Multimedia Tools and Applications (2024) 83:9873–9892
https://doi.org/10.1007/s11042-023-16015-3

A truthful mechanism for time-bound tasks in IoT-based crowdsourcing with zero budget

Vikash Kumar Singh[1] · **Sanket Mishra**[1]

Received: 19 July 2021 / Revised: 29 May 2023 / Accepted: 12 June 2023 /
Published online: 27 June 2023

Abstract
Crowdsourcing is a process of engaging a '*crowd*' or a group of common people for accomplishing the tasks. In this work, the time-bound tasks allocation problem in IoT-based crowdsourcing is investigated in *strategic* setting. The proposed model consists of multiple task providers (or task requesters) and several IoT devices (or task executors), and each of the task providers carries a task that have start time and completion time. Each of the participating IoT devices provide a preference ordering (order of their interest for the tasks) over a subset of tasks. Given the time bound tasks and ranking (or preference ordering) of the task executors, the objectives are: (1) to assign the tasks to different slots so that they are non-conflicting in nature, and (2) to allocate at most one task to each of the task executors from their respective preference ordering. To achieve the above objectives, a *truthful* mechanism is proposed namely Truthful Mechanism for Time-bound Tasks in IoT-based Crowdsourcing (TMTTC). Through theoretical analysis, it is proved that TMTTC satisfies the properties such as *computational efficiency*, *truthfulness*, *Pareto optimality*, and *The Core*. Through simulation, it is shown that TMTTC performs better than benchmark mechanism on the ground of *truthfulness*.

Keywords Crowdsourcing · Scheduling · Truthful · IoT devices · Zero budget

1 Introduction

Crowdsourcing is said to be an open call to the group of common people (or community) for accomplishing one or more task(s) [4, 20, 25, 27]. In crowdsourcing, the works (or tasks) that is to be done are posted to some public domain (or platform) and the crowd workers execute the tasks and submit it to the platform (or third party). The platform provide the completed tasks to the task providers and incentives are given to the crowd workers in return of their

✉ Vikash Kumar Singh
vikash.singh@vitap.ac.in

Sanket Mishra
sanket.mishra@vitap.ac.in

[1] School of Computer Science & Engineering, Vellore Institute of Technology, Amaravati, Andhra Pradesh, India

services. However, when the posted tasks are accomplished by the community with the help of their smart devices (such as mobile phone etc.) then it is termed as *Mobile crowdsourcing* (or *Participatory sensing*) [10, 13, 35]. Internet and the advancement of new technologies have been the catalyst for these type of research over the last decade and have found many application areas (such as healthcare [32], agriculture etc. [8]), thereby creating both the theoretical and the applied research flavor. However, most of the works that are carried out in past in *crowdsourcing* [16, 17, 19, 24, 34] mainly focused on *how to motivate a mass of task executors (or crowd workers) for participating in the system?* One of the solutions that has been provided from large community is to incentivize the participating agents (task executors and task providers) for their services. However, the proposed solutions of dragging large group of common people to the crowdsourcing market generates several other open questions, such as:

1. *How to decide that which agents are to be considered for performing the tasks?*
2. *How to have the information about the good (or quality) crowd workers?*
3. *What price is to be offered to the crowd workers in exchange of their services?*
4. *How to have the services of the crowd workers those are socially motivated?*

In past, several works have been carried out in the directions pointed out in points 1−3 above. In [2, 9, 14, 15, 34] some incentive schemes are proposed for motivating the crowd workers. In [2] the crowdsensing platform publishes the set of sensing tasks to the outside world along with the location of the sensing tasks. Each of the floated sensing tasks is to be sensed during the given time period by the vehicles having sensing devices (acting as crowd workers). For this purpose an incentive based mechanism is developed that is based on *proportional share mechanism* [29]. In [15] the mechanism is developed that along with providing the incentives to the participants helps in protecting the privacy about the location of the participating agents. In past, several *truthful* (or *Incentive Compatible (IC)*) mechanisms (refer Section 3 for definition of truthful) for the crowdsourcing environment is proposed in [3, 6, 9, 34]. However, other than the above discussed scenarios, it may be the case that some crowd workers may be socially motivated and are willing to provide their services to the task providers free of cost. In past, the scenario depicted in query 4 above has not been addressed. In this paper, we have addressed the situation mentioned in query 4 above by utilizing the concept of *mechanism design without money* or under *zero budget* environment [5, 26, 28].

The detailed overview of the proposed framework is shown in Fig. 1.

In the proposed framework there are multiple task providers and several IoT devices. Each of the task providers is having a task with him/her (henceforth him) that is to be completed. The tasks held by the task providers have start time and completion time associated with them. Here, start time of a task means that the time at which the task is available in the

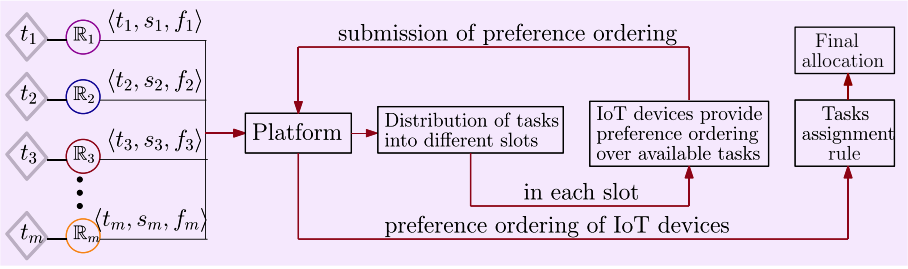


Fig. 1 Proposed framework for IoT-based Crowdsourcing

market whereas completion time of a task is the time by which the task must be completed. The task providers submit the tasks with the additional information *i.e.* start and completion times of each of the tasks to the third party. Once the platform receives the tasks, it puts the available tasks into different slots in such a way that there is no conflict between any two tasks in a given slot (by utilizing the mechanism discussed in subsection 4.1). Now, each slot contains the tasks and a set of task executors, where the task executors provide ranking over all the available tasks or subset of tasks. The task executors may be indifferent between two tasks or in other words the preference lists (or rank lists) may have *ties*. The preference ordering of the task executors is *private* in nature. It means that, the preference ordering of each of the task executors is only known to him and not known to others. In the proposed framework, the IoT devices that are taking part in the crowdsourcing system are *strategic* in nature. Due to this, the IoT devices can misreport their preference ordering for obtaining better task. Given the above discussed set-up the objectives are: (1) to place the available tasks to multiple slots in order to avoid conflict between the tasks, (2) to allocate at most one task to each of the task executors from their reported preference ordering. For these purposes, by utilizing an idea of *mechanism design without money* an *incentive compatible* mechanism is proposed namely **T**ruthful **M**echanism for **T**ime-bound **T**asks in IoT-based **C**rowdsourcing (TMTTC) motivated by [11, 21, 23, 28]. By mechanism design without money it is meant that, no monetary transfers are necessary for accomplishing the desired works.

The key contributions of this paper are:

- First the tasks provided by the task providers along with their start and completion times are distributed to different slots to ensure that in each slot there is no overlapping tasks. It is required as it will help the task executors to show interest over multiple tasks.
- The time-bound task allocation problem in IoT-based crowdsourcing is cast as a mechanism design without money problem or in *zero* budget environment.
- A *truthful* mechanism is proposed for allocating tasks to the IoT devices, that also ensures the placement of tasks in a non-conlict manner to different slots.
- Through theoretical analysis it is shown that TMTTC is *computationally efficient*, *truthful*, *Pareto optimal*, and satisfy the *Core* property.
- The simulations are carried out to show that, if in case of TMTTC the participating agents tries to manipulate the private information (in this paper preference ordering) then they cannot gain. Also, the proposed mechanism is compared with the benchmark mechanism where the agents can gain by misreporting their preference ordering.

The remainder of this paper is organized as follows. In section 2 the related works are discussed. The proposed system model is discussed in detailed manner in Section 3. In Section 4, the proposed mechanism is discussed and presented. An example is illustrated for better understanding of TMTTC in Section 5. In Section 6, the analysis of TMTTC is carried out. The experiments are carried out and obtained results are discussed in Section 7. In section 8 the conclusion of the paper is drawn and future works are discussed.

2 Related prior works

In this section the discussion about the recent development that is carried out in the field of crowdsourcing is done. To have an overview about the works carried out in crowdsourcing, readers can go through [4, 19, 24, 27, 31, 33, 37].

In [31] the discussion is mainly focused on the major challenges in crowdsourcing. In [19], in order to address some of the issues in crowdsourcing a framework is developed

that automate the volunteer selection and matches the criteria of volunteers and tasks. In [24] for identifying the crowd workers that provide the accurate data, the reputation-based incentive mechanism is designed. The crowd workers those who furnish these accurate data are called as the *reputable workers*. It is to be noted that there is high chance that these reputable crowd workers will be getting the rewards and also by following the reputable workers the non-reputable workers can gain reputation. In [1], to have a track about the reliable crowd workers an efficient and dynamic approach is proposed. In [27] in order to improve conventional crowdsourcing system and to develop future crowdsourcing system a systematic survey of crowdsourcing is carried out that focuses on emerging techniques and approaches.

Several incentive based mechanisms are designed that somehow incentivizes the agents in exchange of their services [7, 9, 15, 17, 30, 34]. In [7], the discussed set-up is studied in online environment where the task providers and executors are arriving and departing from the system continuously. Here, the task executors report costs for performing the tasks and is *private* in nature. Along with the private cost each task executor has different skill set that make him eligible to show preference over subset of tasks for execution. For the discussed set-up in [7] an *incentive compatible budget feasible* mechanism is proposed. In [17] an optimal mechanism (one that maximizes the crowdsourcing revenue minus cost) is proposed for incentivizing the crowd workers that is following the principle of all-pay auction. In [40] an incentive compatible mechanism is proposed for mobile crowdsourcing that combines the idea of reverse auction and multi-atribute auction.

In [9, 30, 34] along with providing the incentives to the crowd workers the focus was to have a set of quality agents (or crowd workers) for performing the tasks. In [34] there are multiple task providers and multiple task executors. Each task provider is endowed with a task and a fixed budget for each of the tasks. For this set-up a *truthful budget feasible* mechanism is designed that also take care of determining the quality task executors. In [30], an incentive compatible mechanism is designed so as to assign the subset of quality IoT devices to each of the tasks from the set of tasks of their interest. The assignment should be done in such a way that there exist no conflict between the tasks and also the summed-up value of the bid values of the winning task executors is maximum. In [9], the discussed set-up is, there are multiple IoT devices and a task requester, where the task requester floats multiple tasks that are to be executed. Also, the task requester is having budget that arrives in an incremental way in several phases for all the available tasks. A budget feasible mechanism is designed for the discussed set-up that also satisfy one of the important economic properties such as *truthfulness*.

In [36] an incentive based mechanism is proposed where the incentives are provided to the workers on the basis of quality of sensed data provided by them and not based on the time they have spent on completing the task. In [38], an incentive compatible mechanism is developed for revealing their quality, effort, and data in true sense. In [39], the tasks that are floated by the task providers have some fixed deadlines. The task executors shows the interest over the subset of tasks that he can complete based on his skills and on that basis tasks are assigned to the task executors before its deadline. For this purpose a mechanism is designed that results in maximum overall expected payoff subject to deadline and budget constraints. In [18] for detecting the quality crowd workers the peer review process is carried out. It means that the tasks completed by crowd workers is provided to peer crowd workers for reviewing and based on their recommendation the quality crowd workers are identified.

However, the problems discussed in the aforementioned works are mainly studied in *monetary* environment. By *monetary* environment, it is meant that the money is involved in the market in some sense. In this paper, first time the problem in crowdsourcing or more

specifically in IoT-based crowdsourcing is studied in *zero budget* environment. By *zero budget*, it is meant that money is not present in the market in any sense. Here, the crowd workers are *socially* motivated to provide their services free of cost. In Section 3, the discussed problem in this paper is formulated through the lens of mechanism design without money.

3 System model and problem formulation

In this set-up, the set of n task executors is given as $\mathbb{T} = \{\mathbb{T}_1, \mathbb{T}_2, \ldots, \mathbb{T}_n\}$, where $\mathbb{T}_i \in \mathbb{T}$ denotes i^{th} task executor. Also, there are m task providers and is given as $\mathbb{R} = \{\mathbb{R}_1, \mathbb{R}_2, \ldots, \mathbb{R}_m\}$, where $\mathbb{R}_i \in \mathbb{R}$ denotes i^{th} task provider. Here, m and n may or may not be equal ($i.e.$ $m > n$ or $m < n$ or $m = n$). Each of the task providers $\mathbb{R}_i$ has a task t_i that is to be completed. The set of tasks that is available in the market is represented as $t = \{t_1, t_2, \ldots, t_m\}$. In the proposed model, each of the tasks t_i has start time represented as s_i and completion time represented as f_i, where $f_i \geq s_i$. The set of start and completion times for all the tasks is given as $s = \{s_1, s_2, \ldots, s_m\}$ and $f = \{f_1, f_2, \ldots, f_m\}$ respectively. It may be the case that the two tasks t_j and t_k overlaps. We say that the two tasks t_j and t_k overlaps, when one of the following occurs: (1) $s_j \leq s_k \leq f_j \leq f_k$, or (2) $s_k \leq s_j \leq f_k \leq f_j$, or (3) $s_k \leq s_j \leq f_j \leq f_k$, or (4) $s_j \leq s_k \leq f_k \leq f_j$. The discussed model in this paper is studied as a two fold process. In the first fold, all the tasks along with the task providers are placed in minimum number of slots so that no two tasks are overlapping in any of the slots (refer to example discussed in Section 5 for understanding the task distribution in slots.). Once, the available tasks are placed to multiple slots in non-overlapping manner, then in the second fold, each of the task executors give the preference ordering over the available tasks. The preference ordering of i^{th} task executor is represented by $\succ_i$ over a set of tasks t. $t_j \succ_i t_k$ means that the task executor $\mathbb{T}_i$ prefers the task t_j over the task t_k. The ties in the preference list of i^{th} task executor is represented by $=_i$ over a set of tasks t. Here, $t_j =_i t_k$ means that the task executor $\mathbb{T}_i$ prefers t_j and t_k equally. The set of preference of all the task executors is denoted by $\succ = \{\succ_1, \succ_2, =_3, \ldots, \succ_n\}$. In the proposed framework, the task executors are *strategic*. It means that, the task executors may gain by not providing the privately held preference ordering in truthful manner. By *private* it means that the preference ordering is only known to him and not known to others. Given the ranking over the tasks by the task executors, the TMTTC assigns at most one task to each of the task executors. The resultant allocation vector is given as $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$, where $\mathcal{A}_k \in \mathcal{A}$ is given as $(\mathbb{T}_k, t_k)$. Let us discuss the required concepts and definitions before moving forward.

Definition 1 [**Blocking coalition** [22, 26]] If some of the IoT devices among the available ones form a coalition and via some internal reallocation the IoT devices are making themselves better off then that coalition will be said to be *blocking coalition*.

Definition 2 [**Core allocation** [22, 26]] The allocation resulted by mechanism is said to be *core allocation* if in an allocation any subset of task executors form a coalition and with some internal reallocation only some of the members of the coalition are made themselves better off and not all the members.

Definition 3 [**Truthfulness** [22, 26]] A mechanism is said to be truthful, if the participating agents are not able to gain by misreporting their preference ordering.

Definition 4 [**Pareto optimality** [22, 26]] An allocation is *Pareto optimal*, if in that allocation no agent can be made better off without making someone else worse off.

Definition 5 [**Computational efficiency**] An incentive compatible mechanism is said to be computationally efficient if all the steps is carried out in polynomial time.

4 Proposed mechanism: TMTTC

In this section, TMTTC motivated by [12, 21, 23, 28] is discussed. The TMTTC consists of two components: (1) *Tasks distribution rule*, and (2) *Tasks assignment rule*. In the upcoming subsections the discussion of the components of TMTTC is carried out in detailed manner.

4.1 Tasks distribution rule

The *Tasks distribution rule* is motivated by [12, 30].

4.1.1 Outline of tasks distribution rule

The objective of *Tasks distribution rule* is to schedule all the tasks into slots so that no two tasks in a slot should be overlapping with each other. The outline of the mechanism is that, first the ordering of the available tasks is done in increasing order of their start time. Now, from the ordered list, a task is selected and is placed in the slot to which it is compatible with the available tasks. The process is continued until all the available tasks are processed.

- Order all the available tasks in increasing order of their start time.
- If ties occur, select a task randomly from the ordering.
- Initially, no slot is allocated to any task.
- **while** (each task is not processed):
  - **If** selected task is appeared to be compatible with any of the available slots, then place the task to that available slot.
  - **Otherwise** a new slot is considered and the task is allocated to that new slot.

4.1.2 Detailing of tasks distribution rule

The inputs to *tasks distribution rule* is a set of tasks t, and set of start time s. In line 1, the available tasks are ordered in increasing order of their start time. Once ordered, the tasks are held in t. Now, using *while* loop in line 2-12 the tasks are placed in the slots in such a way that there is no conflict among the task in any given slot. In line 3, a task is selected from a set of tasks and is stored in $\hat{t}$. In line 4, a check is made if the selected task is compatible with any of the slots then it will be placed in that slot otherwise a task is placed in new slot. Line 2-12 iterates until all the tasks are processed.

4.2 Tasks assignment rule

The *Tasks assignment rule* is motivated by [22, 26].

4.2.1 Outline of tasks assignment rule

The objective of *Tasks assignment rule* is to allocate the most preferred task to the task executors. The idea of the mechanism is, first the random numbers are generated from 1 to n

Algorithm 1 Tasks distribution rule (t, s).

```
1  t ← Sort(t, s) ;                                   // Sort t based on start time.
2  while t ≠ φ do
3      t̂ ← Select (t)
4      if task in t̂ is compatible with some slot then
5          Schedule task in t̂ in any such slot.
6      end
7      else
8          Assign a slot ℓ + 1.
9          Place the task present in t̂ to slot ℓ + 1.
10         ℓ ← ℓ + 1.
11     end
12 end
```

and are assigned to IoT devices. After that the task executors are ordered in ascending order of the random number allocated. From the ordered list, a task executor is considered each time and the best available task is allocated from his preference list. The process ends once every task executor is processed.

Tasks assignment rule In each slot:

1. First of all, n distinct numbers are generated randomly and are assigned to the task executors.
2. After that, the task executors are placed in increasing order of random number slapped on them.
3. A task executor is selected from the ordered list and is checked that, whether the rank list provided by the task executor contains any task or not;
   - **If** the preference list is not empty, then from his preference list the top most task is allocated to him among the available one. The allocated task along with the task executor is removed from the crowdsourcing market.
   - **Otherwise**, delete the unallocated task executor from the list of task executors.
4. Step 3 is repeated till task executors list get exhausted.

4.2.2 Detailing of tasks assignment rule

In line 2, variable k is initialized to 0, and the variables $\hat{t}$, $\hat{r}$, and $\mathcal{G}$ are initialized to ϕ. Line 3-12 the random numbers are generated and are assigned to the IoT devices. In line 13, the set of task executors are ordered in increasing order of the number assigned randomly. In line $14-27$ the allocation of tasks is made to task executors. In line $15-22$, from the ordered list of IoT devices, sequentially an IoT device is considered and is checked that, whether the rank list provided by the task executor contains any task or not. If the preference list is not empty then the most preferred task will be assigned to the selected IoT device. Once the allocation is done the allocated task executor and task will be removed. If the rank list is empty then the task executor is deleted from the task executors list using line $23-26$. In line 28 the final allocation is returned.

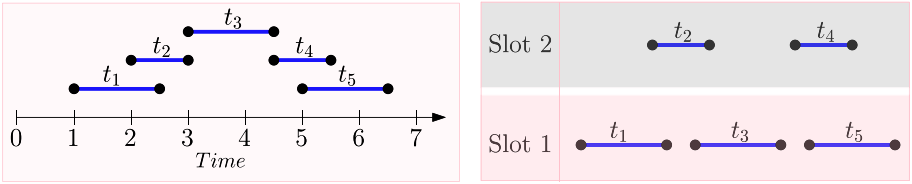
Algorithm 2 Tasks assignment rule $(\mathbb{T}, \mathbb{R}, \succ)$.

Output: $\mathcal{A} \leftarrow \phi$
1 **begin**
2 $k \leftarrow 0, \hat{t} \leftarrow \phi, \hat{r} \leftarrow \phi, \mathcal{G} \leftarrow \phi$
3 **for** $i \leftarrow 1$ *to* n **do**
4 $\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{i\}$
5 **end**
6 **for** $i \leftarrow 1$ *to* n **do**
7 $\quad$ swap $\mathcal{G}[i]$ with $\mathcal{G}[Random(i, n)]$
8 **end**
9 **for** *each* $\mathbb{T}_i \in \mathbb{T}$ **do**
10 $\quad Assign\_number\ (\mathbb{T}_i, \mathcal{G}[k])$
11 $\quad k \leftarrow k + 1$
12 **end**
13 $\mathbb{T} \leftarrow$ Sort$(\mathbb{T}, \mathcal{G})$
14 **while** $\mathbb{T} \neq \phi$ **do**
15 $\quad \hat{t} \leftarrow pick\_TE\ (\mathbb{T})$ /* where, $i = 1, 2, \ldots, n$ */
16 $\quad$ **if** $\succ_i \neq \phi$ **then**
17 $\quad\quad \hat{r} \leftarrow Select\_most\_preferred\ (\succ_i)$
18 $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \cup (\hat{t}, \hat{r})$
19 $\quad\quad \mathbb{T} \leftarrow \mathbb{T}_i \setminus \hat{t}; \mathbb{R} \leftarrow \mathbb{R}_i \setminus \hat{r}$
20 $\quad\quad \succ \leftarrow \succ_i \setminus \hat{r}, \mathbb{T}_i \in \mathbb{T}$
21 $\quad\quad \hat{t} \leftarrow \phi; \hat{r} \leftarrow \phi$
22 $\quad$ **end**
23 $\quad$ **else**
24 $\quad\quad \mathbb{T} \leftarrow \mathbb{T}_i \setminus \hat{t}$
25 $\quad\quad \hat{t} \leftarrow \phi$
26 $\quad$ **end**
27 **end**
28 return $\mathcal{A}$
29 **end**

5 Example elaborating TMTTC

In this section TMTTC is elaborated with the help of an example. The initial configuration of the tasks along with their start time and completion time is depicted in Fig. 2a. Let us apply Algorithm 1 to the set-up depicted in Fig. 2a. So, first the available tasks are ordered in increasing order of their start time and sorted ordering obtained is given as: $t_1 \leq t_2 \leq t_3 \leq t_4 \leq t_5$. So, first task t_1 is considered and is placed to slot 1. Next task t_2 will be picked up and the check in line 5 will be false in this case so following line 7-11 of Algorithm 1 a new slot will be allocated to task t_2. After that, task t_3 is picked up from the ordering and is scheduled in slot 1. Next task t_4 is considered and is assigned to slot 2. Finally, task t_5 will be picked from the ordering and will be assigned to slot 1. So, the tasks distribution obtained is shown in Fig. 2b.

Let us consider Slot 1 to understand Algorithm 2. In Fig. 2b it can be seen that in Slot 1 we have 3 tasks and say we have 4 task executors given as $\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$, and $\mathbb{T}_4$. The preference ordering of the task executors is depicted in Fig. 3a. Using line 3-12 of Algorithm 2 four distinct numbers are generated randomly and are assigned to the task executors as shown in Fig. 3b. The task executor $\mathbb{T}_3$ is picked up and task t_3 is allocated, as it was the most preferred task in the preference ordering of $\mathbb{T}_3$. Now, both the task executor $\mathbb{T}_3$ and task t_3 are removed from the market. Next, task executor $\mathbb{T}_1$ is picked up and the most preferred task t_1 is allocated to him. So, task executor $\mathbb{T}_1$ and task t_1 are removed from the market. From



(a) Initial tasks configuration (b) Non-Overlapping tasks

Fig. 2 Illustration of Tasks distribution rule

the ordering, next task executor $\mathbb{T}_2$ is picked up but as his preference list is empty no task is allocated to him. Finally task t_4 is picked up and the most preferred task t_2 is allocated. Final allocation is depicted in Fig. 3c.

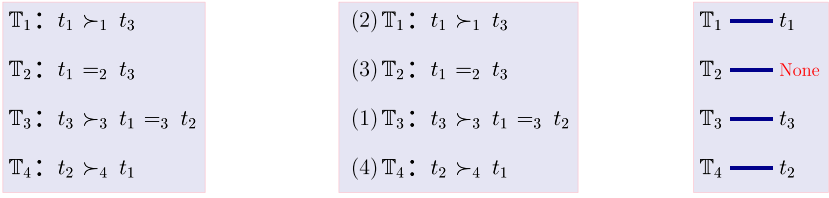
6 Analysis of TMTTC

In this section, first in Theorem 1 it is shown that TMTTC is computationally efficient *i.e.* running time of TMTTC is $O(n^3)$. TMTTC is utilizing the framework of *The Draw* [22]. The Draw exhibits two important properties *truthfulness* and *Pareto optimality*. First, the two propositions regarding *truthfulness* and *Pareto optimality* of *The Draw* is presented and then it is proved that TMTTC is *truthful* (Lemma 1) and *Pareto optimal* (Lemma 2). Further in Lemma 3 it is proved that the allocation resulted by TMTTC is the *unique Core allocation*. In Lemma 4 and 5 the probabilistic analysis is carried out. In Lemma 4, it is shown that, the expected number of task executors (or IoT devices) getting their most preferred task (*i.e.* first preference) is $\frac{n_k}{2}$, where n_k is the number of IoT devices in any k^{th} slot. In Lemma 5, the probability that at least $\frac{2n_k}{3}$ task executors are getting their most preferred task is less than or equal to $\frac{3}{4}$.

Theorem 1 *The computational complexity of TMTTC is* $O(n^3)$.

Proof The computational complexity of TMTTC will be the sum of the computational complexities of *Tasks distribution rule* (Algorithm 1) and *Tasks assignment rule* (Algorithm 2). The computational complexity of the components of TMTTC is shown below:

– **Tasks distribution rule:** Line 1 will take $O(m \lg m)$ time. The body of *while* loop will execute for m times. For each iteration of *while* loop, the body of the *while* loop will end up in $O(m)$. So, while loop in line 2-12 of Algorithm 1 takes $O(m^2)$. So, the computational complexity of *Tasks distribution rule* is $O(m \lg m) + O(m^2) = O(m^2)$.



(a) Preferences of IoT devices (b) Assignment of random numbers (c) Final allocation

Fig. 3 Illustration of Tasks assignment rule

– **Tasks assignment rule:** The random number generation in line $3-12$ takes linear time *i.e.* $O(n)$. Line 13 will take $O(n \lg n)$. The *while* loop in line $14-27$ takes $O(n^3)$. So, the running time of *Tasks assignment rule* is given as $O(n) + O(n \lg n) + O(n^3) = O(n^3)$.

The computational complexity of TMTTC is $O(m^2) + O(n^3) = O(n^3)$.

Proposition 1 *The Draw is truthful [22].*

Proposition 2 *The allocation resulted by Draw is Pareto optimal [22].*

Lemma 1 *TMTTC is truthful.*

Proof Fix a slot i. In TMTTC the numbers generated randomly and assigned to the task executors has no relation with the preference lists of the task executors over the tasks. It is meant that, the task executors considered in any iteration is independent of the random number assigned to him. From the construction of TMTTC, in whichever iteration a task executor is considered, he will be getting his best task from the available ones. So, if any task executor is manipulating his preference ordering then in that case either he will be getting the similar task (which he could have got when he would have reported the true preference ordering) or worse than that. From above argument it is clear that it is not beneficial for the task executors to misreport their ranking over the tasks. So, it will be true for all the task executors in different slots. Hence, TMTTC is truthful.

Lemma 2 *The allocation resulted by TMTTC is Pareto optimal.*

Proof Fix a slot i. From the construction of TMTTC, in any k^{th} iteration the task executor under consideration gets the top ranked task from his ranking over the tasks. As a thought experiment let us run TMTTC in parallel with some other mechanism. The objective here is to show that the allocation resulted by some other mechanism is same as TMTTC and if not then the other mechanism has worsen the allocation of some of the task executors. So, this makes TMTTC Pareto optimal.

The proof can further be carried out by utilizing the idea of mathematical induction. So, before the 1^{st} iteration both TMTTC and other mechanism has not done any allocation so the winning set is empty for both the mechanisms. Let us say till p^{th} iteration the allocation resulted by TMTTC and other mechanism is same. It means that the available set of tasks and the task executors for TMTTC and other mechanism is same for $(p+1)^{th}$ iteration. So, in case of TMTTC if any task executor $\mathbb{T}_i$ is considered then he will get his most preferred task from the available set of tasks. However, if the other mechanism allocates the task to task executor $\mathbb{T}_i$ other than that allocated by TMTTC then it means that the task executor $\mathbb{T}_i$ is worsen off in case of other mechanism. If not, then the two mechanisms would have given same set of allocation, which is *optimal*. So, it will be true for all the task executors in different slots. Hence, TMTTC results in an allocation that is *Pareto optimal*.

Lemma 3 *The allocation resulted by TMTTC is the unique core allocation.*

Proof Fix a slot i. In Lemma 2 it is already proved that the allocation resulted by TMTTC is *unique*. Here, the goal is to just prove that the unique allocation obtained is "*The Core*". Suppose the two task executors *i.e.* $\mathbb{T}_i$ and $\mathbb{T}_j$ form a coalition and by their mutual collaboration both of them reports the preference ordering other than the true preference ordering. As it is already discussed that the random number that are assigned to the task executors is independent of their preference list. So, if any of the task executors *i.e.* $\mathbb{T}_i$ or $\mathbb{T}_j$ is considered

at any point of time then he will be allocated the best available task from their respective ranking. But, if by mutual collaboration they have misreported their ranking then in that case either they will be getting the task that they would have got when reported truthfully or worse than that. So, one can conclude that the task executors cannot gain by mutual collaboration. From above claim it can be true for any task executor in any slot. So, TMTTC results in *core allocation*. Hence, TMTTC results in *unique core allocation*.

Lemma 4 *In any given slot say k, the expected number of IoT devices (or task executors) assigned their most preferred task (i.e. first preference) is given as* $\frac{n_k}{2}$*, where* n_k *is the available number of IoT devices in any* k^{th} *slot. In more formal way, it can be written as* $E[Y_k] = \frac{n_k}{2}$*. Here,* Y_k *is the random variable that determines the total number of IoT devices allocated their first preference.*

Proof Fix any slot k. In this, the main objective is to have an estimate on the expected number of IoT devices allocated their most preferred task. Here, Y_k is a random variable that keep track of total number of IoT devices getting their most preferred task in slot k. So, in slot k the expected number of IoT devices that are allocated their most preferred tasks is given by $E[Y_k]$. Before moving forward let us first have an estimate on the probability that any l^{th} IoT device will be getting their first preference. Let $\mathbb{U}$ be the event that any l^{th} task executor will not be getting his first preference and this will take place only when it has been already taken up by any of the $l-1$ task executors appearing before l^{th} task executor. Let $\mathbb{U}_i$ be the event that any task executor $\mathbb{T}_i$ from $l-1$ task executors are having the same first preference as l^{th} task executor. So, it can be written as:

$$Pr\{\mathbb{U}\} = \sum_{i=1}^{l-1} Pr\{\mathbb{U}_i\} \tag{1}$$

Now, the probability that the l^{th} task executor's first preference will be the first preference of any task executor appearing before l^{th} task executor is $\frac{1}{n_k}$ (as it is equally likely). Putting the value as $\frac{1}{n_k}$ in equation 1,

$$Pr\{\mathbb{U}\} = \sum_{i=1}^{l-1} \frac{1}{n_k} = \frac{l-1}{n_k} \tag{2}$$

Let $\bar{\mathbb{U}}$ be the event that the first preference of any l^{th} agent will not be taken by any of the task executor appearing before him. So, we have

$$Pr\{\bar{\mathbb{U}}\} = 1 - Pr\{\mathbb{U}\} = 1 - \left(\frac{l-1}{n_k}\right) \tag{3}$$

Now, let Y_{kl} be the indicator random variable and is defined as: $Y_{kl} = I\{l^{th}$ task executor in k^{th} slot gets his first preference$\}$

$$I = \begin{cases} 1, & if\, l^{th}\, task\, executor\, gets\, his\, first\, preference \\ 0, & \text{otherwise} \end{cases}$$

As defined already that Y_k keep track of total number of IoT devices getting their most preferred task in slot k. We have,

$$Y_k = \sum_{l=1}^{n_k} Y_{kl} \tag{4}$$

Taking expectation both side in equation 4, we get;

$$E[Y_k] = E\left[\sum_{l=1}^{n_k} Y_{kl}\right]$$

By linearity of expectation, we get

$$E[Y_k] = \sum_{l=1}^{n_k} E[Y_{kl}]$$

As the expectation of the random variable is equal to its probability, so we have

$$E[Y_k] = \sum_{l=1}^{n_k} Pr\{Y_{kl}\} = \sum_{l=1}^{n_k} Pr\{\bar{\mathbb{U}}\}$$

$$= \sum_{l=1}^{n_k} \left(1 - \left(\frac{l-1}{n_k}\right)\right) = \sum_{l=1}^{n_k} 1 - \sum_{l=1}^{n_k} \left(\frac{l-1}{n_k}\right)$$

$$= \sum_{l=1}^{n_k} 1 - \sum_{l=1}^{n_k} \frac{l}{n_k} + \sum_{l=1}^{n_k} \frac{1}{n_k}$$

$$= n_k - \left(\frac{n_k \cdot (n_k+1)}{2 \cdot n_k}\right) + \left(\frac{n_k}{n_k}\right)$$

$$= \frac{n_k}{2} + \frac{1}{2} \simeq \frac{n_k}{2}$$

From here, one can infer that in expectation half the total number of IoT devices will be allocated their most preferred task.

observation 1 If we consider the value of n_k as 200 then we get $E[Y_k] = \frac{n_k}{2} = \frac{200}{2} = 100$. So, in expectation 50% of the available IoT devices will end up getting their most preferred (or first preference) tasks from their preference ordering.

Lemma 5 *For any slot say k, the probability that at least $\frac{2n_k}{3}$ task executors are getting their most preferred task (i.e. first preference) is less than or equal to $\frac{3}{4}$. In other words, it can be written as:*

$$Pr\left\{Y_k \geq \frac{2n_k}{3}\right\} \leq \frac{3}{4}$$

Here, Y_k keep track of total number of IoT devices getting their most preferred task in slot k.

Proof From the above lemma (*i.e.* Lemma 4), we have Y_k is the random variable that keep track of total number of IoT devices getting their most preferred task in slot k. It is represented as I = {number of task executors allocated their first preference}

$$I = \begin{cases} 1, & if\, Y_k \geq \frac{2n_k}{3} \\ 0, & \text{Otherwise} \end{cases} \tag{5}$$

From equation 5, we can write

$$Y_k \geq \frac{2n_k}{3} \implies 1 \leq \frac{Y_k}{\frac{2n_k}{3}}$$

i.e.

$$I \leq \frac{Y_k}{\frac{2n_k}{3}} \tag{6}$$

Taking expectation both side of equation 6, we get

$$E[I] \leq E\left[\frac{Y_k}{\frac{2n_k}{3}}\right]$$

$$= \frac{1}{\frac{2n_k}{3}} \cdot E[Y_k]$$

$$E[I] \leq \frac{3}{2n_k} \cdot E[Y_k] \tag{7}$$

$$Pr\left\{Y_k \geq \frac{2n_k}{3}\right\} \cdot 1 \leq \frac{3}{2n_k} \cdot E[Y_k] \tag{8}$$

From Lemma 4 the value of $E[Y_k]$ is substituted as $\frac{n_k}{2}$ in equation 8,

$$Pr\left\{Y_k \geq \frac{2n_k}{3}\right\} \cdot 1 \leq \frac{3}{2n_k} \cdot \frac{n_k}{2} \tag{9}$$

Equation 9 above can be written as:

$$Pr\left\{Y_k \geq \frac{2n_k}{3}\right\} \leq \frac{3}{4}$$

Hence proved.

observation 2 From Lemma 5, one can infer that if n_k value is taken as 150 then with probability at most 0.75 more than 100 task executors will be allocated their most preferred tasks.

7 Experiments and results

In this section, the experiments are carried out to compare TMTTC with the benchmark mechanism (*i.e.* Random) that is *non-truthful* in nature. For the simulation purpose the data (here, preference ordering of the task executors) has been generated synthetically using Python libraries. In order to strengthen our claim, the simulation is carried out for two different dataset, namely *small dataset* (see Table 1) and *large dataset* (see Table 2). It is to be noted that the benchmark mechanism differ only in terms of *Tasks assignment rule* from TMTTC, the *Tasks distribution rule* is same for both the mechanisms. The underlying idea of *Tasks assignment rule* of Random mechanism is given below. For any given slot,

1. Each time a task executor is picked up from the set of task executors.
2. After that the preference ordering of the selected task executor is checked whether it is empty or not.
3. If preference list is not empty, then
   - From the preference list of the respective task executor a task is randomly selected and is assigned to the task executor.

Table 1 Small data set utilized for comparing TMTTC with Random mechanism

Cases	Number of IoT devices	Preference Ordering Generated
$n = m$	25, 50, 75, 100, 125, 150	Randomly (using Python libraries)
$n < m$	20, 45, 70, 95, 120, 145	Randomly (using Python libraries)
$n > m$	25, 50, 75, 100, 125, 150	Randomly (using Python libraries)

– Once allocated, both the task executor and the task are removed from crowdsourcing market.

4. Else, a task executor is removed from the market.
5. Steps 1-4 are repeated until each of the task executors are processed.

In simulation, the manipulative behavior of IoT devices is taken into consideration. The two mechanisms are compared based on the following parameters: (1) *Number of Preferred Allocation* (NPA) – Number of task executors getting their first preference from their reported preference ordering, and (2) *Utility of Task Executors* (UTE) – It is the difference between the position of the task allocated to him from his preference ordering and the position of the most preferred task in his preference ordering.

It is to be noted that, for the mechanism that results in higher value of NPA and lower value of UTE will be considered as good mechanism.

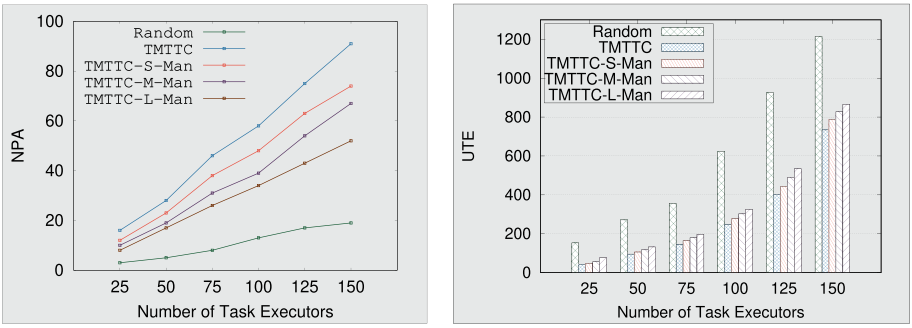
7.1 Simulation setup

In simulation the tasks and task executors are considered from 5 different slots. In any given slot one of the following situations could be pertaining: (1) number of tasks is equal to the number of task executors ($i.e$ m = n), (2) number of tasks is greater than the number of available task executors ($i.e.$ m > n), and (3) number of tasks is less than the number of task executors ($i.e.$ m < n). In all the three situations, the task executors are providing the preference ordering over the tasks. In this paper, the simulation is carried out considering two cases: (1) all the participating task executors are revealing their true ranking, and (2) some subset of the participating task executors are mis-reporting their ranking. More specifically, for the direction mentioned in point 2 above the below mentioned manipulation criteria is followed for the simulation purpose.

1. **Small manipulation (S-Man):** $\frac{1}{10}$ of the task executors are mis-reporting their ranking.
2. **Medium manipulation (M-Man):** $\frac{1}{5}$ of the task executors are mis-reporting their ranking.
3. **Large manipulation (L-Man):** $\frac{1}{2}$ of the task executors are mis-reporting their ranking.

Table 2 Large data set utilized for comparing TMTTC with Random mechanism

Cases	Number of IoT devices	Preference Ordering Generated
$n = m$	500, 1000, 1500, 2000, 2500, 3000	Randomly (using Python libraries)
$n < m$	400, 800, 1200, 1600, 2000, 2400	Randomly (using Python libraries)
$n > m$	700, 1200, 2100, 2800, 3500, 4200	Randomly (using Python libraries)



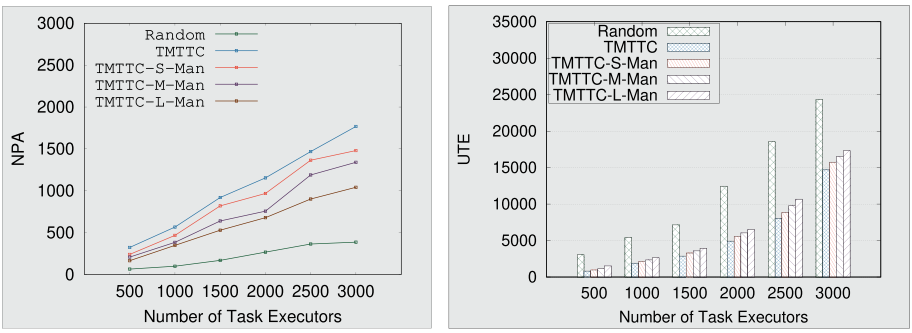
(a) Comparison based on NPA (b) Comparison based on UTE

Fig. 4 Comparison based on NPA and UTE (m=n) for small dataset

As a test case, the simulation is carried out for large data set also, where the number of IoT devices could be bounded above by 4200. The preference ordering of each of the IoT devices is generated randomly. Table 2 depicts the data utilized for comparing TMTTC with Random mechanism.

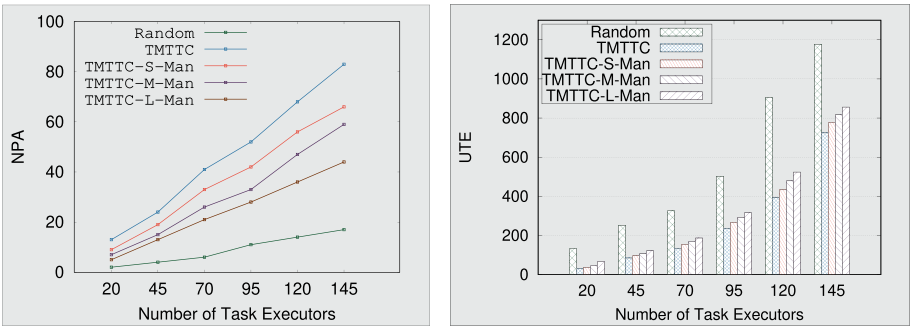
7.2 Result analysis

As the simulation is carried out for three different situations, so the results are discussed considering all the three situations and for both the data sets. The discussion will mainly circumvent around the parameters mentioned in Section 7. The results obtained for the three different situations are depicted in Figs. 4–9. In Figs. 4a, 5a, 6a, 7a, 8a, and 9a x-axis represents the number of task executors and y-axis represents NPA. The NPA value in case of TMTTC is higher than the NPA value in case of Random for all the three situations. It is due to the reason that, each time, in case TMTTC each of the task executors is allocated with the best available tasks. On the other hand, in case of Random the task executors may not be allocated with the best available tasks. In this, the tasks are picked-up randomly from the preference ordering of task executors and allocated.



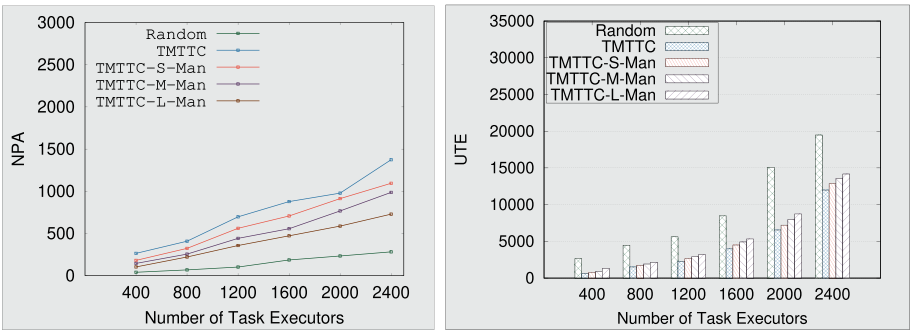
(a) Comparison based on NPA (b) Comparison based on UTE

Fig. 5 Comparison based on NPA and UTE (m=n) for large dataset



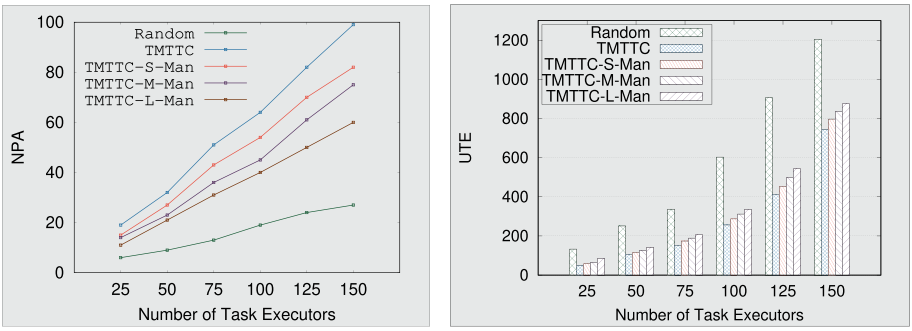
(a) Comparison based on NPA (b) Comparison based on UTE

Fig. 6 Comparison based on NPA and UTE (m < n) for small dataset



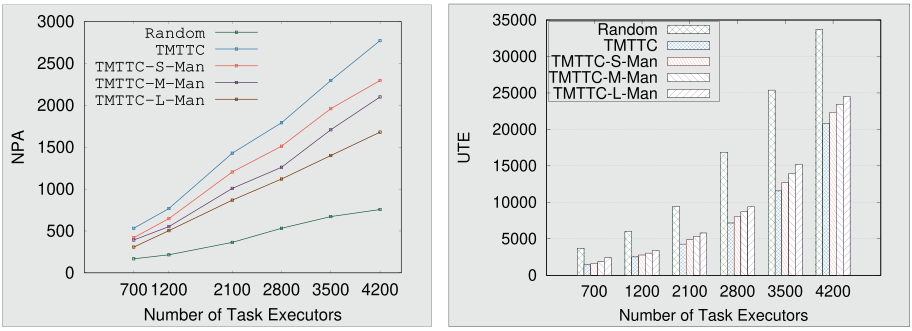
(a) Comparison based on NPA (b) Comparison based on UTE

Fig. 7 Comparison based on NPA and UTE (m > n) for small dataset



(a) Comparison based on NPA (b) Comparison based on UTE

Fig. 8 Comparison based on NPA and UTE (m < n) for small dataset



(a) Comparison based on NPA (b) Comparison based on UTE

Fig. 9 Comparison based on NPA and UTE (m < n) for small dataset

Similar nature of TMTTC and Random can be seen for large dataset (see Table 2) in Figs. 5a, 7a, and 9a. The reason is same as mentioned above.

Considering the manipulative behavior of the participating task executors, it can be seen in Figs. 4a, 6a, and 8a that the NPA value in case of TMTTC with large manipulation (TMTTC-L-Man) is lower than the NPA value in case TMTTC with medium manipulation (TMTTC-M-Man) and is lower than NPA value in case TMTTC with small manipulation (TMTTC-S-Man) and is lower than NPA value in case of TMTTC with no manipulation (TMTTC). So it can be inferred that higher the manipulation lower will be the NPA value in case of TMTTC and it means lower number of task executors are getting their most preferred tasks. As it is natural from the construction of TMTTC and Random.

Similar nature of TMTTC and Random can be seen for large dataset (see Table 2) in Figs. 5a, 7a, and 9a. The reason is same as mentioned above.

Considering the second parameter, in Figs. 4b, 5b, 6b, 7b, 8b, and 9b, x-axis represents the number of task executors and y-axis represents UTE. It can be seen that the UTE value in case of TMTTC is lower than the UTE value in case of Random. As from the construction of TMTTC whenever a particular task executor is considered he is allocated the most preferred task from the available one, whereas in case of Random mechanism some random allocation of task is done to the task executor from his preference list. Due to this in case of TMTTC the value of UTE remain small as compared to UTE value in case of Random. From definition of UTE, it is clear that lower the value of UTE better will be the mechanism. In Figs. 5b, 7b, and 9b the similar nature of TMTTC and Random can be seen for large data set on the ground of UTE value. The reason is same as above.

Considering the manipulative behavior of the participating task executors, it can be seen in Figs. 4b, 6b, and 8b that the UTE value in case of TMTTC with large manipulation (TMTTC-L-Man) is higher than the UTE value in case of TMTTC with medium manipulation (TMTTC-M-Man) and is higher than UTE value in case of TMTTC with small manipulation (TMTTC-S-Man) and is higher than UTE value in case of TMTTC with no manipulation (TMTTC). So it can be inferred that, higher the manipulation higher will be the UTE value and it means higher number of task executors are allocated the tasks that is far away from their most preferred task in their preference ordering. Similar nature of TMTTC and Random can be seen for large dataset (see Table 2) in Figs. 5b, 7b, and 9b. The reason is same as mentioned above.

From above discussion one can infer that in case of TMTTC the participating task executors cannot gain by manipulating their true preference ordering. Dissimilar to the case of TMTTC, in Random mechanism the participating task executors can gain by reporting their preference ordering after manipulation. So, on the ground of *truthfulness* TMTTC beats Random.

8 Conclusion and future works

For the above discussed set-up a truthful mechanism is proposed for distributing the tasks into multiple slots and allocating at most one task to each of the task executors from their reported preference list. Through theoretical analysis it is shown that TMTTC satisfies several properties such as *computational efficiency*, *truthfulness*, *Pareto optimality*, and *The Core*. Further analysis provide the insight about the allocation of most preferred task to each of the task executors from their reported preference ordering. Through simulation it is shown that TMTTC performs better than the benchmark mechanism on the ground of *truthfulness*.

In future, one could consider the set-up where the interests or preference ordering from both the parties (task executors and task providers) is provided and a truthful mechanism could be designed.

Acknowledgements We would like to thanks the faculty members of the School of Computer Science and Engineering (SCOPE), VIT-AP University, Amaravati for their valuable suggestions during the course of this work.

Funding No funds and grants was received.

Code Availability The code will be publicly available upon acceptance.

Declarations

Conflicts of interest There is no conflict of interest.

References

1. Chenxi Qiu, Anna Squicciarini, Dev Rishi Khare, Barbara Carminati, and James Caverlee. Crowdeval: A cost-efficient strategy to evaluate crowdsourced worker's reliability. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 1486-1494, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems
2. Duan Z, Tian L, Yan M, Cai Z, Han Q, Yin G (2017) Practical incentive mechanisms for iot-based mobile crowdsensing systems. IEEE Access 5:20383–20392
3. Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos. TRAC: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1231–1239, Toronto, ON, Canada, April 2014
4. Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys*, 51(1):7:1–7:40, January 2018
5. Gale D, Shapley LS (1962) College admissions and the stability of marriage. American Mathematical Monthly 69:9–15
6. L. Gao, F. Hou, and J. Huang. Providing long-term participation incentive in participatory sensing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2803–2811, Kowloon, Hong Kong, April 2015

7. G. Goel, A. Nikzad, and A. Singla. Mechanism design for crowdsourcing markets with heterogeneous tasks. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, November 2-4, 2014, Pittsburgh, Pennsylvania, USA*, 2014
8. https://en.wikipedia.org/wiki/crowdsourcing, May 2018
9. Jaya Mukhopadhyay, Vikash Kumar Singh, Anita Pal, and Abhishek Kumar. A truthful budget feasible mechanism for iot-based participatory sensing with incremental arrival of budget. *Journal of Ambient Intelligence and Humanized Computing*, Feb 2021
10. Jurairat Phuttharak and Seng Wai Loke (2019) A review of mobile crowdsourcing architectures and challenges: Toward crowd-empowered internet-of-things. IEEE Access 7:304–324
11. B. Klaus, D. F. Manlove, and F. Rossi. Matching under preferences. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, pages 333–355. Cambridge University Press, Cambridge, New York, April 2016
12. Kleinberg Jon, Tardos Eva (2005) Algorithm Design. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA
13. Kong X, Liu X, Jedari B, Li M, Wan L, Xia F (2019) Mobile crowdsourcing in smart cities: Technologies, applications, and future challenges. IEEE Internet of Things Journal 6(5):8095–8113
14. Lefeng Zhang, Ping Xiong, Wei Ren, and Tianqing Zhu. A differentially private method for crowdsourcing data submission. *Concurrency and Computation: Practice and Experience*, 31(19):e5100, 2019. e5100 cpe.5100
15. Li Yang, Zhao Yunlong, Ishak Serrat, Song Hongtao, Wang Nianbin, Yao Nianmin (2018) An anonymous data reporting strategy with ensuring incentives for mobile crowd-sensing. Journal of Ambient Intelligence and Humanized Computing 9(6):2093–2107
16. Li Jiaye, Hao Yu, Zhang Leyuan, Wen Guoqiu (2019) Double weighted k-nearest voting for label aggregation in crowdsourcing learning. Multimedia Tools and Applications 78:33357–33374
17. T. Luo, S. K. Das, H. P. Tan, and L. Xia. Incentive mechanism design for crowdsourcing: An all-pay auction approach. *ACM Transactions on Intelligent Systems and Technology*, 7(3):35:1–35:26, February 2016
18. Masaki Kobayashi, Hiromi Morita, Masaki Matsubara, Nobuyuki Shimizu, and Atsuyuki Morishima. An empirical study on short- and long-term effects of self-correction in crowdsourced microtasks. In *HCOMP*, pages 79–87. AAAI Press, 2018
19. Mazlan Nurulhasanah, Ahmad Sharifah Sakinah Syed, Kamalrudin Massila (2018) Volunteer selection based on crowdsourcing approach. Journal of Ambient Intelligence and Humanized Computing 9(3):743–753
20. Munro Robert (2013) Crowdsourcing and the crisis-affected community. Information Retrieval 16(2):210–266
21. T. Roughgarden. CS269I: Incentives in computer science (Stanford University course), 2016. Lecture 3: Strategic Voting
22. T. Roughgarden. CS269I: Incentives in computer science, (Stanford University Course), Lecture #1: The draw and college admissions, September 2016
23. T. Roughgarden. CS364A: Algorithmic game theory (Stanford University course), lecture #9: Beyond quasi-linearity, October 2013
24. Ruiyun Yu, Jiannong Cao, Rui Liu, Wenyu Gao, Xingwei Wang, and Junbin Liang. Participant incentive mechanism toward quality-oriented sensing: Understanding and application. *ACM Trans. Sen. Netw.*, 15(2):21:1–21:25, February 2019
25. Samarjit Roy, Dhiman Sarkar, and Debashis De. Dewmusic: crowdsourcing-based internet of music things in dew computing paradigm. *Journal of Ambient Intelligence and Humanized Computing*, page 2103-2119, Feb 2021
26. J. Schummer and R. V. Vohra. Mechanism design without money. In E. Tardos N. Nisan, T. Roughgarden and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 209–242. Cambridge University Press, New York, 2007
27. Shahzad Sarwar Bhatti, Xiaofeng Gao, and Guihai Chen. General framework, opportunities and challenges for crowdsourcing techniques: A comprehensive survey. *Journal of Systems and Software*, 167:110611, 2020
28. Shapley L, Scarf H (1974) On cores and indivisibility. Journal of Mathematical Economics 1:23–37
29. Y. Singer. Budget feasible mechanisms. In Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, FOCS '10, pages 765–774, Washington, DC, USA, 2010. IEEE Computer Society
30. V. K. Singh, S. Mukhopadhyay, F. Xhafa, and P. Krause. A quality-assuring, combinatorial auction based mechanism for IoT-based crowdsourcing. In *Advances in Edge Computing: Massive Parallel Processing and Applications*, volume 35, pages 148–177. IOS Press, 2020

31. Slivkins A, Vaughan JW (2014) Online decision making in crowdsourcing markets: Theoretical challenges. SIGecom Exchanges 12(2):4–23
32. Syed Thouheed Ahmed, Vinoth Kumar, and JungYoon Kim. Aitel: ehealth augmented intelligence based telemedicine resource recommendation framework for IoT devices in smart cities. *IEEE Internet of Things Journal*, pages 1–1, 2023
33. Venkatraman S, Surendiran B (2020) Adaptive hybrid intrusion detection system for crowd sourced multimedia internet of things systems. Multimedia Tools and Applications 79:3993–4010
34. Vikash Kumar Singh, Sajal Mukhopadhyay, Fatos Xhafa, and Aniruddh Sharma. A budget feasible peer graded mechanism for iot-based crowdsourcing. *Journal of Ambient Intelligence and Humanized Computing*, 11(4):1531–1551, Jan 2020
35. Wang Xiumin, Tushar Wayes, Yuen Chau, Zhang Xinglin (2020) Promoting users' participation in mobile crowdsourcing: A distributed truthful incentive mechanism (dtim) approach. IEEE Transactions on Vehicular Technology 69(5):5570–5582
36. Wen Yutian, Shi Jinyu, Zhang Qi, Tian Xiaohua, Huang Zhengyong, Hui Yu, Cheng Yu, Shen Xuemin (2015) Quality-driven auction-based incentive mechanism for mobile crowd sensing. IEEE Transactions on Vehicular Technology 64(9):4203–4214
37. Xiaolong Xu, Qing Cai, Guoming Zhang, Jie Zhang, Wei Tian, Xiaorui Zhang, and Alex X. Liu. An incentive mechanism for crowdsourcing markets with social welfare maximization in cloud-edge computing. *Concurrency and Computation: Practice and Experience*, 33(7):e4961, 2021. e4961 cpe.4961
38. Xiaowen Gong and Ness Shroff. Incentivizing truthful data quality for quality-aware mobile data crowdsourcing. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Mobihoc'18, pages 161–170, New York, NY, USA, 2018. ACM
39. P. Xu, A. Srinivasan, K. K. Sarpatwar, and K. Wu. Budgeted online assignment in crowdsourcing markets: Theory and practice. In Proceedings of the 16^{th} Conference on Autonomous Agents and MultiAgent Systems, AAMAS'17, pages 1763–1765, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems
40. Ying Hu, Yingjie Wang, Yingshu Li, and Xiangrong Tong. An incentive mechanism in mobile crowdsourcing based on multi-attribute reverse auctions. *Sensors*, 18(10), 2018