Matchmaker: Stable Task Assignment With Bounded Constraints for Crowdsourcing Platforms

Xiaoyan Yin, *Member, IEEE*, Yanjiao Chen, *Senior Member, IEEE*, Cheng Xu, Sijia Yu, and Baochun Li, *Fellow, IEEE*

***Abstract*—Crowdsourcing has become a popular paradigm to leverage the collective intelligence of massive crowd workers to perform certain tasks in a cost-effective way. Task assignment is an essential issue in crowdsourcing platforms owing to heterogeneous tasks and work skills. In this article, we focus on assigning workers with diversified skill levels to crowdsourcing tasks with different quality requirements and budget constraints. Task assignment is fundamentally a many-to-one matching problem, where one task is allocated to multiple users who can meet the minimum quality requirement of the task within the limited budget. While most existing works try to maximize the utility of the crowdsourcing platform, we take into account the individual preferences of crowdsourcers and workers toward each other to ensure the stability of task assignment results. In this article, we propose task assignment mechanisms that can guarantee stable outcomes for the many-to-one matching problem with lower and upper bounds (i.e., quality requirement and budget constraint) in regard to heterogeneous worker skill levels. Extensive simulation results show that the proposed algorithms can greatly improve the success ratio of task accomplishment and worker happiness compared with existing algorithms.**

***Index Terms*—Crowdsourcing, matching, quality requirement, task assignment.**

I. Introduction

CROWDSOURCING combines efforts from massive workers with heterogeneous skills to accomplish tasks with high qualities and low costs, e.g., new product design, traffic monitoring, and image labeling. Crowdsourcing platforms offer an open and reliable environment for crowdsourcers and individual workers to communicate with each other. Crowdsourcers publish their tasks and hope to recruit workers with sufficient skills and affordable costs. Workers apply to perform tasks in return for certain payments. Crowdsourcers and workers have different preferences toward each other in terms of skill levels and costs.

Task assignment is one of the most fundamental problems in crowdsourcing. Existing algorithms [1]–[10] address the problem of task assignment to realize different design goals for heterogeneous systems, e.g., mobile crowdsensing, fog-based crowdsourcing, and Internet of Vehicles (IoV). Considering limited budgets, a two-phase exploration–exploitation algorithm was designed to maximize the payoff of the crowdsourcer by allocating workers with different skill levels to different tasks [2]. To guarantee the deadline of tasks, the mobility and active time of workers as well as the geotemporal requirements of tasks had been considered in [3]–[5] and [7]. In [4] and [7], the uncertainty of user mobility was tackled. The reliability of task results was studied in [8]. In [9], the relationship between data quality and the profit of workers was taken into consideration. In [10], a utility maximization problem was formulated jointly considering roadside unit deployment and service task assignment. Nonetheless, most existing works only consider the total utility of the crowdsourcing platform but ignore individual preferences of crowdsourcers and workers. A global-optimal task assignment result may deviate if individual crowdsourcers and workers have incentives to seek for better options. Therefore, it is important to ensure stability in an open crowdsourcing platform, where the task assignment results cater to the individual utility of crowdsourcers and workers.

In this article, we present a stable matching framework for task assignment in crowdsourcing, as shown in Fig. 1. In particular, we utilize the many-to-one matching where one task can be assigned to multiple workers as long as the payments can be covered by the budget of the task. Workers have diversified skills due to different experiences, capabilities, and proficiencies. Naturally, high-skill workers cost more than low-skill workers. A crowdsourcer should hire a set of workers with enough skills to successfully accomplish the tasks and also meet the budget constraint. Therefore, the formulated many-to-one matching framework has both lower bound (quality requirement) and upper bound (budget constraint).

As discussed before, we focus on stable rather than optimal matching to ensure that crowdsourcers and workers are willing to obey the task assignment results. Nonetheless, conventional stable solutions to many-to-one matching problems with lower

Manuscript received April 20, 2020; revised June 27, 2020; accepted July 29, 2020. Date of publication August 5, 2020; date of current version January 22, 2021. The work of Xiaoyan Yin was supported in part by the National Natural Science Foundation of China under Grant 61872295; in part by the Shaanxi Natural Science Foundation under Grant 2020JM-416; and in part by the ShaanXi Science and Technology Innovation Team Support Project under Grant 2018TD-026. The work of Yanjiao Chen was supported in part by the National Natural Science Foundation of China under Grant 61972296 and Grant 61702380; and in part by the Wuhan Advanced Application Project under Grant 2019010701011419. *(Corresponding author: Yanjiao Chen.)*

Xiaoyan Yin, Cheng Xu, and Sijia Yu are with the School of Information Science and Technology, Shaanxi International Joint Research Center for Internet of Things, Northwest University, Xi'an 710127, China (e-mail: yinxy@nwu.edu.cn; xucheng@stumail.nwu.edu.cn; dongyu@stumail.nwu.edu.cn).

Yanjiao Chen is with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: chenyanjiao@whu.edu.cn).

Baochun Li is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S3G4, Canada (e-mail: bli@ece.toronto.edu).

Digital Object Identifier 10.1109/JIOT.2020.3014440

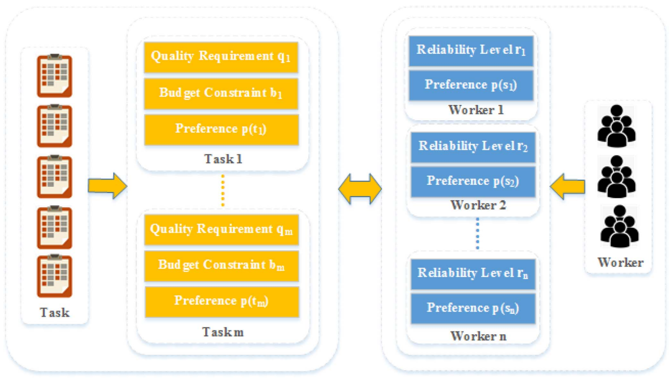

Fig. 1. Architecture of our proposed task assignment framework for crowdsourcing platforms.

and upper bounds cannot be applied here since agents are assumed to be homogeneous in "size," i.e., workers have the same skill levels and ask for the same amount of payment. To introduce heterogeneity into the matching framework is quite challenging since different combinations of workers have different costs and aggregate skill levels even if the numbers of workers are the same. In this article, we design novel algorithms that can efficiently converge to stable task assignments. Through an extensive array of simulations, we compare the proposed algorithms with benchmark algorithms for a many-to-one matching problem with heterogeneous agents but no lower bounds. We show that our proposed matching algorithms can greatly improve the success ratio of completed tasks and worker happiness at the expense of a slight increase in running times.

In this article, we make the following key contributions.

1) We propose an innovative many-to-one matching framework to model the problem of task assignment for crowdsourcing platforms, resolving conflicting interests between workers and tasks. Unlike existing utility optimization-based solutions, our framework aims to find a stable matching result that individual workers are willing to conform to.
2) We develop novel stable matching algorithms for our many-to-one matching framework with a lower bound (quality requirement), an upper bound (budget constraint), and heterogeneous workers. Through theoretical analysis, our proposed algorithms have been proved to produce stable task assignment results.
3) Our extensive simulation results validate the effectiveness of our proposed algorithms, showing their benefits to both tasks and workers in terms of the success ratio of task accomplishment and worker happiness.

A preliminary version of this article has been published in IEEE/ACM IWQoS 2017 [11], featuring our proposed task assignment with the quality requirement (TAQR) algorithm. In this journal version, we have proposed a new extended stable task assignment (ESTA) algorithm with a guaranteed requirement.

The remainder of this article is organized as follows. We introduce the system model in detail in Section II. The ESTA algorithm is proposed, and the stability of its matching result is proved in Section III. The simulation results are presented in Section IV. We survey the related works in Section V. Finally, we conclude this article in Section VI.

II. System Model

We consider a crowdsourcing platform that consists of a set $\mathcal{S}$ of workers. Worker $s \in \mathcal{S}$ has a quality level of r_s, which reflects how good her response to a task is. In this article, we make the simplifying assumption that a worker's quality level r_s is the same for all tasks. This is reasonable for scenarios where the crowdsourcing tasks have similar difficulty levels, e.g., crowdsensing the noise or pollution levels in different locations. In our future work, we will consider the case where a worker's performance across tasks may be different. Different workers have different quality levels due to their diversity in skills, experience, and proficiency. We assume that the workers' quality levels are known to the crowdsourcers. To achieve this, the crowdsourcing platform simply keeps a reputation system to update the quality level of each worker based on their past performance. A worker with a higher quality level will ask for higher compensation for performing a task. This is reasonable since those workers usually spend more time and energy in training, and crowdsourcers are willing to pay more for a more reliable result. We use $f(r_s)$ to denote the payment that worker s requires for completing a task. $f(\cdot)$ is a monotonically increasing function, and we have $f(0) = 0$.

We assume that there is a set $\mathcal{T}$ of crowdsourcers on the crowdsourcing platform, each publishing a specific task. In practice, a crowdsourcer is allowed to publish multiple tasks on the crowdsourcing platform. In this case, we can regard such a crowdsourcer as multiple virtual crowdsourcers, each with a single task. Under this circumstance, with a little abuse of notations, we use $\mathcal{T}$ to denote the set of tasks as well. Since crowdsourcing tasks rely on the wisdom of the crowd, a task can only be successfully accomplished if the crowdsourcer can recruit enough workers. Let q_t denote the quality requirement of task $t \in \mathcal{T}$. We make the simplifying assumption that a set $\mathcal{A}$ of workers can achieve a total quality level of $\sum_{s\in\mathcal{A}} r_s$ when cooperating on the same task. Therefore, the crowdsoucer of task t must ensure that she can attract enough workers to fulfill the quality requirement. If the aggregate quality level of workers assigned to task t is not smaller than its quality requirement, i.e., $\sum_{s\in\mathcal{A}} r_s \geq q_t$, task t can be finished with acceptable results; otherwise, task t will fail.

Even though a larger number of workers means a higher quality level, each crowdsourcer is bounded by her budget, which limits the number of workers she can hire. Let b_t denote the budget of crowdsourcer t. Hence, it must be conformed that $\sum_{s\in\mathcal{A}} f(r_s) \leq b_t$. In this article, we make the simplifying assumption that $f(r_s) = r_s$. In future work, we will study more general forms of function $f(\cdot)$. Combining the quality requirement and the budget constraint, we have the condition for the assignment for a task to be successful: $q_t \leq \sum_{s\in\mathcal{A}} r_s \leq b_t$.

Each worker has a preference list over all the tasks. Let $\succ_s$ denote the complete, reflexive, and transitive preference relation of worker s. $t \succ_s t'$ indicates that worker s is more willing

TABLE I
KEY PARAMETERS

Parameter	Definition
$\mathcal{S}$	set of workers
$\mathcal{T}$	set of tasks
r_s	quality level of worker s
q_t	quality requirement of task t
b_t	budget constraint of task t
$x_{s,t}$	whether worker s is assigned to task t
μ	task assignment matching result
$p(s)$	the preference list of worker s
$p(t)$	the preference list of task t
$Q_{\mathcal{A}}$	total quality of workers in set $\mathcal{A}$
ΔQ	total quality requirement that is fulfilled
ΔR	total skill level of unassigned workers

to work on task t than task t'. For example, if two crowdsourcers aim to collect traffic information at two different locations, a worker may prefer the location near her residence to the location that is far away. Due to worker diversity, their preference lists are different from each other. In this article, we assume that a worker has the same quality level toward all tasks, but it is noteworthy that a worker's preferences toward different tasks may also be affected by her quality level toward the tasks. A worker may be more interested in performing a task that she is more skilled in this task. Also, a worker's skill will evolve over time, indicating that their preference lists may change over time. In the future, we will consider the impact of skill levels on workers' preferences.

Similarly, each crowdsourcer has a preference list over all the workers, depending on the workers' quality levels. Let $\succ_t$ denote the complete, reflexive, and transitive preference relation of crowdsourcer t. Naturally, a crowdsourcer will prefer a worker with a higher quality level. Therefore, the preference lists of all crowdsourcers are the same since we assume that each worker has the same quality level toward all the tasks. Furthermore, we assume that all the preference lists are fixed and known to all the participants (workers and crowdsourcers) of the crowdsourcing platform. Without loss of generality, we assume that all the tasks are acceptable to every worker. If a worker is reluctant to do some of the tasks, she can simply insert an *empty task* in the preference list and put all the unacceptable tasks behind the empty task. We summarize major parameters and their definitions in Table I.

In this article, we impose the restriction that each worker can only focus on one task, but each task can involve multiple workers. The objective of the crowdsourcing platform operator is to realize a stable task assignment that takes into consideration the preferences of workers and crowdsourcers, as well as quality requirements and budget constraints of all tasks. We formulate the task assignment for a crowdsourcing platform as a many-to-one matching problem with a hard upper bound (budget constraint) and a soft lower bound (quality requirement), as follows.

Definition 1 (Task Assignment): A task assignment μ for the crowdsourcing platform is a mapping $\mu : \mathcal{S}\bigcup\mathcal{T} \rightarrow 2^{\mathcal{S}}\bigcup\mathcal{T}$, which satisfies:

1) $\mu(s) \in \mathcal{T}$ for all $s \in \mathcal{S}$;
2) $\mu(t) \subseteq \mathcal{S}$ for all $t \in \mathcal{T}$;
3) for any $s \in \mathcal{S}$ and $t \in \mathcal{T}$, we have $\mu(s) = t$ if and only if $s \in \mu(t)$, where $\mu(t)$ is the set of workers matched to task t;
4) *Hard Upper Bound (Budget Constraint):* $\sum_{s\in\mu(t)} r_s \leq b_t$ for all $t \in \mathcal{T}$;
5) *Soft Lower Bound (Quality Requirement):* Let $\widetilde{\mathcal{T}} = \{t | t \in \mathcal{T}, \sum_{s\in\mu(t)} r_s \geq q_t\}$ denote the set of tasks whose quality requirements are satisfied. Define success ratio as $[(|\widetilde{\mathcal{T}}|)/(|\mathcal{T}|)] \times 100\%$, in which $|\cdot|$ is the number of elements in a set. The success ratio evaluates the extent to which the task assignment μ fulfills the quality requirements of all crowdsourcing tasks.

Most of the existing many-to-one matching models do not impose a lower bound constraint as in our model, which involves the quality requirement of crowdsourcing tasks that rely heavily on joint efforts. We pose a soft lower bound on the task assignment μ since it is difficult to decide whether there exists a task assignment that attains a 100% success ratio. In the existing many-to-one matching models that do consider lower bounds, it is assumed that agents are homogeneous in "size," that is, each worker has a uniform quality level of $r_s = 1 \ \ \forall s \in \mathcal{S}$. Under this simplified model, a task assignment that fully satisfies the lower bound exists if and only if the total quality of all workers is not smaller than the total quality requirement of all tasks, i.e., $\sum_{s\in\mathcal{S}} r_s \geq \sum_{t\in\mathcal{T}} q_t$.

Unfortunately, when workers have heterogeneous quality levels, $\sum_{s\in\mathcal{S}} r_s \geq \sum_{t\in\mathcal{T}} q_t$ no longer guarantees that the quality requirement of every task can be satisfied. For example, if we have two workers with quality levels as 0.3 and 0.7, respectively, and two tasks with quality requirements as 0.4 and 0.5, respectively. We have $0.3+0.7 > 0.4+0.5$, but no task assignment can simultaneously meet the quality requirements of the two tasks. Let $x_{s,t}$ be the task assignment indicators, and $x_{s,t} = 1$ if $\mu(s) = t$; otherwise, $x_{s,t} = 0$. To validate the existence of a task assignment with a 100% success ratio is equivalent to checking whether the following integer linear programming problem has a feasible solution:

$$\max_{x_{s,t}} \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}} x_{s,t} \tag{1}$$

$$\text{subject to} \quad \sum_{t} x_{s,t} \leq 1 \quad \forall s \tag{2}$$

$$q_t \leq \sum_{s} x_{s,t} r_s \leq b_t \quad \forall t \tag{3}$$

$$x_{s,t} \in \{0, 1\} \quad \forall\ s, t. \tag{4}$$

Instead of solving the integer linear programming problem, we only have to verify whether the feasible region is empty or not because we only care about the existence of a task assignment with a 100% success ratio. Even so, the problem is NP-hard. Therefore, in this article, we only treat quality requirements as soft lower bounds, but will try to improve the success ratio as much as possible.

A task assignment is stable if no worker or crowdsourcer has the incentive to deviate from the assignment result. Workers and crowdsourcers are selfish and rational individuals who will break off from the task assignment if they have better choices. On a crowdsourcing platform, where workers can freely choose tasks and crowdsourcers can freely hire or sack workers, a task assignment can be implemented only if it is stable. A stable task assignment features *individual rationality*, *fairness*, and *nonwastefulness*.

Definition 2 (Individual Rationality): A task assignment μ is individually rational if:

1) every worker prefers being assigned to the current task of being unassigned;
2) every crowdsourcer prefers the current set of assigned workers to any subset of these workers.

Being individually rational is the basic property of a task assignment. It ensures that workers are not reluctant to perform crowdsourcing tasks and crowdsourcers are willing to accomplish their tasks in the form of crowdsourcing. To define fairness [12], we have to introduce the concept of type I blocking pair.

Definition 3 (Type I Blocking Pair): Given a task assignment μ, worker s and task t form a type I blocking pair (s, t), if there exists a nonempty subset of workers, denoted by $\mathcal{A}$, who are matched to task t, i.e., $\mathcal{A} \subseteq \mu(t)$, and satisfy the following conditions.

1) Worker s prefers task t to her current assignment $\mu(s)$. Crowdsourcer t prefers worker s to any worker in $\mathcal{A}$, and crowdsourcer t prefers worker s to the whole worker set $\mathcal{A}$.
2) Worker s can displace the workers in $\mathcal{A}$ without violating the budget constraint of task t.
3) The leaving of worker s will not violate the quality requirement of her currently assigned task $\mu(s)$.

Mathematically speaking, worker s and task t form a type I blocking pair if there exists a nonempty subset of workers $\mathcal{A} \subseteq \mu(t)$, and:

1) $t \succ_s \mu(s)$, $s \succ_t s' \ \ \forall s' \in \mathcal{A}$, and $r_s \geq \sum_{s' \in \mathcal{A}} r_{s'}$;
2) $r_s + \sum_{s' \in \mu(t) \setminus \mathcal{A}} r_{s'} \leq b_t$;
3) $\sum_{s' \in \mu(\mu(s))} r_{s'} - r_s \geq q_{\mu(s)}$.

The type I block pair makes a task assignment unstable because the worker in concern has the chance to shift to a more preferred task by replacing some of the less-preferred workers who have been assigned to that task. The definition of type I blocking pair is simpler in many-to-one matching problems without lower bounds. In particular, condition 3) is not entailed. However, in our proposed framework, condition 3) is needed to suppress the violation of quality requirements.

Definition 4 (Fairness): A task assignment μ is fair if and only if there is no type I blocking pairs.

A fair task assignment indicates that it is fair for a worker to be assigned to her current task because she cannot replace the workers assigned to her more-preferred tasks. Apart from the type I blocking pairs, we also have the type II blocking pair.

Definition 5 (Type II Blocking Pair): Given a task assignment μ, worker s and task t form a type II blocking pair (s, t), if:

1) worker s prefers task t to her current assignment $\mu(s)$;
2) add worker s to task t will not violate the budget constraint of task t;
3) the leaving of worker s will not violate the quality requirement of her currently assigned task $\mu(s)$.

Mathematically speaking, under the task assignment μ, worker s and task t form a type II blocking pair, if:

1) $t \succ_s \mu(s)$;
2) $r_s + \sum_{s' \in \mu(t)} r_{s'} \leq b_t$;
3) $\sum_{s' \in \mu(\mu(s))} r_{s'} - r_s \geq q_{\mu(s)}$.

The type II blocking pair makes a task assignment unstable because a crowdsourcer has enough budget to hire one more worker who is willing to work for her.

Definition 6 (Nonwastefulness): A feasible task assignment μ is nonwasteful if and only if there are no type II blocking pairs.

Nonwastefulness [12] ensures that crowdsourcers make the best use of their budgets in recruiting workers. The difference between type I and type II blocking pairs is whether a worker can take the budget paid to some of the workers currently assigned to a task. In type II blocking pair (s, t), only the remaining budget of task t is considered to cover the payment to worker s; whereas in a type I blocking pair (s, t), the budget for task t's currently assigned workers can be reclaimed for compensating worker s.

III. Extended Stable Task Assignment With Guaranteed Quality

In this section, we design a stable task assignment algorithm with guaranteed quality. The basic idea is to divide each original task into two extended tasks, called the regular task and the shadow task. The regular task is responsible for the minimum quality requirement of the original task, and thus in an outstanding important position. The shadow task is responsible for the remaining budget (the budget subtract the minimum quality requirement) to improve the quality of task complishment. Let t^r and t^d denote the regular task and the shadow task of the original task t, respectively. We set the budget of the regular task t^r as $b_{t^r} = q_t$ and the budget of the shadow task t^d as $b_{t^d} = b_t - q_t$. By introducing two kinds of extended tasks, TAQRs and budget constraints are transformed into task assignment with only budget constraints. Thus, there is no minimum quality requirement for all extended tasks. Generally speaking, if we reserve at least $\sum_t q_t$ for the regular tasks, their quality will be guaranteed, which implies that the minimum quality requirements of all original tasks will be satisfied. Nevertheless, due to the heterogeneous skill levels of workers, to calculate how many workers to preserve for regular tasks in task assignment is a challenging problem.

A. Problem Transformation

The transformation of the task assignment problem with minimum quality requirement into that without minimum quality requirement is shown in Algorithm 1. The worker set remains the same. The preference lists of the regular task and shadow task are the same as the original task (line 7). The preference list of each worker is rebuilt by inserting the shadow task right after the regular task with the sequence of the original preference list unchanged (line 11).

Algorithm 1 Task Assignment Transformation

```
Input: Worker set $\mathcal{S}$, task set $\mathcal{T}$, preference lists of workers $\{\succ_s\}_{s\in\mathcal{S}}$, preference lists of tasks $\{\succ_t\}_{t\in\mathcal{T}}$, quality requirements of tasks $\{q_t\}_{t\in\mathcal{T}}$, and budget constraints of tasks $\{b_t\}_{t\in\mathcal{T}}$.
Output: Worker set $\widetilde{\mathcal{S}}$, task set $\widetilde{\mathcal{T}}^r$, $\widetilde{\mathcal{T}}^d$, preference lists of workers $\{\widetilde{\succ}_s\}_{s\in\widetilde{\mathcal{S}}}$, preference lists of tasks $\{\widetilde{\succ}_{t^r}\}_{t^r\in\widetilde{\mathcal{T}}^r}$, $\{\widetilde{\succ}_{t^d}\}_{t^d\in\widetilde{\mathcal{T}}^d}$, and budget constraints of tasks $\{b_{t^r}\}_{t^r\in\widetilde{\mathcal{T}}^r}$, $\{b_{t^d}\}_{t^d\in\widetilde{\mathcal{T}}^d}$.
1:  $\widetilde{\mathcal{S}} = \mathcal{S}$.
2:  $\widetilde{\mathcal{T}}^r = \Phi$, $\widetilde{\mathcal{T}}^d = \Phi$.
3:  for $t \in \mathcal{T}$ do
4:      $\widetilde{\mathcal{T}}^r = \widetilde{\mathcal{T}}^r \cup \{t^r\}$,
5:      $\widetilde{\mathcal{T}}^d = \widetilde{\mathcal{T}}^d \cup \{t^d\}$.
6:      $b_{t^r} = q_t$, $b_{t^d} = b_t - q_t$.
7:      $\widetilde{\succ}_{t^r} := \widetilde{\succ}_{t^d} := \succ_t$.
8:  end for
9:  $\widetilde{\mathcal{T}} = \widetilde{\mathcal{T}}^r \cup \widetilde{\mathcal{T}}^d$.
10: for $s \in \mathcal{S}$ do
11:     Change $\succ_s := t_{s,1} \succ_s t_{s,2} \succ_s \cdots$ into $\widetilde{\succ}_s := t^r_{s,1} \widetilde{\succ}_s t^d_{s,1} \widetilde{\succ}_s t^r_{s,2} \widetilde{\succ}_s t^d_{s,2} \widetilde{\succ} \cdots$
12: end for
```

B. Proposed Algorithm ESRT

With the transformed workers and tasks, the ESTA algorithm with the guaranteed quality requirement is shown in Algorithm 2. Tasks are partitioned into regular tasks and shadow tasks and have different rules to decide whether to accept or reject a worker.

1) If task t belongs to the regular task set $\widetilde{T}^r$, no more than $|\mathcal{A}|$ workers will be selected (lines 7 and 8), where $\sum_{s\in\mathcal{A}} r_s \leq b_t$.
2) If task t belongs to the shadow task set $\widetilde{T}^d$, we need to take all the other regular tasks into account to ensure their quality, i.e., the minimum quality requirements of all original tasks. In Algorithm 2, in line 10, we calculate $\Delta\widetilde{Q}$, the total quality requirement that has not been fulfilled yet. In line 11, we compute $\Delta\widetilde{R}$, the total quality of the unassigned workers (including worker s). It is obvious that the quality requirements of other tasks cannot be fulfilled without worker s if $\Delta\widetilde{R} - r_s$ is less than $\Delta\widetilde{Q}$. In this way, we check whether worker s can be assigned to task t.
    a) If $\Delta\widetilde{R} - r_s < \Delta\widetilde{Q}$, worker s cannot be assigned to task t, and the task assignment remains unchanged.
    b) If $\Delta\widetilde{R} - r_s \geq \Delta\widetilde{Q}$, task assignment will be determined by Algorithm 3 (the ReDA).

Given the matching result $\widetilde{\mu}$ of Algorithm 2, we can obtain the final task assignment as $\forall t \in \widetilde{\mathcal{T}}$, $\widetilde{\mu}(t) = \widetilde{\mu}(t^r) \cup \widetilde{\mu}(t^d)$, as shown in lines 19–21 $\forall s \in \widetilde{\mathcal{S}}$, if $s \in \widetilde{\mu}(t^r)$ or $s \in \widetilde{\mu}(t^d)$, we have $s \in \widetilde{\mu}(t)$, as shown in lines 22–26.

Toy Example: We use the settings as shown in Table II. Using Algorithm 1, we can obtain the transformed workers and tasks, as shown in Table III. We have six workers with updated preference lists and four extended tasks, i.e., two regular tasks and two shadow tasks, with their updated preference lists and budget constraints. The task assignment algorithm runs as follows.

Algorithm 2 ESTA Algorithm With Minimum Requirement

```
Input: Worker set $\widetilde{\mathcal{S}}$, task set $\widetilde{\mathcal{T}}^r$, $\widetilde{\mathcal{T}}^d$, preference lists of workers $\{\widetilde{\succ}_s\}_{s\in\widetilde{\mathcal{S}}}$, preference lists of tasks $\{\widetilde{\succ}_{t^r}\}_{t^r\in\widetilde{\mathcal{T}}^r}$, $\{\widetilde{\succ}_{t^d}\}_{t^d\in\widetilde{\mathcal{T}}^d}$, and budget constraints of tasks $\{b_{t^r}\}_{t^r\in\widetilde{\mathcal{T}}^r}$, $\{b_{t^d}\}_{t^d\in\widetilde{\mathcal{T}}^d}$.
Output: Task assignment $\widetilde{\mu}$.
1:  Initialization $\widetilde{\mu}(s) = \emptyset$, $\widetilde{\mu}(t) = \emptyset$ $\forall s \in \widetilde{\mathcal{S}}$ $\forall t \in \widetilde{\mathcal{T}}$.
2:  for $s \in \widetilde{\mathcal{S}}$ do
3:      $p(s)$ := ordered list of tasks according to $\widetilde{\succ}_s$.
4:  end for
5:  while $\exists s$, $\widetilde{\mu}(s) = \emptyset$ and $p(s) \neq \emptyset$ do
6:      $t$ = highest ranked task in $p(s)$. Remove $t$ from $p(s)$.
7:      if $t \in \widetilde{\mathcal{T}}^r$ then
8:          $\widetilde{\mu} = ReDA(s, t, \widetilde{\mu})$.
9:      else
10:         $\Delta\widetilde{Q} = \sum_{t\in\widetilde{\mathcal{T}}}(b_{t^r} - \widetilde{Q}_{\widetilde{\mu}(t^r)} - \widetilde{Q}_{\widetilde{\mu}(t^d)})^+$.
11:         $\Delta\widetilde{R} = \sum_{\widetilde{\mu}(s)=\emptyset} r_s$.
12:         if $\Delta\widetilde{R} - r_s < \Delta\widetilde{Q}$ then
13:             $\widetilde{\mu}(t) = \widetilde{\mu}(t)$, $\widetilde{\mu}(s) = \emptyset$.
14:         else
15:             $\widetilde{\mu} = ReDA(s, t, \widetilde{\mu})$.
16:         end if
17:     end if
18: end while
19: for $\forall t \in \widetilde{\mathcal{T}}$ do
20:     $\widetilde{\mu}(t) = \widetilde{\mu}(t^r) \cup \widetilde{\mu}(t^d)$.
21: end for
22: for $\forall s \in \widetilde{\mathcal{S}}$ do
23:     if $s \in \widetilde{\mu}(t^r)$ or $s \in \widetilde{\mu}(t^d)$ then
24:         $s \in \widetilde{\mu}(t)$.
25:     end if
26: end for
```

TABLE II
WORKERS AND TASKS

	s_1	s_2	s_3	s_4	s_5	s_6
$\succ$	t_1	t_2	t_2	t_2	t_2	t_2
	t_2	t_1	t_1	t_1	t_1	t_1
r_s	0.4	0.55	0.3	0.2	0.6	0.1

	t_1	t_2
q_t	1	1.1
b_t	1.9	1.3

Round 1: $s_1 \to t^r_1$ (worker s_1 proposes to task t^r_1). Task t^r_1 has enough budget, worker s_1 is assigned to task t^r_1, and we have $\mu(s_1) = t^r_1$ and $\mu(t^r_1) = \{s_1\}$.

Round 2: $s_2 \to t^r_2$ (worker s_2 proposes to task t^r_2). Task t^r_2 has enough budget, worker s_2 is assigned to task t^r_2, and we have $\mu(s_2) = t^r_2$ and $\mu(t^r_2) = \{s_2\}$.

Round 3: $s_3 \to t^r_2$ (worker s_3 proposes to task t^r_2). Task t^r_2 has enough budget, worker s_3 is assigned to task t^r_2, and we have $\mu(s_3) = t^r_2$ and $\mu(t^r_2) = \{s_2, s_3\}$.

Round 4: $s_4 \to t^r_2$ (worker s_4 proposes to task t^r_2). Task t^r_2 has enough budget, worker s_4 is assigned to task t^r_2, and we have $\mu(s_4) = t^r_2$ and $\mu(t^r_2) = \{s_2, s_3, s_4\}$.

Algorithm 3 ReDA: Revised Deferred Acceptance Algorithm

```
Input: Task $t$, worker $s$, temporary task assignment $\mu$.
Output: Updated task assignment $\mu$
1:  if $r_s \leq b_t - \sum_{s' \in \mu(t)} r_{s'}$ then
2:      Assign worker $s$ to task $t$: $\mu(t) = \mu(t) \bigcup \{s\}$, $\mu(s) = t$.
3:  else
4:      $\mathcal{A} = \{s' | s' \in \mu(t), s' \prec_t s\}$.
5:      if $\exists \mathcal{B} \subseteq \mathcal{A}, Q_{\mathcal{B}} < r_s$ and $r_s \leq b_t - Q_{\mu(t) \setminus \mathcal{B}}$ then
6:          Find such $\mathcal{B}$ with the minimum total quality level.
7:          for $\forall s' \in \mathcal{B}_{min}$ do
8:              $\mu(s') = \emptyset$.
9:          end for
10:         Assign worker $s$ to task $t$: $\mu(t) = \mu(t) \bigcup \{s\} \setminus \mathcal{B}_{min}$, $\mu(s) = t$.
11:     else
12:         $\mu(t) = \mu(t), u(s) = \emptyset$.
13:     end if
14: end if
```

TABLE III
TRANSFORMED WORKERS AND TASKS. (a) ORIGINAL TASKS. (b) TRANSFORMED TASKS. (c) ORIGINAL WORKERS. (d) TRANSFORMED WORKERS

(a)

	$\succ t_1$	$\succ t_2$
	$s_5 : 0.6$	$s_5 : 0.6$
	$s_2 : 0.55$	$s_2 : 0.55$
	$s_1 : 0.4$	$s_1 : 0.4$
	$s_3 : 0.3$	$s_3 : 0.3$
	$s_4 : 0.2$	$s_4 : 0.2$
	$s_6 : 0.1$	$s_6 : 0.1$
q_t	1	1.1
b_t	1.9	1.3

(b)

	$\tilde{\succ} t_1^r$	$\tilde{\succ} t_1^d$	$\tilde{\succ} t_2^r$	$\tilde{\succ} t_2^d$
	$s_5 : 0.6$	$s_5 : 0.6$	$s_5 : 0.6$	$s_5 : 0.6$
	$s_2 : 0.55$	$s_2 : 0.55$	$s_2 : 0.55$	$s_2 : 0.55$
	$s_1 : 0.4$	$s_1 : 0.4$	$s_1 : 0.4$	$s_1 : 0.4$
	$s_3 : 0.3$	$s_3 : 0.3$	$s_3 : 0.3$	$s_3 : 0.3$
	$s_4 : 0.2$	$s_4 : 0.2$	$s_4 : 0.2$	$s_4 : 0.2$
	$s_6 : 0.1$	$s_6 : 0.1$	$s_6 : 0.1$	$s_6 : 0.1$
b_t	1	0.9	1.1	0.2

(c)

	$\succ s_1$	$\succ s_2$	$\succ s_3$	$\succ s_4$	$\succ s_5$	$\succ s_6$
	t_1	t_2	t_2	t_2	t_2	t_2
	t_2	t_1	t_1	t_1	t_1	t_1

(d)

	$\tilde{\succ} s_1$	$\tilde{\succ} s_2$	$\tilde{\succ} s_3$	$\tilde{\succ} s_4$	$\tilde{\succ} s_5$	$\tilde{\succ} s_6$
	t_1^r	t_2^r	t_2^r	t_2^r	t_2^r	t_2^r
	t_1^d	t_2^d	t_2^d	t_2^d	t_2^d	t_2^d
	t_2^r	t_1^r	t_1^r	t_1^r	t_1^r	t_1^r
	t_2^d	t_1^d	t_1^d	t_1^d	t_1^d	t_1^d

Round 5: $s_5 \rightarrow t_2^r$ (worker s_5 proposes to task t_2^r). Task t_2^r does not have enough budget for s_5. The set of workers who are less preferred than worker s_5 is $\mathcal{A} = \{s_2, s_3, s_4\}$. The potential subsets of workers in $\mathcal{A}$ whose total quality is less than r_5 and who can vacate enough budget for s_5 include $\{s_2\}$ and $\{s_3, s_4\}$. We choose to remove set $\mathcal{B}_{\min} = \{s_2\}$ with the optimal total quality. Therefore, we have $\mu(s_5) = t_2^r$ and $\mu(t_2^r) = \{s_3, s_4, s_5\}$.

Round 6: $s_6 \rightarrow t_2^r$ (worker s_6 proposes to task t_2^r). Task t_2^r does not have enough budget for s_6, and the set of workers who are less preferred than worker s_6 is $\mathcal{A} = \emptyset$. Therefore, worker s_6 is rejected by task t_2^r.

Round 7: $s_2 \rightarrow t_2^d$ (worker s_2 proposes to task t_2^d). Worker s_2 cannot be assigned to task t_2^d because doing so will make it impossible to satisfy the quality requirement of the original task t_1. Hence, worker s_2 is rejected by task t_2^d.

Round 8: $s_2 \rightarrow t_1^r$ (worker s_2 proposes to task t_1^r). Task t_1^r has enough budget, worker s_2 is assigned to task t_1^r, and we have $\mu(s_2) = t_1^r$ and $\mu(t_1^r) = \{s_1, s_2\}$.

Round 9: $s_6 \rightarrow t_2^d$ (worker s_6 proposes to task t_2^d). Worker s_6 cannot be assigned to task t_2^d because doing so will make it impossible to satisfy the quality requirement of the original task t_1. Hence, worker s_6 is rejected by task t_2^d.

Round 10: $s_6 \rightarrow t_1^r$ (worker s_6 proposes to task t_1^r). Task t_1^r does not have enough budget for s_6, and the set of workers who are less preferred than worker s_6 is $\mathcal{A} = \emptyset$. Therefore, worker s_6 is rejected by task t_1^r.

Round 11: $s_6 \rightarrow t_1^d$ (worker s_6 proposes to task t_1^d). The quality requirement of the original task t_2 (the other task in the system) is fulfilled, and task t_1^d has enough budget for worker s_6. Thus, worker s_6 is assigned to task t_1^d, and we have $\mu(s_6) = t_1^d$ and $\mu(t_1^d) = \{s_6\}$.

We summarize the stable task assignment process in Table IV. The final task assignment is $\mu(t_1) = \{s_1, s_2, s_6\}$ and $\mu(t_2) = \{s_3, s_4, s_5\}$. It can be easily checked that this task assignment satisfies budget constraints and quality requirements of all tasks, achieving a 100% success ratio.

C. Theoretical Analysis

Theorem 1 (Computational Complexity): The ESTA algorithm converges with a computational complexity of $O(|\mathcal{S}||\mathcal{T}|\tau)$, in which τ is the computational complexity of finding $\mathcal{B}_{\min}$ in Algorithm 3.

Proof: To transform the task assignment as in Algorithm 1 that has a computational complexity of $O(|\mathcal{S}|+|\mathcal{T}|)$. As shown in Algorithm 2, for regular tasks, the computation complexity is $O(|\mathcal{S}||\mathcal{T}|\tau)$, and for shadow tasks, the computation complexity is also $O(|\mathcal{S}||\mathcal{T}|\tau)$. In summary, the ESTA algorithm has a computational complexity of $O(|\mathcal{S}||\mathcal{T}|\tau)$. ■

Theorem 2 (Individual Rationality): The matching result of the ESTA algorithm is individually rational.

Proof: Every worker prefers the current task assignment to being unassigned. Both regular tasks and shadow tasks prefer a larger set of workers as it improves the total quality level, thus every crowdsourcer prefers the current set of assigned workers to any subset of these workers as long as its budget permits. Therefore, the ESTA algorithm is individually rational. ■

Lemma 1: The task assignment algorithm in Algorithm 2 guarantees that through the iteration, the total quality of workers assigned to a task, i.e., $Q_{\mu(t)}$, is nondecreasing.

We ignore the proof of Lemma 1. Interested readers refer to the proof in the conference version [11].

Theorem 3 (Nonwastefulness): The matching result of the ESTA algorithm is nonwasteful.

TABLE IV
TASK ASSIGNMENT PROCESS OF THE ESTA ALGORITHM

Round	Action	Available budget $b'_{t_1^r}; b'_{t_1^d}; b'_{t_2^r}; b'_{t_2^d}.$	Aggregate quality $q'_{t_1^r}; q'_{t_1^d}; q'_{t_2^r}; q'_{t_2^d}.$	Matching result $\mu(t_1^r); \mu(t_1^d); \mu(t_2^r); \mu(t_2^d).$
1	$s_1 \to t_1^r$	0.6; 0.9; 1.1; 0.2.	0.4; 0; 0; 0.	$\{s_1\}; \emptyset; \emptyset; \emptyset.$
2	$s_2 \to t_2^r$	0.6; 0.9; 0.55; 0.2.	0.4; 0; 0.55; 0.	$\{s_1\}; \emptyset; \{s_2\}; \emptyset.$
3	$s_3 \to t_2^r$	0.6; 0.9; 0.25; 0.2.	0.4; 0; 0.85; 0.	$\{s_1\}; \emptyset; \{s_2, s_3\}; \emptyset.$
4	$s_4 \to t_2^r$	0.6; 0.9; 0.05; 0.2.	0.4; 0; 1.05; 0.	$\{s_1\}; \emptyset; \{s_2, s_3, s_4\}; \emptyset.$
5	$s_5 \to t_2^r$	0.6; 0.9; 0; 0.2.	0.4; 0; 1.1; 0.	$\{s_1\}; \emptyset; \{s_3, s_4, s_5\}; \emptyset.$
6	$s_6 \to t_2^r$	0.6; 0.9; 0; 0.2.	0.4; 0; 1.1; 0.	$\{s_1\}; \emptyset; \{s_3, s_4, s_5\}; \emptyset.$
7	$s_2 \to t_2^d$	0.6; 0.9; 0; 0.2.	0.4; 0; 1.1; 0.	$\{s_1\}; \emptyset; \{s_3, s_4, s_5\}; \emptyset.$
8	$s_2 \to t_1^r$	0.05; 0.9; 0; 0.2.	0.95; 0; 1.1; 0.	$\{s_1, s_2\}; \emptyset; \{s_3, s_4, s_5\}; \emptyset.$
9	$s_6 \to t_2^d$	0.05; 0.9; 0; 0.2.	0.95; 0; 1.1; 0.	$\{s_1, s_2\}; \emptyset; \{s_3, s_4, s_5\}; \emptyset.$
10	$s_6 \to t_1^r$	0.05; 0.9; 0; 0.2.	0.95; 0; 1.1; 0.	$\{s_1, s_2\}; \emptyset; \{s_3, s_4, s_5\}; \emptyset.$
11	$s_6 \to t_1^d$	0.05; 0.8; 0; 0.2.	0.95; 0.1; 1.1; 0.	$\{s_1, s_2\}; \{s_6\}; \{s_3, s_4, s_5\}; \emptyset.$

Proof: We prove the nonwastefulness of the ESTA algorithm by contradiction. Assume that under the final task assignment μ, there is a type II blocking pair (s, t). Since $t \succ_s \mu(s)$, worker s must have proposed to corresponding task t^r and t^d, but has been rejected. Let μ' denote the task assignment at the time worker s proposes to task t^r or t^d. One of the following two situations must be true.

1) *s Proposes to t^r:* Task t^r does not have enough budget for worker s because otherwise, worker s will be assigned to task t^r immediately, we have $Q_{\mu'(t^r)} + r_s > b_{t^r}$. According to Lemma 1, the aggregate quality of task t^r is nondecreasing, i.e., $Q_{\mu(t^r)} \geq Q_{\mu'(t^r)}$, where $\mu(t^r)$ is the final assignment result of task t^r and $Q_{\mu(t^r)}$ is the aggregate quality of workers assigned to task t^r. Therefore, under the final task assignment, we have $Q_{\mu(t^r)} + r_s > b_{t^r}$, and worker s and task t^r cannot form a type II blocking pair.
2) *s proposes to t^d:*
   a) Task t^d has enough budget for worker s, but without worker s, the remaining unassigned workers cannot satisfy the quality requirements of other tasks. In this case, in the final task assignment, worker s cannot shift to task t^d because the quality requirement of task $\mu'(s)$ will be violated.
   b) Task t^d does not have enough budget for worker s, and the remaining unassigned workers can satisfy the quality requirements of other tasks, i.e., $Q_{\mu'(t^d)} + r_s > b_{t^d}$. According to Lemma 1, the aggregate quality of task t^d is nondecreasing, i.e., $Q_{\mu(t^d)} \geq Q_{\mu'(t^d)}$. Therefore, under the final task assignment, we have $Q_{\mu(t^d)} + r_s > b_{t^d}$, and worker s and task t^d cannot form a type II blocking pair.
   c) Task t^d does not have enough budget for worker s, and without worker s, the remaining unassigned workers cannot satisfy the quality requirement of other tasks. In this case, in the final assignment, worker s cannot shift to task t^d because the quality requirement of task $\mu'(s)$ will be violated.

In summary, worker s and task t cannot form a type II blocking pair. Therefore, the matching result of the ESTA algorithm is nonwasteful. ■

To prove that the ESTA algorithm can achieve a fair task assignment result, we first introduce the following lemma.

Lemma 2: If worker s is rejected by shadow task t^d, under the final task assignment $\widetilde{\mu}$, it must be true that $\forall s' \in \widetilde{\mu}(t^d)$, we have $s' \succsim_{t^d} s$.

Proof: Assume that worker s is rejected by shadow task t^d at round w, and $\widetilde{\mu}_w(t^d)$ is the (temporary) matching result at round w. There are three situations in which worker s will be rejected by shadow task t^d.

1) $\sum_{s' \in \widetilde{\mu}_w(t^d)} r_{s'} < b_{t^d}$, but without worker s, the remaining unassigned workers cannot satisfy the quality requirements of other tasks.
2) $\sum_{s' \in \widetilde{\mu}_w(t^d)} r_{s'} < b_{t^d}$, and the quality requirements of other tasks have been fulfilled, but shadow task t^d does not have enough budget for worker s, and a subset of workers who are less preferred than worker s cannot be found at the same time.
3) $\sum_{s' \in \widetilde{\mu}_w(t^d)} r_{s'} = b_{t^d}$, and a subset of workers who are less preferred than worker s cannot be found. Therefore, if worker s is rejected by shadow task t^d at round w, it means that $\forall s' \in \widetilde{\mu}_w(t^d)$, $s' \succsim_{t^d} s$.

For round $w + 1$, the following situations may occur.

1) If worker s is rejected to reserve workers for other regular tasks, the $(|\widetilde{\mu}_w(t^d)| + 1)$th worker is also rejected by t^d. For example, as shown in the toy example, s_2 is rejected by shadow task t_2^d at round 7, and s_6 is also rejected by shadow task t_2^d at round 9. Therefore, we have $\widetilde{\mu}_{w+1}(t^d) = \widetilde{\mu}_w(t^d)$, and $\forall s'' \in \widetilde{\mu}_{w+1}(t^d)$, $s'' \succsim_{t^d} s$.
2) If worker s is rejected because the set of workers who are less preferred than worker s does not exist, according to Lemma 1, since the aggregate quality of workers allocated to a task is nondecreasing, we know that either the $(|\widetilde{\mu}_w(t^d)| + 1)$th worker has higher quality level than worker s or the $(|\widetilde{\mu}_w(t^d)| + 1)$th worker can fill the remaining budget for task t^d, otherwise, the $(|\widetilde{\mu}_w(t^d)| + 1)$th worker will also be rejected. Thus $\forall s'' \in \widetilde{\mu}_{w+1}(t^d)$, $s'' \succsim_{t^d} s$. Therefore, in the final matching result, we can derive that $\forall s' \in \widetilde{\mu}(t^d)$, $s' \succsim_{t^d} s$. ■

Theorem 4 (Fairness): The matching result of the ESTA algorithm is fair.

Proof: Assume that under the final task assignment $\widetilde{\mu}$, there is a type I blocking pair (s, t). This implies that worker s has applied for but been rejected by both t^r and t^d. Because, otherwise, worker s would have applied for t^r and t^d according to lines 2–17 in Algorithm 2. Consider any worker $s' \in \widetilde{\mu}(t^r)$,

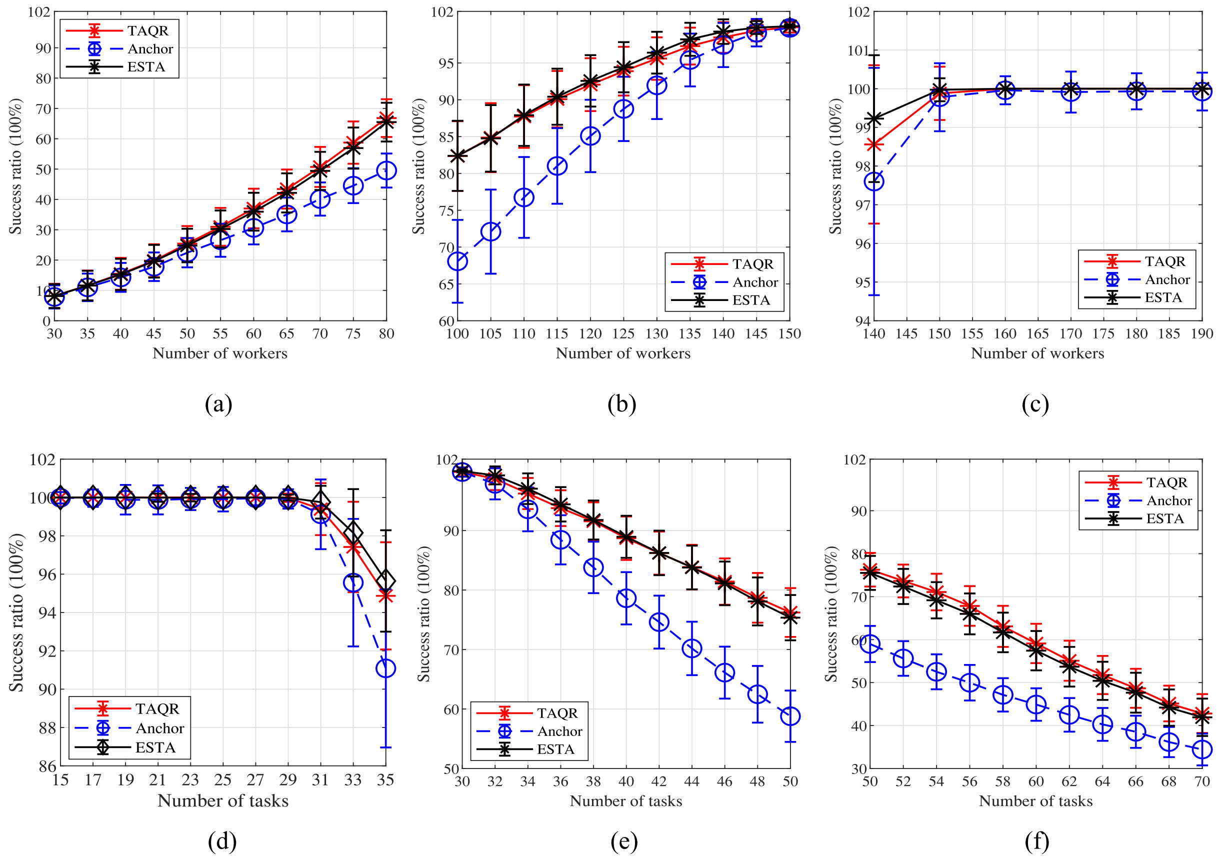

Fig. 2. Success ratio. (a) Limited workers, $jTj = 30$. (b) Balanced workers, $jTj = 30$. (c) Abundant workers, $jTj = 30$. (d) Limited tasks, $jSj = 150$. (e) Balanced tasks, $jSj = 150$. (f) Overloaded tasks, $jSj = 150$.

it must be true that $s' \succsim_{t^r} s$ because s is rejected in favor of the currently matched workers who are more preferred than s. Consider any worker $s'' \in \widetilde{\mu}(t^d)$, according to Lemma 2, we have $s'' \succsim_{t^d} s$. This verifies that no type I blocking pair will exist. Therefore, the matching result of the ESTA algorithm is fair. ■

Theorem 5 (Stability): The ESTA algorithm produces a stable task assignment.

Proof: According to Theorems 2–4, the final task assignment is individually rational, fair, and nonwasteful. Therefore, the ESTA task assignment algorithm is stable. ■

IV. Simulation

In this section, we evaluate the performance of the proposed algorithms, referred to as the TAQR algorithm (the conference version [11]) and the ESTA algorithm with a guaranteed requirement, respectively. We implement Anchor [13], an algorithm for many-to-one matching with upper bounds and heterogeneous agents but no lower bounds, as the benchmark for comparison. The key idea of Anchor is that whenever a task rejects a worker, it also rejects any other workers that are less preferred than this worker, even if the budget allows for these workers. In this way, Anchor achieves fairness but not nonwastefulness. We compare our proposed matching algorithms TAQR and ESTA with Anchor in terms of success ratio, worker happiness, task happiness, and running time. Each simulation runs for 500 times on a ThinkPad laptop with Intel Core i5-3230M CPU at 2.60 GHz and 4.00-GB RAM.

A. Success Ratio

Recall that we define success ratio as the ratio of successfully completed tasks to all tasks in Definition 1. A task can be successfully accomplished only if its quality requirement is satisfied. We compare the performance of our proposed algorithms with the baseline algorithm in terms of success ratio.

1) Success Ratio Versus the Number of Workers: In this set of simulations, the quality requirements $q_t \ \forall t \in \mathcal{T}$, budget constraints $b_t \ \forall t \in \mathcal{T}$, and worker quality levels $r_s \ \forall s \in \mathcal{S}$ are randomly chosen from [3, 5], [6, 10], and [1, 2], respectively. Naturally, the success ratio increases with the number of workers with a fixed number of tasks, as shown in Fig. 2(a)–(c). When there is a limited number of workers available in the system, the success ratio is very low but will increase rapidly with the increase of workers, as shown in Fig. 2(a). When the number of workers and the number of tasks are relatively balanced, the success ratio has a quasilinear relationship with the number of workers, as shown in Fig. 2(b). Compared with the benchmark, our proposed algorithms can achieve up to 16% improvement. With massive workers, the gap between the proposed algorithms and the benchmark narrows down, as shown in Fig. 2(c). The success ratio will reach 100% when the number of workers is large enough for all tasks to fill

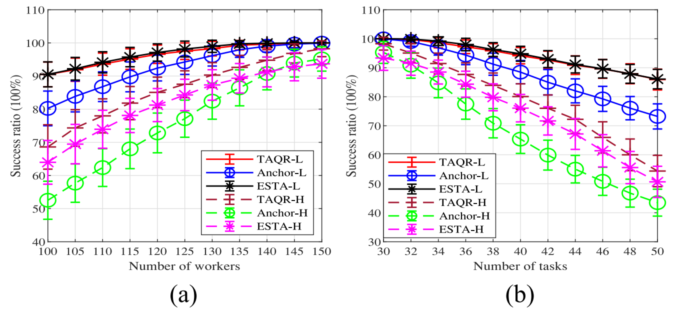

Fig. 3. Success ratio under different quality requirements. (a) $|\mathcal{T}| = 30$. (b) $|\mathcal{S}| = 150$.

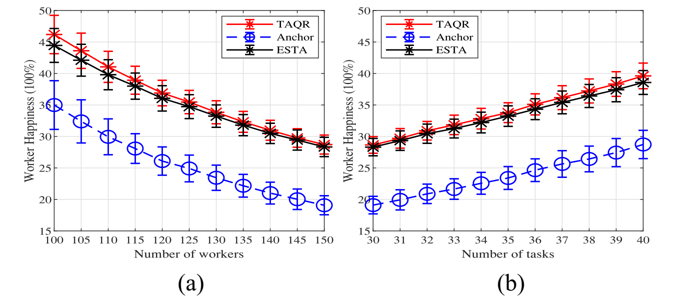

Fig. 4. Happiness comparison. (a) $|\mathcal{T}| = 30$. (b) $|\mathcal{S}| = 150$.

up their quality requirements. It is shown that our proposed algorithms can deal with worker shortages more effectively since we prioritize tasks with unsatisfied quality requirements during the assignment, and make the best use of the budget to hire as many workers as possible, leading to a higher success ratio than the benchmark.

2) Success Ratio Versus the Number of Tasks: In this set of simulations, the quality requirements $q_t \quad \forall t \in \mathcal{T}$, budget constraints $b_t \quad \forall t \in \mathcal{T}$, and worker quality levels $r_s \quad \forall s \in \mathcal{S}$ are randomly chosen from [3, 5], [6, 10], and [1, 2], respectively. The success ratio will decrease as more tasks compete for a fixed number of workers, as shown in Fig. 2(d)–(f). The success ratio stays at 100% when there is a small number of tasks and drops as the number of tasks increases, as shown in Fig. 2(d). Our proposed algorithms have much slower decrease rates than the benchmark, and can achieve up to 18% gain in success ratio, as shown in Fig. 2(e). As the number of tasks further increases, the gap between the proposed algorithms and the benchmark first grows then decreases, as shown in Fig. 2(f). The benchmark suffers from worker deficiency more severely as it aggressively rejects workers to ensure fairness without considering fulfilling the quality requirements of tasks. Both TAQR and ESTA have a higher success ratio than the benchmark under the same conditions. Thanks to task partition, ESTA is better at recruiting low-quality workers than TAQR, and can make full use of the budgets of tasks.

3) Success Ratio Versus Quality Requirements: In this set of simulations, the budget constraints $b_t \quad \forall t \in \mathcal{T}$ and worker quality levels $r_s \quad \forall s \in \mathcal{S}$ are randomly chosen from [6, 10] and [1, 2], respectively. To investigate the influence of quality requirements on the success ratio, we draw the low-quality requirements randomly from [2, 4], corresponding to "-*L*" in Fig. 3, and the high-quality requirements randomly from

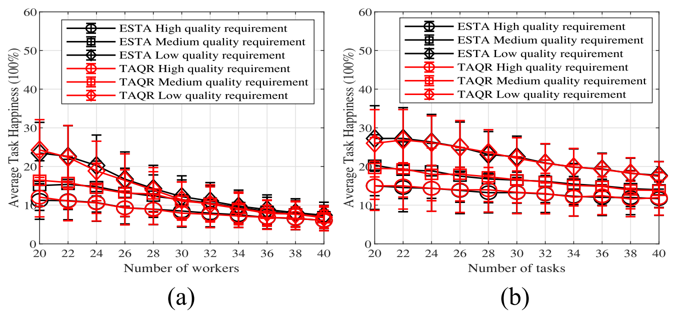

Fig. 5. Happiness versus quality requirements. (a) $|\mathcal{T}| = 125$. (b) $|\mathcal{S}| = 30$.

[4, 6], corresponding to "-*H*" in Fig. 3. It is shown that our proposed algorithms outperform the benchmark. It is noteworthy that higher quality requirements lead to lower success ratio with a fixed number of tasks, as shown in Fig. 3(a). With a fixed number of workers, the success ratio is lower if the quality requirements are higher, as shown in Fig. 3(b). With higher quality requirements, workers with moderate quality levels may not be recruited due to task partition, thus TAQR has a slightly higher success ratio than ESTA.

B. Happiness of Workers and Tasks

We have the same definition of happiness as [13]. The happiness of a worker is the rank percentile of her assigned task. For example, if there are three tasks, a worker's happiness is 100% if she is assigned to her most-preferred task; her happiness is 33% if she is assigned to her least-preferred task; and her happiness is 0% if she is unassigned. The happiness of a task is the average rank percentile of its assigned workers. For example, if there are four workers, and a task is assigned to its most-preferred worker and the third-preferred worker, its happiness will be $(100 + 50)/2 \times 100\% = 75\%$. In this set of simulations, the quality requirements $q_t \quad \forall t \in \mathcal{T}$, budget constraints $b_t \quad \forall t \in \mathcal{T}$, and worker quality levels $r_s \quad \forall s \in \mathcal{S}$ are randomly chosen from [3, 5], [6, 10], and [1, 4], respectively.

As shown in Fig. 4(a), worker happiness goes down as more workers join the crowdsourcing platform, making it more and more difficult for an individual worker to be assigned to her preferred tasks. TAQR and ESTA maintain 11% and 12% gain, respectively, over the benchmark because they do not reject workers as radically as the benchmark and make better use of their budgets to employ more workers. The worker's happiness grows as there are more tasks since there is a higher chance for a worker to be assigned to her preferred tasks, as shown in Fig. 4(b). The worker happiness of TAQR and ESTA is around 11% and 12% higher than that of the benchmark, respectively.

C. Impact of Quality Requirement

Lower bounds, i.e., the quality requirements, are not considered in the benchmark, thus we only show the impact of quality requirement on TAQR and ESTA, but not the benchmark. In this set of simulations, the budget constraints $b_t \quad \forall t \in \mathcal{T}$ and worker quality levels $r_s \quad \forall s \in \mathcal{S}$ are randomly chosen from [6, 10] and [1, 4], respectively. We study the scenarios when tasks have high-, medium-, and low-quality requirements, with

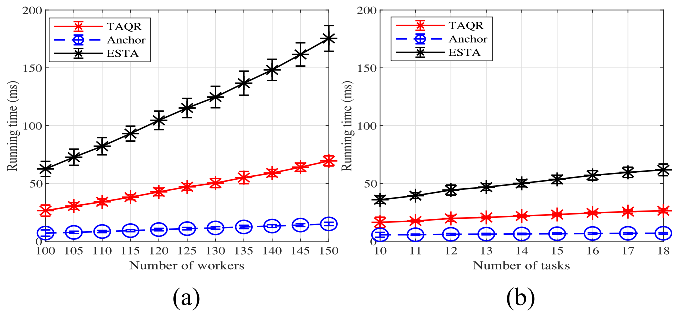

Fig. 6. Running time comparison. (a) $|\mathcal{T}| = 30$. (b) $|\mathcal{S}| = 100$.

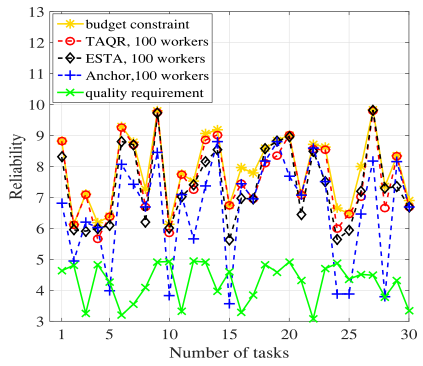

Fig. 7. Matching result of the aggregate quality level.

$q_t \quad \forall t \in \mathcal{T}$ randomly chosen from [3, 4], [2, 3], and [1, 2], respectively. As illustrated in Fig. 5, the average task happiness is higher when the quality requirements are lower, as it is unnecessary to recruit less-preferred workers to meet the quality requirements. This also explains the interesting observation in Fig. 5(a) that the average task happiness decreases with the number of workers. This is because more less-preferred workers are assigned to each task, dragging down the average task happiness. When the number of tasks goes up, the average task happiness gradually descends due to a lack of workers, as shown in Fig. 5(b).

D. Running Time

In this set of simulations, the quality requirements $q_t \quad \forall t \in \mathcal{T}$, budget constraints $b_t \quad \forall t \in \mathcal{T}$, and worker quality levels $r_s \quad \forall s \in \mathcal{S}$ are randomly chosen from [0.8, 1.5], [1.4, 2], and [0, 1], respectively. As shown in Fig. 6, the running time of TAQR and ESTA is linear in the number of workers and the number of tasks, and is longer than that of the benchmark. Taking into account the results in Figs. 2 and 4, it is clear that there is a tradeoff between the running time and performance improvement. The benchmark rapidly culls out less-preferred workers to ensure fairness but not the quality requirements of tasks. In comparison, both TAQR and ESTA carefully examine every worker for potential assignment and make better use of the budget to hire as many workers as possible, resulting in higher success ratio and worker happiness at the expense of longer running time. Due to task partition and problem transformation, the running time of ESTA is relatively higher than that of TAQR.

E. Aggregate Quality-Level Comparison

By taking a closer look at the task assignment result, we demonstrate that both TAQR and ESTA make better use of the budget and improve the success ratio of tasks. We consider 30 tasks and 100 workers, with the quality requirements $q_t \quad \forall t \in \mathcal{T}$, budget constraints $b_t \quad \forall t \in \mathcal{T}$, and worker quality levels $r_s \quad \forall s \in \mathcal{S}$ randomly chosen from [3, 5], [6, 10], and [1, 4], respectively. We run the simulation once, and show the aggregate quality of assigned workers to each task in Fig. 7. It is clear that both TAQR and ESTA meet the quality requirement of every task and try to use available budgets to attain the highest possible quality. The benchmark fails to reach the quality requirements of several tasks and leaves a considerable amount of room for quality improvement. This further corroborates the fact that both TAQR and ESTA can improve the success ratio (Fig. 2) and worker happiness (Fig. 4).

V. Related Work

Task Assignment for Crowdsourcing: Crowdsourcing has become a promising and popular paradigm for collecting and sharing information [1], and task assignment is one of the fundamental concerns in crowdsourcing platforms. In [2], workers were assigned to tasks in a way that maximizes the total benefit of the crowdsourcing platform by using a two-phase exploration–exploitation assignment algorithm. The exploration phase assessed worker skills while the exploitation phase performed the task assignment for profit maximization. Boutsis and Kalogeraki [3] presented CRITICAL to determine the appropriate group of workers to every task under reliability and time constraints. Feng *et al.* [4] described a truthful auction mechanism with an optimal task allocation algorithm and near-optimal truthful mechanism for offline and online task assignment, respectively. In [5], taking both spatial coverage and temporal coverage into account, He *et al.* proposed a greedy approximation and a genetic algorithm to achieve high-quality crowdsourcing results by recruiting workers who best match the application requirements, based on their predicted mobility. Xiao *et al.* [6] proposed an offline task assignment algorithm and an online task assignment algorithm based on a greedy strategy. Karaliopoulos *et al.* [7] devised a greedy heuristics worker selection algorithm to deal with the uncertainty of worker mobility. Han *et al.* [14] presented an online algorithm to achieve robust crowdsensing. In [15], by taking geographical characteristics of sensing tasks and the spatial movement constraints of mobile workers into account, the problem of allocating location-dependent tasks was studied. Cheung *et al.* [16] designed a distributed task selection algorithm to collect time-sensitive and location-dependent data from heterogeneous workers. In [17], based on the service quality factor (SQF), link reliability factor (LRF), and region heat factor (RHF), An *et al.* evaluated preferences of users, put forward a credible crowdsourcing assignment model, and proposed a crowdsourcing algorithm that solved the credible interaction issue between mobile users finally. In [18] and [19], network effects have been leveraged to incentivize workers to undertake crowdsourcing tasks. Jeong *et al.* [20] leveraged carrier resources, distributing CPU resources as well as user demands, and proposed a hierarchical trust computing algorithm to assign task efficiently based on the Vickrey–Clarke–Groves auction. Yu *et al.* [8] introduced a temporal–spatial task allocation algorithm to match the tasks, fog nodes (FNs), and workers, which achieves efficient resource allocation and system robustness. Tao *et al.* [9] proposed a genetic algorithm and a detective algorithm to maximize data quality and the profit of workers. In [10], to meet the requirements of delivery delay and task assignment, Ni *et al.* proposed a utility-based maximization solution for the roadside unit deployment problem. Bhatti *et al.* [21] developed a constant-ratio approximation solution for bounded and heterogeneous task assignment to maximize the sum of the rewards of workers. Sarker *et al.* [22] introduced a task allocation policy by taking worker utilities and platform profit into consideration.

Existing works mostly focused on utility maximization for the entire crowdsourcing platform, ignoring personal preferences of individual workers and crowdsourcers. An optimal but unstable task assignment may not be easily implemented, as unsatisfactory workers and crowdsourcers have incentives to deviate from the assignment result. By formulating the task assignment in crowdsourcing as a matching problem and generating a stable result, we ensure that every participant is willing to abide by the task assignment result.

Stable Matching: Stable matching has been studied extensively since 1962. Gale and Shapley [23] first proposed and then analyzed the problems of stable matching. According to [24], the deferred acceptance algorithm was used to achieve a stable matching. Variants of matching problems in economics have been examined [25]–[28]. Hamada *et al.* [29] focused on the hospital-residents matching problem with quota, and proposed an exponential-time algorithm. The theory of matching has also been explored in computer science. Fragiadakis *et al.* [12] introduced two classes of strategy-proof mechanisms for many-to-one matching with minimum quotas. Zhang *et al.* [30] investigated the resource allocation problem between a set of data service subscribers (DSSs) and a set of low-power FNs. Xu and Li [13] proposed both online and offline algorithms to match virtual machines to heterogeneous sized jobs in the cloud. In [31], two generalized stable matching problems that exist in the higher education sector with lower and common quotas were studied. In [32] and [33], the spectrum resources from sellers were matched with the requirements of buyers. In our previous work [34], we proposed a stable worker–task matching framework for crowdsourcing but did not consider the quality requirements of tasks. However, none of these matching frameworks can be applied to TAQRs and budget constraints, where the heterogeneity in worker quality levels makes it challenging to reach a stable matching result.

VI. Conclusion

In this article, we investigated the task assignment problem for crowdsourcing platforms. Instead of finding a task assignment that can maximize the total utility of the crowdsourcing platform, we focused on the diverse preferences of individual workers and crowdsourcers toward each other and introduced a many-to-one matching framework with lower and upper bounds to account for the quality requirements and budget constraints of crowdsourcing tasks. To conquer the difficulty of heterogeneous worker skill levels, we proposed two stable matching algorithms, which can yield task assignment results that are individually rational, fair, and nonwasteful. Through extensive simulation results, we verified that our proposed algorithms can improve both the task success ratio and the happiness of both workers and tasks with acceptable time complexity.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[2] C. J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 45–51.

[3] I. Boutsis and V. Kalogeraki, "On task assignment for real-time reliable crowdsourcing," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2014, pp. 1–10. .

[4] Z. Feng *et al.*, "Towards truthful mechanisms for mobile crowdsourcing with dynamic smartphones," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2014, pp. 483–492.

[5] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 2542–2550.

[6] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu, "Multi-task assignment for crowdsensing in mobile social networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 1–9.

[7] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 2254–2262.

[8] Y. Yu, F. Li, S. Liu, J. Huang, and L. Guo, "Reliable fog-based crowdsourcing: A temporal-spatial task allocation approach," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3968–3976, May 2020.

[9] X. Tao and W. Song, "Location-dependent task allocation for mobile crowdsensing with clustering effect," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1029–1045, Feb. 2019.

[10] Y. Ni, J. He, L. Cai, J. Pan, and Y. Bo, "Joint roadside unit deployment and service task assignment for Internet of Vehicles (IoV)," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3271–3283, Apr. 2019.

[11] X. Yin, Y. Chen, and B. Li, "Task assignment with guaranteed quality for crowdsourcing platforms," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Service*, 2017, pp. 1–10.

[12] D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof matching with minimum quotas," *ACM Trans. Econ. Comput.*, vol. 4, no. 1, p. 6, 2016.

[13] H. Xu and B. Li, "Anchor: A versatile and efficient framework for resource management in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1066–1076, Jun. 2013.

[14] K. Han, C. Zhang, and J. Luo, "BLISS: Budget limited robust crowdsensing through online learning," in *Proc. IEEE Int. Conf. Sens. Commun. Netw. (SECON)*, 2014, pp. 555–563.

[15] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2014, pp. 745–753.

[16] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 157–166.

[17] J. An, X. Gui, Z. Wang, J. Yang, and X. He, "A crowdsourcing assignment model based on mobile crowd sensing in the Internet of Things," *IEEE Internet Things J.*, vol. 2, no. 5, pp. 358–369, Oct. 2015.

[18] Y. Chen, X. Wang, B. Li, and Q. Zhang, "An incentive mechanism for crowdsourcing systems with network effects," *ACM Trans. Internet Technol.*, vol. 19, no. 4, pp. 1–21, 2019.

[19] Y. Chen, B. Li, and Q. Zhang, "Incentivizing crowdsourcing systems with network effects," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[20] S. Jeong, W. Na, J. Kim, and S. Cho, "Internet of Things for smart manufacturing system: Trust issues in resource allocation," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4418–4427, Dec. 2018.

[21] S. S. Bhatti, J. Fan, K. Wang, X. Gao, F. Wu, and G. Chen, "An approximation algorithm for bounded task assignment problem in spatial crowdsourcing," *IEEE Trans. Mobile Comput.*, early access, Apr. 2, 2020, doi: 10.1109/TMC.2020.2984380.

[22] S. Sarker, M. A. Razzaque, M. M. Hassan, A. Almogren, G. Fortino, and M. Zhou, "Optimal selection of crowdsourcing workers balancing their utilities and platform profit," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8602–8614, Oct. 2019.

[23] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Month.*, vol. 69, no. 1, pp. 9–15, 1962.

[24] A. E. Roth, "Deferred acceptance algorithms: History, theory, practice, and open questions," *Int. J. Game Theory*, vol. 36, nos. 3–4, pp. 537–569, 2008.

[25] M. Pycia, "Many-to-one matching without substitutability," Ind. Perform. Center, MIT, Cambridge, MA, USA, Working Paper, vol. 8, 2005, pp. 1–48.

[26] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, *Peer Effects and Stability in Matching Markets* (Lecture Notes in Computer Science), vol. 6982. Heidelberg, Germany: Springer, 2011, pp. 117–129.

[27] A. E. Roth, "The college admissions problem is not equivalent to the marriage problem," *J. Econ. Theory*, vol. 36, no. 2, pp. 277–288, 1985.

[28] L. Ehlers, I. E. Hafalir, M. B. Yenmez, and M. A. Yildirim, "School choice with controlled choice constraints: Hard bounds versus soft bounds," *J. Econ. Theory*, vol. 153, pp. 648–683, 2014.

[29] K. Hamada, K. Iwama, and S. Miyazaki, "The hospitals/residents problem with quota lower bounds," in *Proc. Eur. Symp. Algorithms*, 2011, pp. 180–191.

[30] H. Zhang, Y. Xiao, S. Bu, D. Niyato, R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1204–1215, Oct. 2017.

[31] P. Biró, T. Fleiner, R. W. Irving, and D. F. Manlove, "The college admissions problem with lower and common quotas," *Theor. Comput. Sci.*, vol. 411, no. 34, pp. 3136–3153, 2010.

[32] Y. Chen, Y. Xiong, Q. Wang, X. Yin, and B. Li, "Ensuring minimum spectrum requirement in matching-based spectrum allocation," *IEEE Trans. Mobile Comput.*, vol. 17, no. 9, pp. 2028–2040, Sep. 2018.

[33] Y. Chen, Y. Xiong, Q. Wang, X. Yin, and B. Li, "Stable matching for spectrum market with guaranteed minimum requirement," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2017, pp. 1–10.

[34] Y. Chen and X. Yin, "Stable job assignment for crowdsourcing," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 26–47.

Xiaoyan Yin (Member, IEEE) received the Ph.D. degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2010.

She is currently an Associate Professor with the School of Information Science and Technology, Northwest University, Xi'an. Her research interests include wireless networks, Internet of Things, network economy, and social networks.

Yanjiao Chen (Senior Member, IEEE) received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2010, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2015.

She is currently a Professor with Wuhan University, Wuhan, China. Her research interests include computer networks, wireless system security, and network economy.

Cheng Xu received the B.S. degree in electrical engineering from Anhui University, Hefei, China, in 2017, and the M.S. degree in electronic and communication engineering from Northwest University, Xi'an, China, in 2020.

He is currently an Engineer with Shanghai Pudong Development Bank, Hefei. His main research interests include Internet of Things, data transmission, and wireless networks.

Sijia Yu received the B.S. degree in software engineering from Northwest University, Xi'an, China.

Her research interests include data transmission and Internet of Things.

Baochun Li (Fellow, IEEE) received the B.Engr. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1995, and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 1997 and 2000, respectively.

Since 2000, he has been with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, where he is currently a Professor. He has coauthored more than 340 research papers, with a total of over 16 000 citations, an H-index of 74, and an i10-index of 222, according to Google Scholar Citations. His research interests include large-scale distributed systems, cloud computing, applications of network coding, peer-to-peer networks, and wireless networks.

Dr. Li was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the field of communications systems in 2000. He is a member of ACM.