manipulative nature of the task executors in case of BM can be seen easily in the simulation results. It is to be noted that, our BM differs only in terms of payment rule, the allocation rule is similar to that of BDoTTuDA.

As a payment rule of BM, each of the winning task executors will be paid his respective reported bid value and each of the winning task requesters will be paying his respective reported bid value. The unit of bid values reported by the task executors and task requesters is taken as $. The simulations are done using Python.

5.1 Simulation Setup

In our setup, the experiment runs for 100 times and the obtained values are plotted by taking average over these 100 times. The simulations are done considering the *uniform distribution* (UD). In case of UD, for all the agents the bid value ranges from 100 to 200. The performance metric that is considered in order to compare the two mechanisms is *utility*. The purpose for considering the utility parameter is to verify the two mechanisms based on *truthfulness*.

5.2 Result Analysis

In this section, BDoTTuDA is simulated against the benchmark mechanism (depicted as BM in the figures shown in the simulation results). BDoTTuDA is claimed to be *truthful* in our setting. To present the manipulative nature of the bechmark mechanism, the bid values of the subset of the agents participating in the system are varied. It is considered that 20% of the agents (in our case it is said to be small deviation) are manipulating their bid values by 30% of their true value (task executors are increasing their bid values by 30% of their true values and task requesters are decreasing their bid values by 30% of their true value). Similarly for the medium deviation (30% of the agents are manipulating their bid values by 30% of their true value) and large deviation (45% of the agents are manipulating their bid values by 30% of their true value). In the simulation results, BM-S-Dev, BM-M-Dev, and BM-L-Dev represents benchmark mechanism with small deviation, benchmark mechanism with medium deviation, and benchmark mechanism with large deviation respectively.

It is shown in Fig. 3 and 4 that the utility of agents in case of BDoTTuDA is more as compared to the utility of agents in case of benchmark mechanism (the utility of agents are 0 in case of BM.). This is due to the reason that in case of BDoTTuDA, the task executors are paid more than their true valuation and the task requesters are paying less than their true valuation, whereas in case of BM the task executors are paid equal to their reported bid value and the task requesters are paying equal to their reported bid value.

Considering the manipulative nature of the agents, it can be seen in Fig. 3 and 4 that if the agents are manipulating their bid values in case of BM, then they are gaining. More formally, for the BM case, the utility of the agents are higher in case of large

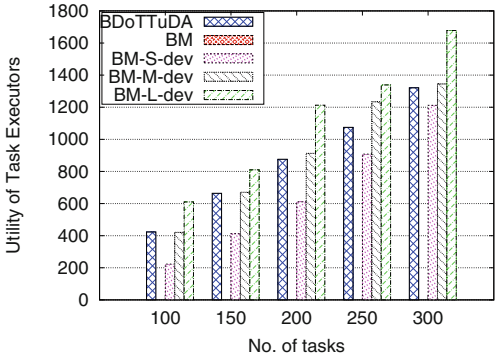


Fig. 3. Comparison of utility of task executors

deviation than in case of medium deviation than in case of small deviation than in case of no deviation. This is due to the reason that, the task executors are paid their reported bid values which is higher than their true valuation (in case of manipulation) and the task requesters are paying their reported bid value which is lower than their true valuation (in case of manipulation). So, it can be concluded that, more the number of agents manipulating their bid by some fixed amount, higher will be the utility of the agents. As the agents are increasing utility by manipulation in case of BM, so we can say that BM is vulnerable to manipulation. It is not a truthful mechanism.

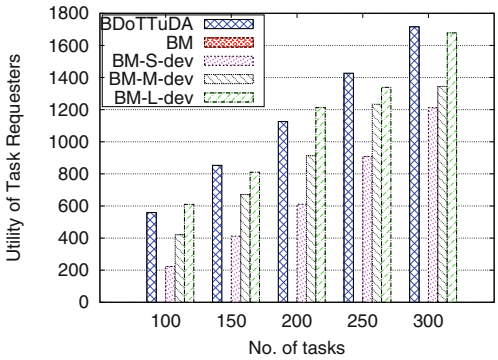


Fig. 4. Comparison of utility of task requesters

Another thing is that, in case of BM, if the task executors are deviating (increasing their bid values) by large amount from their true values, then in that case the task requesters will have to pay very high value. In that case, the task requesters will not be willing to pay such a huge amount. Also, if the task requesters are constrained by some budget, then many more task executors could not be served. So, if the manipulation

is their in the system, even if the task executors are gaining but it is not good for the system.

6 Conclusion and Future Works

In this paper a double auction framework is developed to distribute time bound tasks to the task executors and thereby achieving balance in distributions. In our future work balance distribution can be performed by thresholding on the number of tasks being allocated to the sellers. The other two directions could be to distribute the tasks by considering the location information and quality of the agents in our settings.

References

1. Hasenfratz, D., Saukh, O., Sturzenegger, S., Thiele, L.: Participatory air pollution monitoring using smartphones. In: Mobile Sensing: From Smartphones and Wearables to Big Data, Beijing, China, April 2012. ACM (2012)
2. J, W., et al.: Fine-grained multitask allocation for participatory sensing with a shared budget. IEEE Internet Things J. **3**(6), 1395–1405 (2016)
3. Phuttharak, J., Loke, S.W.: A review of mobile crowdsourcing architectures and challenges: toward crowd-empowered Internet-of-Things. IEEE Access **7**, 304–324 (2019)
4. Yu, R., Cao, J., Liu, R., Gao, W., Wang, X., Liang, J.: Participant incentive mechanism toward quality-oriented sensing: understanding and application. ACM Trans. Sen. Netw. **15**(2), 21:1–21:25 (2019)
5. Duan, Z., Tian, L., Yan, M., Cai, Z., Han, Q., Yin, G.: Practical incentive mechanisms for IoT-based mobile crowdsensing systems. IEEE Access **5**, 20383–20392 (2017)
6. Singh, V.K., Mukhopadhyay, S., Xhafa, F., Krause, P.: A quality-assuring, combinatorial auction based mechanism for IoT-based crowdsourcing. In: Advances in Edge Computing: Massive Parallel Processing and Applications, vol. 35, pp. 148–177. IOS Press (2020)
7. Mukhopadhyay, J., Singh, V.K., Mukhopadhyay, S., Pal, A.: Clustering and auction in sequence: a two fold mechanism for participatory sensing. In: Yadav, N., Yadav, A., Bansal, J.C., Deep, K., Kim, J.H. (eds.) Harmony Search and Nature Inspired Optimization Algorithms, pp. 347–356. Springer, Singapore (2019)
8. Singh, V.K., Mukhopadhyay, S., Xhafa, F., Sharma, A.: A budget feasible peer graded mechanism for IoT-based crowdsourcing. J. Ambient Intell. Humanized Comput. **11**(4), 1531–1551 (2020)
9. Restuccia, F., Das, S.K., Payton, J.: Incentive mechanisms for participatory sensing: survey and research challenges. Trans. Sensor Netw. **12**(2), 13:1–13:40 (2016)
10. Lee, J.S., Hoh, B.: Dynamic pricing incentive for participatory sensing. Elsevier J. Pervasive Mob. Comput. **6**(6), 693–708 (2010)
11. Zhao, D., Li, X.Y., Ma, H.: How to crowdsource tasks truthfully without sacrificing utility: online incentive mechanisms with budget constraint. In: Annual IEEE International Conference on Computer Communications (INFOCOM), pp. 1213–1221 (2014)
12. Feng, Z., Zhu, Y., Zhang, Q., Ni, L.M., Vasilakos, A.V.: Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing, pp. 1231–1239 (2014)
13. Gao, L., Fen, H., Jianwei, H.: Providing long-term participation incentive in participatory sensing (2015). arXiv preprint arXiv:1501.02480
14. Yu, R., Liu, R., Wang, X., Cao, J.: Improving data quality with an accumulated reputation model in participatory sensing systems. Sensors **14**(3), 5573–5594 (2014)

15. Bhattacharjee, J., Pal, A., Mukhopadhyay, S., Bharasa, A.: Incentive and quality aware participatory sensing system. In: 12^{th} International Conference on Dependable, Autonomic and Secure Computing (DASC), pp. 382–387. IEEE Computer Society (2014)
16. Xu, W., Huang, H., Sun, Y. E., Li, F., Zhu, Y.: Data: a double auction based task assignment mechanism in crowdsourcing systems. In: 2013 8th International Conference on Communications and Networking in China (CHINACOM), pp. 172–177 (2013)
17. Wei, Y., Zhu, Y., Zhu, H., Zhang, Q., Xue, G.: Truthful online double auctions for dynamic mobile crowdsourcing. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 2074–2082 (2015)
18. Huang, H., Xin, Y., Sun, Y., Yang, W.: A truthful double auction mechanism for crowdsensing systems with max-min fairness. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2017)
19. Kleinberg, J., Tardos, É.: Algorithm Design. Addison-Wesley, Boston (2006)
20. Bredin, J., Parkes, D.C.: Models for truthful online double auctions. In: 21^{st} International Conference on Uncertainty in Artificial Intelligence, pp. 50–59. AUAI press (2005)
21. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2009)