Countermeasures against False-name Attacks on Truthful Incentive Mechanisms for Crowdsourcing

Xiang Zhang, *Student Member, IEEE*, Guoliang Xue, *Fellow, IEEE*, Ruozhou Yu, *Student Member, IEEE*, Dejun Yang, *Member, IEEE*, Jian Tang, *Senior Member, IEEE*

***Abstract*—The proliferation of crowdsourcing brings both opportunities and challenges in various fields, such as environmental monitoring, healthcare, and so on. Often, the collaborative efforts from a large crowd of users are needed in order to complete crowdsourcing jobs. In recent years, the design of crowdsourcing incentive mechanisms has drawn numerous interests from the research community, where auction is one of the commonly adopted mechanisms. However, few of these auctions consider the robustness against false-name attacks (a.k.a. sybil attacks), where dishonest users generate fake identities to increase their utilities without devoting more efforts. To provide countermeasures against such attacks, we design TAFA as an auction-based incentive mechanism for crowdsourcing. We prove that TAFA is truthful, individually rational, budget-balanced, and computationally efficient. We also prove that TAFA provides countermeasures against false-name attacks, such that each user is better off not generating any false name. Extensive performance evaluations are conducted and the results further confirm our theoretical analysis.**

***Index Terms*—Game theory; crowdsourcing; incentive mechanism; false-name proofness; truthfulness.**

1. INTRODUCTION

The popularity of smart devices and the emergence of social networks have made crowdsourcing a new computing and communication paradigm [12, 20]. Applications of crowdsourcing include WiFi map [11], indoor navigation [5], transportation [18], and environmental monitoring [2], *etc*. Amazon Mechanical Turks (AMT) [1] is an online platform, where the Human Intelligent Tasks (HITs) are posted for users to complete. Instead of hiring professional workers, AMT outsources these tasks to the crowd and offers monetary rewards to incentivize users to complete these tasks, which demonstrates the essence of crowdsourcing: utilizing the power of the crowd.

The AMT example also demonstrates that to complete different kinds of HITs, such as pattern recognition, image classification, and survey, collaborative efforts from multiple users are needed. This kind of collaboration does not only exist in AMT, but many other crowdsourcing applications as well. Another instance of such collaboration is Wi-Fi signal sensing [11], where one user may not be able to single-handedly detect all signal strength levels over a large region. However, the combination of several small areas sensed by a group of users can cover a relatively large area. Thus, with proper arrangements, users can collaboratively complete jobs that a single user alone cannot complete.

Since working on crowdsourcing tasks incurs costs to users in terms of time, power, privacy leakage, and so on, most users would not participate voluntarily. However, with appropriate incentives, users would be more inclined to devote their efforts. Thus, an important functionality of the crowdsourcing platform is to provide a good incentive mechanism. Auction theory, which is a branch of game theory, has been widely applied in the design of incentive mechanisms for crowdsourcing. Various design goals and properties of mechanisms have been studied [4, 9, 10, 13, 14, 20, 25]. Among these properties, truthfulness ensures that a user has no incentive to bid dishonestly. An untruthful auction is vulnerable to price manipulation and users may leave the system in fear of unfair treatment.

Despite the large body of research efforts in truthful crowdsourcing auction design, few of them consider another kind of dishonest behavior: false-name attack (a.k.a. sybil attack), where a dishonest user may generate fake identities (false names) and gain an increment in utility without extra efforts. A robust crowdsourcing incentive mechanism should be resistant against such attacks. However, many existing truthful crowdsourcing auctions are vulnerable to false-name attacks. We will discuss some of these works in Section 2 and give an example of such vulnerability in Section 4, respectively.

In this paper, we consider the crowdsourcing model where collaborations from different users are required. We aim to design an incentive mechanism to satisfy individual rationality, budget-balance, computational efficiency, and truthfulness, which are introduced in Section 3-C. We also want the mechanism to be robust against false-name attacks. Driven by the above motivation, we propose **TAFA** (**T**ruthful **A**uction with countermeasures against **F**alse-name **A**ttacks.

The main contributions of this paper are as follows.

- To the best of our knowledge, we are the first to study truthful and false-name proof auctions for crowdsourcing, where collaborative efforts from users are required.
- We consider the crowdsourcing model where users collaboratively complete tasks with intrinsic cost functions, while dishonest users may launch false-name attacks or submit untruthful bids.
- We design **TAFA** and prove that it satisfies the following

Zhang, Xue, and Yu are with Arizona State University, Tempe, AZ 85287. Email: {xzhan229, xue, ruozhouy}@asu.edu. Yang is with Colorado School of Mines, Golden, CO 80401. Email: djyang@mines.edu. Tang is with Syracuse University, Syracuse, NY 13244. Email: jtang02@syr.edu. This research was supported in part by NSF grants 1421685, 1444059, 1457262, 1461886, and 1525920. The information reported here does not reflect the position or the policy of the federal government. This is an extension of the paper [26].

properties: individual rationality, budget-balance, computational efficiency, and truthfulness. We also prove that it is robust against false-name attacks.

The rest of this paper is organized as follows. In Section 2, we briefly review the state-of-art works in truthful crowdsourcing auctions and false-name proof auctions. In Section 3, we present the crowdsourcing and auction model, introduce the false-name attack, and state the desired properties and the design goal. In Section 4, we demonstrate the vulnerability to false-name attacks in an existing truthful auction. In Sections 5 and 6, we propose **TAFA** and present the theoretical analysis, respectively. We conduct extensive performance evaluations in Section 7 and draw the conclusions in Section 8.

2. RELATED WORK

There is a prosperity in truthful crowdsourcing auctions. Singer and Mittal [13] proposed two online truthful and constant-competitive auctions to maximize the number of completed tasks and to minimize the total payment, respectively. An online truthful auction, BP-UCB, was proposed by Singla and Krause [14] using regret minimization to optimize the social welfare of all users. Yang *et al.* [20] studied two crowdsourcing models, where a Stackelberg Equilibrium was computed in the platform-centric model, and a truthful auction was proposed under the user-centric model. Koutsopoulos [7] designed a truthful auction for participatory mobile crowdsourcing to compute participation levels and payments. Zhao *et al.* [27] designed two truthful auctions for crowdsensing with constant competitiveness, where the first one applies for the zero arrival-departure model and the second one applies for the general model. In [6], Feng *et al.* proposed two auctions for location-aware crowdsourcing, where the first one computes a near-optimal task allocation and the second one guarantees truthfulness. Zhang *et al.* [24] proposed a truthful auction for labelling in crowdsourcing with a budget constraint. In [26] by Zhang *et al.*, three crowdsourcing models were proposed with user collaborations, and a truthful auction was proposed for each of the models. None of the above truthful auctions considered the impact of false-name attacks. Li *et al.* studied truthful and group strategy-proof set cover auctions [8], where the auction model is similar to the one in this paper. However, they did not consider false-names attacks either.

The hardness of designing a false-name proof auction for crowdsourcing resides in the fact that a dishonest user may deliberately let some of its false names lose the auction in order to increase the payments of its other false names, so as to achieve a higher utility. Earlier works [15, 16, 21–23] were proposed to study truthful and false-name proof auctions. [16, 21] studied characteristics of false-name proof auctions. The auctions in [15, 23] considered false-name proofness for homogenous items. These auctions cannot be directly applied to the scenario considered in this paper, where users need to complete heterogeneous tasks. [22] provided a false-name proof auction for heterogeneous items. It requires a pre-defined "leveled decision set" (LDS) from the auctioneer as input. In an LDS, there is only one copy for each item, whereas in our scenario, multiple copies of a task need to be allocated. Wang *et al.* proposed ALETHEIA [17], which, with a conflict graph as an input, is truthful and false-name proof for the secondary spectrum market. With no such input in crowdsourcing, ALETHEIA cannot be applied to the system model introduced in Section 3. To our best knowledge, we are the first to design a truthful and false-name proof auction for crowdsourcing.

3. SYSTEM MODEL

In this section, we first present the crowdsourcing model and the corresponding auction model. We introduce the formal definition of a false-name attack. We state the desired properties and the design goal of this paper at the end of this section.

A. Crowdsourcing and Auction Models

The crowdsourcing system consists of a *platform* and a set of *users* $\mathcal{U} = \{1, 2, ..., n\}$. There is a universal set of *tasks* $\Gamma = \{\tau_1, \tau_2, ..., \tau_h\}$, where the h tasks are distinct and each task is indivisible. The platform has a set of *jobs* $\mathcal{J} = \{J_1, J_2, \ldots, J_m\}$ for users to complete, where each job is a multi-subset of Γ. A job is *completed iff* each task in it is completed by the users. For example, there is a job $J_6 = \{\tau_1, \tau_1, \tau_1, \tau_4, \tau_5, \tau_5\}$. J_6 is completed *iff* three copies of τ_1, one copy of τ_4, and two copies of τ_5 are completed.

Each $J_i \in \mathcal{J}$ is associated with a *value* $\nu_i > 0$, which is known only by the platform and private to the users. If J_i is completed, the platform gains ν_i. We define the value of a set of jobs $\mathcal{J}$ as the sum of each job's value, i.e., $v(\mathcal{J}) = \sum_{J_i \in \mathcal{J}} \nu_i$. We use $\mathbf{v}$ to denote the vector $(\nu_1, \nu_2, ..., \nu_m)$.

For each user j, to complete a multi-subset of tasks $\mathcal{S}_j \subseteq \Gamma$, it incurs a *cost* $c_j(\mathcal{S}_j) \geq 0$. The *cost function* $c_j(\cdot)$ is monotonically increasing and known only by user j. We set $c_j(\mathcal{S}_j) = +\infty$ if user j is not able to complete all tasks in $\mathcal{S}_j$.

The auction model works as follows. The platform serves as the *auctioneer* and announces the set $\mathcal{J}$ of jobs. Each user j submits $m_j > 0$ *bids*: $(\mathcal{T}_j^1, a_j^1), (\mathcal{T}_j^2, a_j^2), ..., (\mathcal{T}_j^{m_j}, a_j^{m_j})$. For each $(\mathcal{T}_j^k, a_j^k)$, $\mathcal{T}_j^k$ is the multi-subset of tasks that user j can complete, and a_j^k is the corresponding *ask*, which is the minimum amount of reward that user j requires to complete all tasks in $\mathcal{T}_j^k$. Note that a_j^k and $c_j(\mathcal{T}_j^k)$ are not necessarily the same. Since there may be an exponential number of bids from a user, we impose an upper bound on the number of bids that a user can submit. We use M to denote this upper bound, i.e., $m_j \leq M$ for each user j. We define $A_j = ((\mathcal{T}_j^1, a_j^1), ..., (\mathcal{T}_j^{m_j}, a_j^{m_j}))$ and $\mathbf{A} = (A_1, A_2, ..., A_n)$.

Upon receiving the bids, the platform selects a subset of users $\mathcal{U}_w \subseteq \mathcal{U}$ as winning users, and assigns a task set $\mathcal{S}_j \in \{\mathcal{T}_j^1, \mathcal{T}_j^2, ..., \mathcal{T}_j^{m_j}\}$ to user $j \in \mathcal{U}_w$. The platform also computes a *payment* p_j for user j. We define *assignment vector* $\mathbb{S} = (\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n)$ and *payment vector* $\mathbf{p} = (p_1, p_2, ..., p_n)$. The *utility* of user j is defined as the payment received minus the incurred cost, i.e., $u_j = p_j - c_j(\mathcal{S}_j)$. The *platform utility* is defined as the value of the completed jobs minus the payment to the users, i.e., $u_0 = v(\mathcal{J}) - \sum_{j \in \mathcal{U}} p_j$.

B. False-name Attacks

In a false-name attack, a user j generates $\kappa_j \geq 2$ false names, denoted as $j_1, j_2, ..., j_{\kappa_j}$. Each false name j_k submits m_{j_k} bids: $(\mathcal{T}^1_{j_k}, a^1_{j_k}), (\mathcal{T}^2_{j_k}, a^2_{j_k}), ..., (\mathcal{T}^{m_{j_k}}_{j_k}, a^{m_{j_k}}_{j_k})$. After the auction, each j_k is assigned to complete $\mathcal{S}_{j_k}$ and its payment is p_{j_k}. The utility of user j from this false-name attack is defined as the payment collected by all false names minus the cost of the union of all the assigned tasks, i.e., $u_j = \sum_{k=1}^{\kappa_j} p_{j_k} - c_j(\biguplus_{j_k \in \mathcal{U}} \mathcal{S}_{j_k})$.

C. Desired Properties and Design Goal

The crowdsourcing platform plays an important rule to apply appropriate mechanism to allocate tasks and compute payments. A keen concern is that users may gain higher utilities from dishonest behaviors. This makes the mechanism vulnerable to malicious price manipulation and untrustworthy to other users. There are several desirable properties that a good auction mechanism should possess. We list them in the following.

- **Individual Rationality**: An auction is individually rational if for each individual, its utility is non-negative when revealing its true valuation or cost.
- **Budget Balance**: An auction is budget-balanced if the auctioneer's utility is non-negative.
- **Computational Efficiency**: An auction is computationally efficient if the auction mechanism can be executed within a polynomial time complexity.
- **Truthfulness**: An auction is truthful if for each individual, it cannot increase its utility by unilaterally deviating from revealing its true valuation or cost.
- **False-name Proofness**: An auction is false-name proof if for each individual, it cannot increase its utility by launching a false-name attack.

The design goal of this paper is to design an incentive mechanism under the crowdsourcing model proposed in Section 3-A. The incentive mechanism needs to guarantee the following properties: individual rationality, budget-balance, computational efficiency, truthfulness, and false-name proofness.

4. False-name Attacks on Incentive Mechanisms

In this section, we discuss the vulnerability to false-name attacks in truthful incentive mechanisms. We present a detailed example of how a false-name attack increases a dishonest user's utility in an existing truthful auction [26].

As discussed in Section 2, there are many truthful auction-based incentive mechanisms for crowdsourcing [6, 7, 20, 24, 26]. However, these auctions do not regard the false-name attack as a potential threat to the robustness of the mechanisms. As a result, many auctions suffer from false-name attacks.

We use the incentive mechanisms in [26] as an illustration of such vulnerability. In [26], three crowdsourcing models, SS, SM, and MM, were proposed. An auction-based incentive mechanism was designed for each model, named IMC-SS, IMC-SM, and IMC-MM, respectively. It has been proved that these auctions are truthful, individually rational, budget-balanced, and computationally efficient. We choose IMC-SS as the mechanism for illustration, whose system model is a special case of our model in Section 3 where a user can make at most one bid. The reason why we choose IMC-SS instead of IMC-SM (whose system model is similar to the one in Section 3) is that IMC-SS is easier to follow.

In IMC-SS, there are three major steps: job selection, winning provider (user) selection, and pricing. In job selection, jobs are selected with no *monopoly providers*. Monopoly providers are the ones without whom the selected jobs cannot be completed. To guarantee truthfulness, monopoly providers need to be excluded. In the winning provider selection step, providers are selected iteratively according to the minimum ratio of ask to the number of tasks that can be used to complete the selected jobs. The provider is assigned to work on these tasks. In the pricing step, for each winning provider j, a process similar to the winning provider selection is run with j excluded. During this new selection, each provider is also selected with the minimum ratio of ask to the number of tasks that can be completed by the provider. We use this ratio to multiply with the number of tasks that j can complete at this moment, and record the value. The maximum of these values is j's payment.

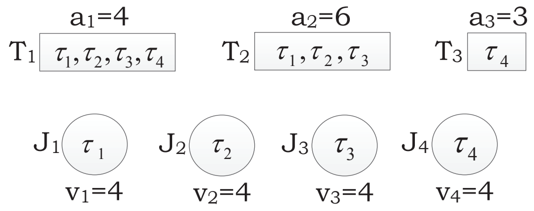


Fig. 1. An illustration of IMC-SS

We illustrate the vulnerability of false-name attacks on IMC-SS using the example in Fig. 1. There are four jobs, each consisting of one task and tagged with a private value of 4. There are three users: 1, 2, and 3. Their submitted bids are $(\{\tau_1, \tau_2, \tau_3, \tau_4\}, 4)$, $(\{\tau_1, \tau_2, \tau_3\}, 6)$, and $(\{\tau_4\}, 3)$, respectively. For user 1, it's true cost to work on $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ is 4.

IMC-SS works as follows.

Job Selection: It is easy to verify that all four jobs can be completed without any monopoly user, since without any user, all four jobs can still be completed by the other two.

Winning Provider Selection: User 1 is selected first with the minimum ratio ($\frac{a_1}{|T_1 \cap \{\tau_1, \tau_2, \tau_3, \tau_4\}|} = 1$). Since user 1 can complete all the tasks, there is only one winning user.

Pricing: This is similar to Winning Provider Selection with user 1 excluded. At first $p_1 = 0$ and the available task set is $\mathcal{T}' = \{\tau_1, \tau_2, \tau_3, \tau_4\}$. User 2 is selected with the minimum ratio: $\frac{a_2}{|T_2 \cap \mathcal{T}'|} = \frac{a_2}{|\{\tau_1, \tau_2, \tau_3\}|} = 2$. Then p_1 is set to $\max\{0, 2 \times |\mathcal{T}' \cap T_1|\} = 8$. The available task set is updated to $\mathcal{T}' = \{\tau_4\}$. Next, user 3 is selected, and p_1 is set to $\max\{8, \frac{3}{1} \times |\mathcal{T}' \cap T_1| = 3\} = 8$. Thus, $p_1 = 8$ and *user 1's utility is* 4 $(= 8 - 4)$.

Now suppose that user 1 launches a false-name attack by generating two false names: 1_1 with bid $(\{\tau_1, \tau_2, \tau_3, \tau_4\}, 4)$ and 1_2 with bid $(\{\tau_1\}, 1.5)$, as shown in Fig. 2.

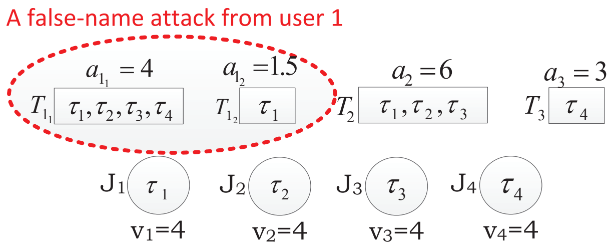


Fig. 2. A false-name attack from user 1

After the job selection and winning provider selection, user 1_1 is the only winner with assigned tasks $\{\tau_1, \tau_2, \tau_3, \tau_4\}$. To compute p_{1_1}, we exclude user 1_1 and set $\mathcal{T}' = \{\tau_1, \tau_2, \tau_3, \tau_4\}$. User 1_2 is selected first and $p_{1_1} = \max\{0, \frac{1.5}{1} \times |\mathcal{T}' \cap T_1|\} = 6$. $\mathcal{T}'$ is updated to $\{\tau_2, \tau_3, \tau_4\}$. Next, user 2 is selected and $p_{1_1} = \max\{6, \frac{6}{|\mathcal{T}' \cap T_2|} \times |\mathcal{T}' \cap T_1| = \frac{6}{2} \times 3\} = 9$. $\mathcal{T}'$ is updated to $\{\tau_4\}$. In the end, user 3 is selected and $p_{1_1} = \max\{9, \frac{3}{1} \times |\mathcal{T}' \cap T_1|\} = 9$. Thus, *user* 1 *gains a higher utility of* 5 $(= 9 - 4)$ *by launching the above false-name attack*. This shows that IMC-SS is vulnerable to false-name attacks. It can be shown that IMC-SM is also vulnerable to false-name attacks. We omit details here.

Note that false-name attacks may impact the auction results in two ways. First, a dishonest user may increase its utility, at the cost of other entities in the system (the job owner or other users). This behaviour may result in fear of other users over market manipulation. Second, false-name attacks will increase the computational overhead for the auction system, because the number of users (true users and false names) will increase. Therefore, it is of our interest to design an incentive mechanism that is robust against false-name attacks.

Remark 4.1: This paper is an extension of the conference paper [26]. Therefore, there are similarities between the two, especially in system models. A key difference between the two is that this paper studies false-name attacks, demonstrates the vulnerabilities of the incentive mechanisms in [26] to false-name attacks, and presents an incentive mechanism (**TAFA**) that is both truthful and robust against false-name attacks. $\square$

5. Detailed Design of TAFA

The rationale behind the design of **TAFA** is that we break the bid from each user into *atomic bids* where each atomic bid bids only one task. Thus, as long as the auction is robust against false-name attacks for atomic bids, the false names have no impact on the auction results. In **TAFA**, we use the ratio of ask value over task size as the atomic ask value. To provide robustness against false-name attacks from atomic bids, we use principles from the $(d_k + 1)$-th price auction, where d_k is the number of tasks in τ_k that need to be completed. In the (d_k+1)-th price auction, the d_k smallest bids are winning bids and the payment is the $(d_k + 1)$-th smallest value. We use these payments to compute the final payment for each user.

Different from the algorithms in [26], we assume that there is no *monopoly user* in our crowdsourcing scenario. This is justifiable because in reality, there is often a large number of users in crowdsourcing applications [1, 2, 11, 18, 19] and many related works eliminate the assumption of monopoly users [3, 8] due to such abundance.

There are two major steps in **TAFA**: winning user selection and pricing. Algorithm 1 presents the winning user selection and Algorithm 2 presents the pricing. These two algorithms are used in Algorithm 3, which is the main algorithm of **TAFA**.

Algorithm 1: $WinUserSel(\mathcal{U}, \mathcal{J}, \mathbf{A})$

```
   Output: $\mathcal{U}_w, \delta(\cdot), \mathbb{S}$
1  $\mathcal{U}_w \leftarrow \emptyset$, $\mathcal{S}_j \leftarrow \emptyset$, and $\delta(j) \leftarrow 0$ for each $j \in \mathcal{U}$;
2  $\mathcal{T}^* \leftarrow \biguplus_{J_i \in \mathcal{J}} J_i$;
3  while $\mathcal{T}^* \neq \emptyset$ do
4      $\mathcal{T}_{j^*}^{k^*} \leftarrow \arg\min_{\mathcal{T}_j^k}\{\frac{a_j^k}{|\mathcal{T}_j^k|} | j \in \mathcal{U} \setminus \mathcal{U}_w, \mathcal{T}_j^k \cap \mathcal{T}^* \neq \emptyset\}$;
5      $\delta(j^*) \leftarrow k^*$, $\mathcal{S}_{j^*} \leftarrow \mathcal{T}_{j^*}^{k^*}$;
6      $\mathcal{T}^* \leftarrow \mathcal{T}^* \setminus \mathcal{T}_{j^*}^{k^*}$, $\mathcal{U}_w \leftarrow \mathcal{U}_w \cup \{j^*\}$;
7  end
8  return $\mathcal{U}_w, \delta(\cdot), \mathbb{S}$
```

Algorithm 2: $Pricing(j, \mathcal{U}_w, \mathcal{U}, \mathcal{J}^*, \mathbf{A}, \mathcal{S}_j)$

```
    Output: $p_j$
1   if $j \notin \mathcal{U}_w$ then
2       $p_j \leftarrow 0$;
3   else
4       $p_j' \leftarrow 0$, $\mathcal{U}_w' \leftarrow \emptyset$, $\mathcal{T}' \leftarrow \biguplus_{J_i \in \mathcal{J}} J_i$;
5       $\mathcal{S}_l' \leftarrow \emptyset$ and $\delta'(l) \leftarrow 0$ for each $l \in \mathcal{U}$;
6       while $\mathcal{T}' \neq \emptyset$ do
7           $\mathcal{T}_{j'}^{k'} \leftarrow \arg\min_{\mathcal{T}_l^k}\{\frac{a_l^k}{|\mathcal{T}_l^k|} | l \in \mathcal{U} \setminus \mathcal{U}_w', \mathcal{T}_l^k \cap \mathcal{T}^* \neq \emptyset, l \neq j\}$;
8           $\mathcal{S}_{j'}' \leftarrow \mathcal{T}_{j'}^{k'}$, $\delta'(j') \leftarrow k'$;
9           $\mathcal{U}_w' \leftarrow \mathcal{U}_w' \cup \{j'\}$, $\mathcal{T}' \leftarrow \mathcal{T}' \setminus \mathcal{T}_{j'}^{k'}$;
10      end
11      for each $\tau_k \in \mathcal{S}_j$ do
12          for each user $l$ such that $\tau_k \in \mathcal{S}_l'$ do
13              $p_j' \leftarrow \max\{p_j', \frac{a_l^{\delta'(l)}}{|\mathcal{S}_l'|}\}$;
14          end
15      end
16      $p_j \leftarrow p_j' |\mathcal{S}_j|$;
17  end
18  return $p_j$;
```

Winning User Selection: Algorithm 1 takes the user set $\mathcal{U}$, job set $\mathcal{J}$, and bids $\mathbf{A}$ as input, and returns the winning user set $\mathcal{U}_w$, assignment function $\delta(\cdot)$, and assignment vector $\mathbb{S}$. $\delta(j) = k$ indicates that user j is assigned to complete $\mathcal{T}_j^k$. If j is not assigned to any task, $\delta(j) = 0$. The algorithm proceeds iteratively. In each iteration (Lines 4 to 6), we select a user with the minimum ask-per-task value.

In Line 1, we initialize the winning user set $\mathcal{U}_w$ to $\emptyset$. For each user j, we initialize $\mathcal{S}_j$ to $\emptyset$ and $\delta(j)$ to 0. In Line 2, all tasks from different jobs are combined into a multi-set of tasks

$\mathcal{T}^*$, which is the set of tasks yet to be assigned. To proceed with each iteration, we select a submitted task set from users not in $\mathcal{U}_w$ with the minimum ratio of ask value to the task set size (Line 4). We break ties with an arbitrary order, e.g., a non-decreasing user index order, if there are ties during this selection. Once we select the task set $T_{j^*}^{k^*}$, $\delta(j^*)$ is updated to k^*, and $\mathcal{T}_{j^*}^{k^*}$ is assigned to user j^* as $\mathcal{S}_{j^*}$ (Line 5). We update the remaining task set $\mathcal{T}^*$ and add the selected user into $\mathcal{U}_w$ in Line 6. The iteration stops when $\mathcal{T}^*$ becomes $\emptyset$.

Pricing: Algorithm 2 computes the payment for each user j. A process similar to $WinUserSel$ is run over the users with user j excluded temporarily. For each task assigned to j in the previous step, we compute a value which is the maximum per-task ask of the allocated tasks in the pricing step. The maximum of these values over all tasks assigned to j is the per-task payment for user j. Then the payment p_j is the per-task payment times the number of tasks assigned to user j.

In Line 2, user j's payment is 0 if it is not a winning user. Otherwise, we initialize the per-task payment p'_j to be 0 in Line 4. Lines 5 to 10 are similar to Algorithm 1, except that user j cannot be selected into the winning user set $\mathcal{U}'_w$ according to Line 7. To compute p_j, we first compute the per-task payment p'_j. In Line 11, we examine each task $\tau_k \in \mathcal{S}_j$ that is assigned to user j by Algorithm 1. If this task is assigned to another user l who is selected into $\mathcal{U}'_w$, p'_j is set to the higher one of p'_j and user l's per-task ask value (Lines 12 and 13). In Line 16, the payment p_j is computed as the per-task payment p'_j times the number of tasks assigned to j.

The main algorithm of **TAFA** is illustrated in Algorithm 3. We run $WinUserSel$ to select the winning users and assign tasks. Then we compute the payment for each user. We compare the value of the completed jobs v and the total payment p to guarantee budget-balance. If $v \geq p$, the auction is successful; it is cancelled otherwise.

Algorithm 3: TAFA$(\mathcal{U}, \mathcal{J}, \mathbf{v}, \mathbf{A})$

```
1 $(\mathcal{U}_w, \delta(\cdot), \mathbb{S}) \leftarrow WinnerSel(\mathcal{U}, \mathcal{J}, \mathbf{A})$;
2 for each $j \in \mathcal{U}$ do
3   | $p_j \leftarrow Pricing(j, \mathcal{U}_w, \mathcal{U}, \mathcal{J}^*, \mathbf{A}, \mathcal{S}_j)$;
4 end
5 $v \leftarrow \sum_{J_i \in \mathcal{J}} \nu_i$, $p \leftarrow \sum_{j \in \mathcal{U}} p_j$;
6 if $v < p$ then
7   | $\mathcal{S}_j \leftarrow \emptyset$, for each $j \in \mathcal{U}_w$, $\mathcal{U}_w \leftarrow \emptyset$, $\mathbf{p} \leftarrow \mathbf{0}$;
8 end
9 return $\mathcal{U}_w, \mathcal{S}, \mathbf{p}$.
```

A Walk-through Example:

To provide a better understanding of how **TAFA** works, we present a walk-though example in Fig. 3.

There are two jobs $\mathcal{J} = \{J_1, J_2\}$ with $v_1 = 14$ and $v_2 = 18$, respectively. There are four users. The submitted tasks and the corresponding ask values are shown in the figure.

Winning User Selection: Initially, $\mathcal{U}_w = \emptyset$ and $\mathcal{T}^* = J_1 \biguplus J_2 = \{\tau_1, \tau_2, \tau_3, \tau_4\}$. $\mathcal{T}_1^1$ is selected as $\mathcal{T}_{j^*}^{k^*}$ in the first iteration with $\frac{a_1^1}{|\mathcal{T}_1^1 \cap \mathcal{T}^*|} = \frac{4}{|\{\tau_1, \tau_3\}|} = 2$. We update $\delta(1) = 1$, $\mathcal{S}_1 = \{\tau_1, \tau_3\}$, $\mathcal{T}^* = \{\tau_2, \tau_4\}$, and $\mathcal{U}_w = \{1\}$. Next, $\mathcal{T}_2^1$ is selected. We update $\delta(2) = 1$, $\mathcal{S}_2 = \{\tau_2, \tau_3, \tau_4\}$, $\mathcal{T}^* = \emptyset$, and $\mathcal{U}_w = \{1, 2\}$.

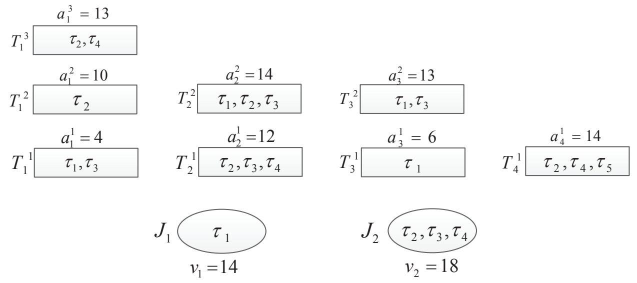


Fig. 3. A walk-though example of **TAFA**

Pricing:

- p_1: Initially, $p'_1 = 0, \mathcal{U}'_w = \emptyset$, and $\mathcal{T}' = \{\tau_1, \tau_2, \tau_3, \tau_4\}$. In the first iteration, $\mathcal{T}_2^1$ is selected as $\mathcal{T}_{j'}^{k'}$ with $\frac{a_2^1}{|\mathcal{T}_2^1 \cap \mathcal{T}'|} = \frac{12}{3} = 4$. We update $\mathcal{S}'_2 = \{\tau_2, \tau_3, \tau_4\}$, $\delta'(2) = 1$, $\mathcal{U}'_w = \{2\}$, and $\mathcal{T}' = \{\tau_1\}$. In the next iteration, $\mathcal{T}_3^1$ is selected as $\mathcal{T}_{j'}^{k'}$. We update $\mathcal{S}'_3 = \{\tau_1\}$, $\delta'(3) = 1$, $\mathcal{U}'_w = \{2, 3\}$, and $\mathcal{T}' = \emptyset$. For $\tau_1 \in \mathcal{S}_1$, we have $\tau_1 \in \mathcal{S}'_3$ and $p'_1 = \max\{p'_1, \frac{a_3^{\delta'(3)}}{|\mathcal{S}'_3|}\} = 6$. For $\tau_3 \in \mathcal{S}_1$, we have $\tau_3 \in \mathcal{S}'_2$, and $p'_1 = \max\{p'_1, \frac{a_2^{\delta'(2)}}{|\mathcal{S}'_2|} = 4\} = 6$. Thus, $p_1 = p'_1|\mathcal{S}_1| = 12$.
- p_2: With the same method, we have $p_2 = 14$.

Since $v = v_1 + v_2 = 32$, $p = p_1 + p_2 = 26$, and $v > p$, the auction is successful, with $\mathcal{U}_w = \{1, 2\}$, $\mathcal{S}_1 = \{\tau_1, \tau_3\}$, $\mathcal{S}_2 = \{\tau_2, \tau_3, \tau_4\}$, $p_1 = 12$, and $p_2 = 14$.

6. Economic Properties of TAFA

In this section, we perform theoretical analysis on **TAFA**.

Theorem 1: **TAFA** is individually rational, budget-balanced, computationally efficient, and truthful. No user can increase its utility by unilaterally launching a false-name attack. □

We prove Theorem 1 with the following lemmas.

Lemma 6.1: **TAFA** is individually rational. □

Proof: Let j be any user that bids truthfully, i.e., $a_j^k = c_j(\mathcal{T}_j^k)$ for $k = 1, \ldots, m_j$. We need to prove that $u_j \geq 0$.

If $j \notin \mathcal{U}_w$, $u_j = 0$. If $j \in \mathcal{U}_w$, when j is selected to $\mathcal{U}_w$, $\frac{a_j^{\delta(j)}}{|\mathcal{T}_j^{\delta(j)}|} = \frac{c_j(\mathcal{T}_j^{\delta(j)})}{|\mathcal{T}_j^{\delta(j)}|}$ is no larger than $\frac{a_{j^*}^{k^*}}{|\mathcal{T}_{j^*}^{k^*}|}$ for any j^* selected after j. In $Pricing$, $p'_j \geq \max\{\frac{a_{j^*}^{k^*}}{\mathcal{T}_{j^*}^{k^*}} | j^* \neq j, j^* \in \mathcal{U}'_w\} \geq \frac{c_j(\mathcal{T}_j^{\delta(j)})}{|\mathcal{T}_j^{\delta(j)}|}$. Thus, $p_j = p'_j|\mathcal{S}_j| \geq c_j^{\delta(j)}$ and $u_j \geq 0$. ■

Lemma 6.2: **TAFA** is budget-balanced. □

Proof: If $\mathcal{U}_w = \emptyset$, $u_0 = 0$. If $\mathcal{U}_w \neq \emptyset$, $u_0 = \sum_{\mathcal{J}_i \in J} v_i - \sum_{j \in \mathcal{U}} p_j \geq 0$ according to Line 6 in Algorithm 3. ■

Lemma 6.3: **TAFA** is computationally efficient. □

Proof: The time complexity of **TAFA** is $O(n^2 Mt)$, where $t = \max\{|\mathcal{T}^*|, m\}$. Thus, it is polynomial-time executable. ■

Lemma 6.4: **TAFA** is truthful. □

Proof: We show that for any user j, it cannot have a higher utility by unilaterally changing its bid from $\bar{A}_j = ((\mathcal{T}_j^1, c_j(\mathcal{T}_j^1)), \ldots, (\mathcal{T}_j^{m_j}, c_j(\mathcal{T}_j^{m_j})))$ to $\hat{A}_j = ((\mathcal{T}_j^1, a_j^1), \ldots, (\mathcal{T}_j^{m_j}, a_j^{m_j}))$ with $(a_j^1, \ldots, a_j^{m_j}) \neq (c_j(\mathcal{T}_j^1), \ldots, c_j(\mathcal{T}_j^{m_j}))$. We use case analysis.

Case 1: *User j loses the auction by bidding $\bar{A}_j$*, which implies that at any iteration during the winner selection stage we have $\frac{a_{j^*}^{k^*}}{|\mathcal{T}_{j^*}^{k^*}|} \leq \min_k\{\frac{c_j(\mathcal{T}_j^k)}{|\mathcal{T}_j^k|} | T_j^k \cap \mathcal{T}^* \neq \emptyset\}$. If by changing the bid, user j remains a loser, its utility does not increase. If user j unilaterally changes its bid to $\hat{A}_j$, and wins the auction to work on $\mathcal{T}_j^{\delta(j)}$ with payment $\hat{p}_j$, then we have $\hat{p}'_j = \max\{\frac{a_{j'}^{k'}}{|\mathcal{T}_{j'}^{k'}|}\} = \max\{\frac{a_{j^*}^{k^*}}{|\mathcal{T}_{j^*}^{k^*}|}\} \leq \frac{c_j(\mathcal{T}_j^{\delta(j)})}{|\mathcal{T}_j^{\delta(j)}|}$. Thus its resulting utility is $u_j = \hat{p}_j - c_j(\mathcal{T}_j^{\delta(j)}) = \hat{p}'_j|\mathcal{T}_j^{\delta(j)}| - c_j(\mathcal{T}_j^{\delta(j)}) \leq 0$.

Case 2: *User j wins the auction by bidding $\bar{A}_j$*, earning a payment of p_j to work on $\mathcal{T}_j^{\delta(j)}$. Hence at the iteration where user j is selected to enter $\mathcal{U}_w$, $\frac{c_j^{\delta(j)}}{|\mathcal{T}_j^{\delta(j)}|}$ is the minimum among all values $\frac{a_{j^*}^{k^*}}{|\mathcal{T}_{j^*}^{k^*}|}$, such that $\mathcal{T}_{j^*}^{k^*} \cap \mathcal{T}^* \neq \emptyset$. By Lemma 6.1, $u_j = p_j - c_j(\mathcal{T}_j^{\delta(j)}) \geq 0$.

Case 2.1: *By changing its bid, j becomes a loser, or j still works on $T_j^{\delta(j)}$*. If j becomes a loser, j's utility becomes 0. If j still works on $T_j^{\delta(j)}$, by Algorithm 2, the payment p_j does not change, hence user j's utility does not change as well.

Case 2.2: *By changing its bid, user j wins to work on $\mathcal{T}_j^{k'}$ with $k' \neq \delta(j)$*, earning a payment $\hat{p}_j$. Since in the pricing stage, the payment for user j does not depend on the bid of j (user j is excluded from $\mathcal{U}_w$ in Algorithm 2), we have $\hat{p}'_j = p'_j$. The second statement in Case 2 implies that $\frac{c_j(\mathcal{T}_j^{\delta(j)})}{|\mathcal{T}_j^{\delta(j)}|} \leq \frac{c_j(\mathcal{T}_j^{k'})}{|\mathcal{T}_j^{k'}|}$. Therefore $(\hat{p}'_j - \frac{c_j(\mathcal{T}_j^{k'})}{|\mathcal{T}_j^{k'}|})|\mathcal{T}_j^{\delta(j)}| \leq (p'_j - \frac{c_j(\mathcal{T}_j^{\delta(j)})}{|\mathcal{T}_j^{\delta(j)}|})|\mathcal{T}_j^{\delta(j)}| = p_j - c_j(\mathcal{T}_j^{\delta(j)})$, i.e., the utility of user j cannot be increased by changing its bid from $\bar{A}_j$ to $\hat{A}_j$ unilaterally. This proves the lemma. ■

Lemma 6.5: TAFA provides countermeasures against false-name attacks, i.e., no user can increase its utility by unilaterally launching a false-name attack. □

Proof: We prove this lemma by proving that a user cannot increase its utility by generating two false names. The case with more false names can be inferred by induction.

Suppose for user j, it generates two false names j_1 and j_2, each with bid $A_{j_1} = \{(\mathcal{T}_{j_1}^1, a_{j_1}^1), \ldots, (\mathcal{T}_{j_1}^{m_{j_1}}, a_{j_1}^{m_{j_1}})\}$ and $A_{j_2} = \{(\mathcal{T}_{j_2}^1, a_{j_2}^1), \ldots, (\mathcal{T}_{j_2}^{m_{j_2}}, a_{j_2}^{m_{j_2}})\}$, respectively. After the auction, j_1 is assigned to work on $\mathcal{S}_{j_1}$ with payment p_{j_1} and j_2 is assigned to work on $\mathcal{S}_{j_2}$ with payment p_{j_2}. We prove that if j does not generate these two false names, the utility when bidding its truthful bid $(\mathcal{S}_{j_1} \uplus \mathcal{S}_{j_2}, c_j(\mathcal{S}_{j_1} \uplus \mathcal{S}_{j_2}))$, is no less than the utility from the false-name attack, i.e., $p_{j_1} + p_{j_2} - c_j(\mathcal{S}_{j_1} \uplus \mathcal{S}_{j_2})$. We use case analysis.

Case 1: *None of j_1 and j_2 is a winning user.* The utility from this false-name attack is 0. By Lemma 6.1, j's utility is non-negative with truthful bids. Thus, j's utility is not increased.

Case 2: *One of j_1 and j_2 is a winning user.* Without loss of generality, assume that j_1 is the winner and j_2 is the loser. Let $\mathcal{U}_{-j} = \{1, 2, ..., j-1, j+1, ..., n\}$, i.e., all users except j. Let $\bar{\mathcal{U}}'_w$ be the winning user set in Algorithm 2 for user j with j's truthful bid and $\hat{\mathcal{U}}'_w$ be the winning user set in Algorithm 2 for j_1 with the false-name attack, respectively. For each task $\tau_k \in \mathcal{S}_{j_1}$, the corresponding per-task payment p'_{j_1} is computed as the d_k-th smallest payment among the winning users in Algorithm 2, where d_k is the number of tasks τ_k in the multi-task $\mathcal{T}^*$. When selecting $\bar{\mathcal{U}}'_w$ and $\hat{\mathcal{U}}'_w$, $\bar{\mathcal{U}}'_w$ is selected from $\mathcal{U}_{-j}$ and $\hat{\mathcal{U}}'_w$ is selected from $\mathcal{U}_{-j} \cup \{j_2\}$ according to line 7 in Algorithm 2. Each time a user is to be selected into $\bar{\mathcal{U}}'_w$ and $\hat{\mathcal{U}}'_w$, we select the one with the minimum per-task ask value. Since $\bar{\mathcal{U}}'_w$ is selected from $\mathcal{U}_{-j}$ and $\hat{\mathcal{U}}'_w$ is selected from $\mathcal{U}_{-j} \cup \{j_2\}$, we know that the selected minimum per-task ask value from $\mathcal{U}_{-j} \cup \{j_2\}$ is no more than the one selected from $\mathcal{U}_{-j}$. Considering that p'_{j_1} is the d_k-th smallest per-task ask value in $\hat{\mathcal{U}}'_w$ and p'_j is the d_k-th smallest per-task ask value in $\bar{\mathcal{U}}'_w$, we have $p'_{j_1} \leq p'_j$. Therefore, we have $p'_{j_1}|\mathcal{S}_{j_1}| - c_j(\mathcal{S}_{j_1}) \leq p'_j|\mathcal{S}_j| - c_j(\mathcal{S}_j)$ with $\mathcal{S}_j = \mathcal{S}_{j_1} \uplus \mathcal{S}_{j_2}$ when j_2 loses the auction.

Case 3: *Both j_1 and j_2 are winning users.* With a similar analysis approach to Case 2, we have $p'_{j_1}|\mathcal{S}_{j_1}| \leq p'_j|\mathcal{S}_j|$ and $p'_{j_2}|\mathcal{S}_{j_2}| \leq p'_j|\mathcal{S}_j|$. Thus, $p'_{j_1}|\mathcal{S}_{j_1}| + p'_{j_2}|\mathcal{S}_{j_2}| - c_j(\mathcal{S}_{j_1} \uplus \mathcal{S}_{j_2}) \leq p'_j(|\mathcal{S}_{j_1}| + |\mathcal{S}_{j_2}|) - c_j(\mathcal{S}_{j_1} \uplus \mathcal{S}_{j_2}) = p_j - c_j(\mathcal{S}_j)$.

Cases 1-3 complete the proof for this lemma. ■

7. Performance Evaluation

We implemented our incentive mechanism **TAFA** for the crowdsourcing model, and ran extensive tests on a Linux PC with Intel Core I7-4770 3.5Hz processor and 16GB memory.

A. Performance Metrics and Simulation Setup

Performance Metrics: We study *platform utility*, *average user utility*, and *running time*. We compare **TAFA** with **IMC-SM** [26] (whose system model is similar to the one in Section 3) using these metrics. We also evaluate the truthfulness and robustness against false-name attacks, by selecting some users, letting them submit untruthful bids or launch false-name attacks unilaterally, and monitoring the corresponding utilities.

Simulation Setup: We evaluate the platform utility, average user utility, and running time by varying the number of users n and the number of jobs m from 50 to 800 with an increment of 50, respectively. To evaluate the impact of n, we fixed $m = 100$. Similarly, to evaluate the impact of m, we fixed $n = 100$. We set $|\Gamma| = 10$, and generated each job as a random multi-subset of Γ with a maximum size of 10. The valuation of each job is uniformly distributed over $(0, 80]$. Each user can submit up to 5 random multi-subsets of $\mathcal{T}$. The size of each multi-subset does not exceed 10, and the ask-per-task value for each subset is uniformly distributed over $(0, 2.5]$. The results are averaged over 100 instances for each parameter configuration.

B. Results and Observations

Fig. 4 shows the impact of m and n on platform utility for **TAFA** and IMC-SM. In Fig. 4(a), with the increase of m, platform utilities in **TAFA** and IMC-SM increase. This is because users can complete more jobs when m increases. However, with more jobs, the competition among users are less fierce, and thus the payment for each user increases. That is why the growth of platform utility is slower than the linear speed in terms of m. On the other hand, with more users, the competition among users is more fierce, which leads to decrements in payments. That explains why the platform utilities in **TAFA** and IMC-SM increase in Fig. 4(b). When the market is saturated with users, the platform utility stays steady. Another observation is that the platform utility of IMC-SM is higher than that of **TAFA**. This is because **TAFA** takes the minimum per-task ask from users as the selection metric while IMC-SM uses the minimum ask value. Therefore, the total payment of IMC-SM is lower than that of **TAFA**. Since the margin between the platform utilities of **TAFA** and IMC-SM is relatively small, we take this margin as a tolerable tradeoff to the robustness against false-name attacks.

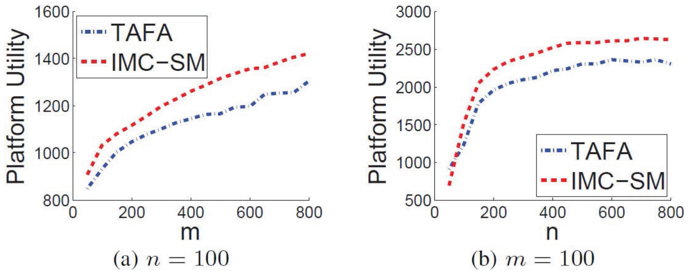


Fig. 4. Platform Utility

Fig. 5 shows the average user utility, defined as the total utilities of all users over the number of users. With the increase of m, average user utility rises and then remains steady. The reason is that with more tasks to be completed, the maximum per-task ask in Algorithm 2 increases for each user. With the increase of n, the average user utility drops dramatically for both **TAFA** and IMC-SM. This is because with more users, the competition among users becomes more fierce and it leads to a decrement in payments. Furthermore, the average user utility of **TAFA** is higher than that of IMC-SM, as the payment for users in **TAFA** is higher than that of IMC-SM. The detailed reasoning has been provided in the previous paragraph.

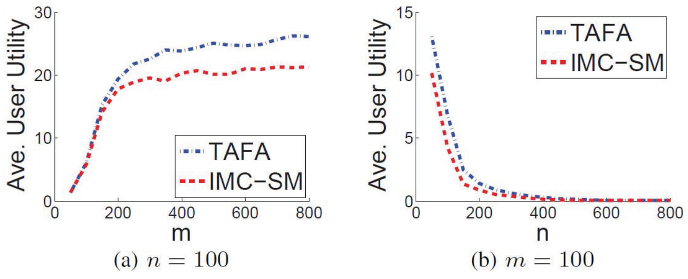


Fig. 5. Average user utility

Fig. 6 shows the impact of m and n on the running time of **TAFA** and IMC-SM. We only record the time spent on winning user selection and pricing steps for IMC-SM, in order to make a fair comparison with **TAFA**. With the increment of m and n, the running time of **TAFA** and IMC-SM increases with an approximated speed of $O(m)$ and $O(n^2)$, respectively. These results match our analysis in Lemmas 6.3. Furthermore, the time consumption of **TAFA** is higher than that of IMC-SM, since **TAFA** calculates the per-task payment with two more *for*-loops in Algorithm 2, Line 11 to Line 15. With the relatively small gap between the two running times, the extra time consumption of **TAFA** is tolerable.

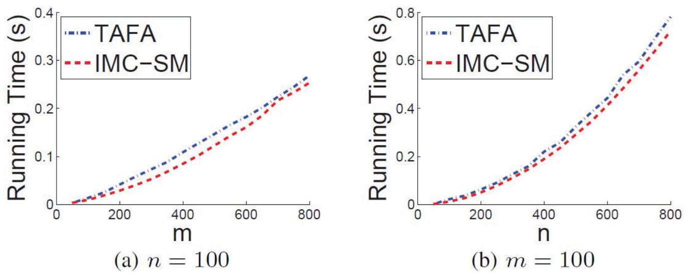


Fig. 6. Running time

Fig. 7 shows the impact of (untruthful) bids and false-name attacks on user utilities. We fixed $m = 100$ and $n = 1000$, and selected three users: 30, 51, and 62. User 30 is a loser when revealing its cost 9.66. Users 51 and 62 are winners when revealing their costs 4.11 and 5.25, respectively. To validate truthfulness, we let each of these users unilaterally change its bid in $[0, 15]$, see Fig. 7(a). We observe that a user cannot increase its utility by bidding dishonestly. To validate robustness against false-name attacks, we let each of these users create up to 8 false names. For each false name, the submitted task set is a subset of the submitted tasks of the user, and the ask-per-task value is randomly distributed around its cost-per-task value. The results are presented in Fig. 7(b). We observe that a user cannot increase its utility by unilaterally launching false-name attacks.

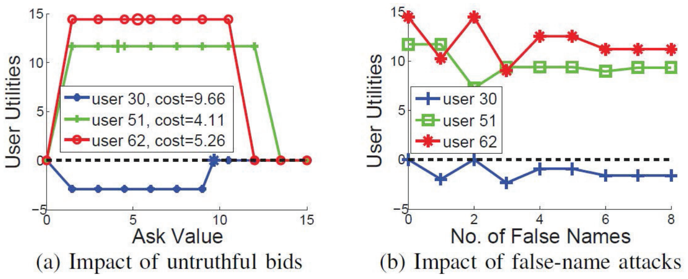


Fig. 7. Impact of (untruthful) bids and false-name attacks on user utilities

8. Conclusions

In this paper, we studied truthful and false-name proof auctions in crowdsourcing, where collaborative efforts from crowdsourcing users are needed. We designed an incentive mechanism

TAFA, and proved that it is individually rational, budget-balanced, computationally efficient, and truthful. We also proved that **TAFA** provides countermeasures against false-name attacks. Extensive simulation results are presented to further study the mechanism and confirm our analysis.

References

[1] Amazon Mechanical Turks, https://www.mturk.com/mturk/welcome.

[2] Creek Watch, http://creekwatch.researchlabs.ibm.com.

[3] N. Devanur, M. Mihail, and V. Vazirani, "Strategyproof cost-sharing mechanisms for set cover and facility location games," in *EC '03*, pp. 108–114.

[4] D. DiPalantino and M. Vojnovic, "Crowdsourcing and all-pay auctions," in *ACM EC'09*.

[5] J. Dong, Y. Xiao, Z. Ou, Y. Cui, and A. Yla-Jaaski, "Indoor tracking using crowdsourced maps," in *ACM/IEEE IPSN'16*, pp. 1–6.

[6] Z. Feng, Y. Zhu, Q. Zhang, H. Zhu, J. Yu, J. Cao, and L. Ni, "Towards truthful mechanisms for mobile crowdsourcing with dynamic smartphones," in *IEEE ICDCS'14*, pp. 11–20.

[7] I. Koutsopoulos, "Optimal incentive-driven design of participatory sensing systems," in *IEEE INFOCOM'13*, pp. 1402–1410.

[8] X.-Y. Li, Z. Sun, W. Wang, X. Chu, S. Tang, and P. Xu, "Mechanism design for set cover games with selfish element agents," *Theor. Comput. Sci.*, vol. 411, no. 1, pp. 174–187, 2010.

[9] Q. Ma, L. Gao, Y. F. Liu, and J. Huang, "A contract-based incentive mechanism for crowdsourced wireless community networks," in *IEEE WiOpt'16*, pp. 1–8.

[10] M. Pouryazdan, B. Kantarci, T. Soyata, and H. Song, "Anchor-assisted and vote-based trustworthiness assurance in smart city crowdsensing," *IEEE Access'16*, vol. 4, pp. 529–541.

[11] Sensorly, https://www.sensorly.com.

[12] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a service: Challenges, solutions and future directions," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3733–3741, 2013.

[13] Y. Singer and M. Mittal, "Pricing mechanisms for crowdsourcing markets," in *WWW'13*, pp. 1157–1166.

[14] A. Singla and A. Krause, "Truthful incentives in crowdsourcing tasks using regret minimization mechanisms," in *WWW'13*, pp. 1167–1178.

[15] K. Terada and M. Yokoo, "False-name-proof multi-unit auction protocol utilizing greedy allocation based on approximate evaluation values," in *AAMAS'03*, pp. 48–62.

[16] T. Todo, A. Iwasaki, M. Yokoo, and Y. Sakurai, "Characterizing false-name-proof allocation rules in combinatorial auctions," in *AAMAS '09*, pp. 265–272.

[17] Q. Wang, B. Ye, B. Tang, T. Xu, S. Guo, S. Lu, and W. Zhuang, "Aletheia: Robust large-scale spectrum auctions against false-name bids," in *ACM MobiHoc'15*, pp. 27–36.

[18] Waze, https://www.waze.com.

[19] Yahoo! Answers, http://answers.yahoo.com/.

[20] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *ACM MOBICOM'12*, pp. 173–184.

[21] M. Yokoo, Y. Sakurai, and S. Matsubara, "The effect of false-name declarations in mechanism design: towards collective decision making on the internet," in *IEEE ICDCS'00*, pp. 146–153.

[22] ——, "Robust combinatorial auction protocol against false-name bids," *AI'01*, vol. 130, no. 2, pp. 167 – 181.

[23] ——, "Robust multi-unit auction protocol against false-name bids," in *IEEE IJCAJ'01*.

[24] Q. Zhang, Y. Wen, X. Tian, X. Gan, and X. Wang, "Incentivize crowd labeling under budget constraint," in *IEEE INFOCOM'15*.

[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 TPDS*, vol. PP, no. 99, pp. 1–1, 2014.

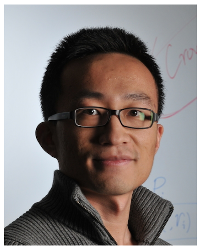
[26] X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, "Truthful incentive mechanisms for crowdsourcing," in *IEEE INFOCOM'15*, pp. 2830–2838.

[27] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *IEEE INFOCOM'14*, pp. 1213–1221.

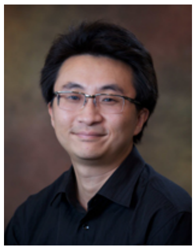
Xiang Zhang (Student Member 2013) received his B.S. degree from University of Science and Technology of China, Hefei, China, in 2012. Currently he is a Ph.D student in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. His research interests include network economics, incentive mechanism design, and game theory in crowdsourcing and cognitive radio networks.

Guoliang Xue (Member 1996, Fellow, 2011) is a Professor of Computer Science and Engineering at Arizona State University. He received the PhD degree in computer science from the University of Minnesota in 1991. His research interests include survivability, security, and resource allocation issues in networks. He is the Area Editor of *IEEE Transactions on Wireless Communications* for the area of Wireless Networking. He is the Vice President for Conferences of the IEEE Communications Society.

Ruozhou Yu (Student Member 2013) received his B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently pursuing the Ph.D degree in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. His interests include software-defined networking, data center networking, and network function virtualization.



Dejun Yang (Student Member 2008, Member 2013) is the Ben L. Fryrear Assistant Professor of Computer Science at Colorado School of Mines. He received his PhD degree from Arizona State University in 2013 and BS degree from Peking University in 2007. His research interests lie in optimization and economic approaches to networks. He received Best Paper Awards at IEEE ICC'2011 and IEEE ICC'2012. He has served as a TPC member for many conferences including IEEE INFOCOM.



Jian Tang is an associate professor in the Department of Electrical Engineering and Computer Science at Syracuse University. He earned his Ph.D degree in Computer Science from Arizona State University in 2006. His research interests lie in the areas of Cloud Computing, Big Data and Wireless Networking. He received an NSF CAREER award in 2009 and a Best Paper Award in IEEE ICC'2014. He serves as an editor for IEEE Transactions on Vehicular Technology and an editor for IEEE Internet of Things Journal.