Task Assignment with Guaranteed Quality for Crowdsourcing Platforms

Xiaoyan Yin[‡], Yanjiao Chen[†*], Baochun Li[§]
[‡]School of Information Science and Technology, Northwest University, Xi'an, China.
[†]State Key Lab of Software Engineering, Computer School of Wuhan University, China
[*] Corresponding author
[§]Department of Electrical and Computer Engineering, University of Toronto
Email: ‡SCxiaoyanyin@gmail.com, †chenyanjiao@whu.edu.cn, §bli@ece.toronto.edu

Abstract—Crowdsourcing leverages the collective intelligence of the massive crowd workers to accomplish tasks in a cost-effective way. On a crowdsourcing platform, it is challenging to assign tasks to workers in an appropriate way due to heterogeneity in both tasks and workers. In this paper, we explore the problem of assigning workers with various skill levels to tasks with different quality requirements and budget constraints. We first formulate the task assignment as a many-to-one matching problem, in which multiple workers are assigned to a task, and the task can be successfully completed only if a minimum quality requirement can be satisfied within its limited budget. Different from traditional task assignment mechanisms which focus on utility maximization for the crowdsourcing platform, our proposed matching framework takes into consideration the preferences of individual crowdsourcers and workers towards each other. We design a novel algorithm that can generate a stable outcome for the many-to-one matching problem with lower and upper bounds (i.e., quality requirement and budget constraint), as well as heterogeneous worker skill levels. Through extensive simulations, we show that the proposed algorithm can greatly improve the success ratio of task accomplishment and worker happiness, when compared with existing algorithms.

Index Terms—Task Assignment, Crowdsourcing, Quality Requirement.

1 Introduction

Crowdsourcing has provided an open, voluntary, and reliable environment for crowdsourcers and individuals to cooperate in achieving a cumulative result. On crowdsourcing platforms, individual workers can choose and contribute to tasks that are published by crowdsourcers. Combining efforts from numerous workers, many crowdsourcing tasks can be accomplished with high quality and low costs, such as the collection of environmental data and the design and evaluation of new products.

Task assignment is one of the most essential problems in crowdsourcing. There is an abundance of existing works [1], [2], [3], [4], [5], [6], [7] proposing task assignment algorithms in order to fulfill a certain design objective. In [2], a two-phase exploration-exploitation algorithm is presented to assign workers with different skill levels to tasks with limited budgets, aiming at maximizing the profit of the crowdsourcer. Geographical and temporal requirement of tasks and the mobility and active time period of workers have been considered in [3], [4], [5], [7] to ensure that tasks are finished within the deadline, and with enough coverage. The uncertainty of user mobility has been studied in [4], [7]. Most of the existing works focus on maximizing the total utility of the crowdsourcing platform, but disregard the preferences of individual crowdsourcers and workers towards each other. However, in an open crowdsourcing platform without tight enforcement, a task assignment will be unstable if there exists a pair of crowdsourcer and worker who prefer each other but are not assigned together.

To address this issue, in this paper, we formulate the problem of assigning tasks to workers as a many-to-one matching problem, taking into consideration the preference profiles of both workers and crowdsourcers. As shown in Fig. 1, workers are heterogeneous in their skill levels due to their diversity in experience, capabilities, and proficiency. Workers with higher skill levels will ask for a higher compensation, while workers with lower skill levels are cheaper to hire. Due to the nature of crowdsourcing, the more workers engaged in a task, the better results can be achieved. On the one hand, it is vital to recruit enough workers to reach a certain quality requirement, below which a task cannot be successfully completed. On the other hand, crowdsourcers are constrained by their budgets, thus cannot employ too many workers. This adds up to the lower bound (quality requirement) and upper bound (budget constraint) for the many-to-one matching problem.

Rather than optimality, *stability* is the main objective of our task assignment framework to address the conflicting interests among stakeholders on crowdsourcing platforms. An optimal but unstable task assignment will be disturbed by those workers and crowdsourcers who have incentives to deviate from the assignment result to pursue a higher utility. Unfortunately, existing stable algorithms for many-to-one matching problem with both lower and upper bounds cannot be applied here, since they assume that agents are homogeneous in terms of their "size", and thus unable to handle the heterogeneity in worker skill levels and their required payments. Such heterogeneity brings in great difficulties, because different combinations of workers lead to different quality levels and incur different costs. To tackle this problem, as a highlight in this paper, we design a novel algorithm that can efficiently converge to a stable task assignment. Through an extensive ar-

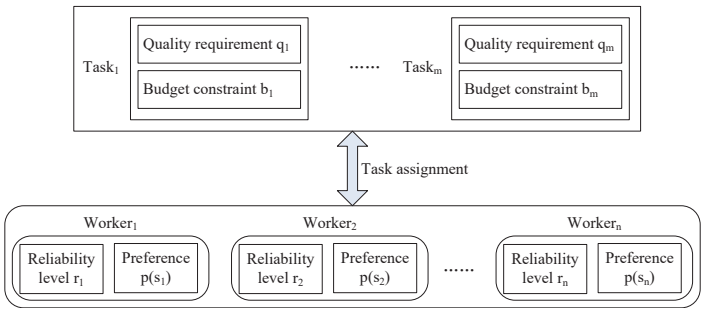


Fig. 1. The architecture of our proposed task assignment framework for crowdsourcing platforms.

ray of simulations, we compare the proposed algorithm with a benchmark algorithm for many-to-one matching problem with heterogeneous agents but no lower bounds. We show that our proposed matching algorithm can greatly improve the success ratio of completed tasks and worker happiness, at the expense of a slight increase in running times.

Our original contributions in this paper are two-fold. *First*, we propose a new many-to-one matching framework to model task assignment for crowdsourcing platforms, resolving conflicting interests between workers and crowdsourcers. *Second*, in the context of our new framework, we develop a novel stable matching algorithm for many-to-one matching with a lower bound (quality requirement), an upper bound (budget constraint), and heterogeneous agents (workers). We have conducted an extensive array of simulations to validate the effectiveness of our proposed algorithm and evaluate its performance, showing its benefits to both crowdsourcers and workers.

The remainder of this paper is organized as follows. We introduce the system model in details in Section 2. In Section 3, we describe the proposed task assignment algorithm, and prove the stability of the matching result. Simulation results are presented in Section 4. We survey the related works in Section 5. Finally, we conclude this paper in Section 6.

2 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.[1] 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 a 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 $f(0) = 0$.

1. In this paper, we make the simplifying assumption that a worker's quality level r_s is the same for all tasks. In our future work, we will consider the case where a worker's performance across tasks may be different.

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 greater 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 paper, we make the simplifying assumption that $f(r_s) = r_s$. In the 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 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 than the location that is far away. Due to worker diversity, their preference lists are different from each other.

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 towards 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 1.

In this paper, 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:*

TABLE 1
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 set of tasks that have not rejected worker s yet
$Q_{\mathcal{A}}$	total quality of workers in set $\mathcal{A}$
ΔQ	total quality requirement that has not been fulfilled
ΔR	total skill level of unassigned workers

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 $\frac{|\widetilde{\mathcal{T}}|}{|\mathcal{T}|} \times 100\%$, 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 bounds exists if and only if the total quality level of all workers is greater 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:

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

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

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

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

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 paper, 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 have 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 to 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 [8], 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 non-empty 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 non-empty 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 are 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 [8] ensures that crowdsourcers make the best use of their budget 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 a 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.

3 Stable Task Assignment

In this section, we propose a new algorithm to compute a stable task assignment that satisfies hard budget constraints and improves task success ratio regarding quality requirements.

Our proposed algorithm is based on the Deferred Acceptance (DA) algorithm, a conventional way to produce stable matching results for many-to-one matching problems with upper bounds. A direct application of DA to our problem would run as follows. At the beginning, all workers propose to her favorite task, and each crowdsourcer will select an optimal set of workers among those who propose to her. The total quality of the selected workers is the highest possible under the budget constraint. The crowdsourcer puts the selected workers in her waiting list and rejects all the other workers. After that, each unassigned worker will propose to her most preferred task that has not rejected her before, and every crowdsourcer will again select an optimal set of workers among the new proposers and those in her waiting list. The iterations continue until no workers have any tasks to propose to.

Nevertheless, the direct application of the DA algorithm has two drawbacks. Firstly, it is proved that when the agents have heterogeneous sizes — *i.e.*, heterogeneous quality levels of workers in our model — the matching result of DA is unstable [9]. Secondly, the DA algorithm does not consider lower bounds, i.e., quality requirements of the tasks, and will lead to a low success ratio.

To address these problems, we propose a new task assignment algorithm as shown in Algorithm 1. In this algorithm, we first initialize the task assignment to be empty. For worker s, we keep a list of tasks that have not rejected her, denoted by $p(s)$. Worker s can propose to tasks in $p(s)$, but not the tasks outside $p(s)$.

In each iteration, we choose an unassigned worker with a non-empty $p(s)$. We find the worker's most preferred task in $p(s)$, e.g., task t. Let $Q_{\mathcal{A}}$ denote the total quality level of workers in a certain set $\mathcal{A}$, i.e., $Q_{\mathcal{A}} = \sum_{s \in \mathcal{A}} r_s$. If the quality requirement of task t has not been satisfied, i.e., $Q_{\mu(t)} < q_t$, we use algorithm ReDA in Algorithm 2 to determine whether worker s will be assigned to task t. If the quality requirement of task t has already been satisfied, we check whether the quality requirements of the other tasks can be fulfilled without worker s. If the answer is no, we will not assign worker s to task t; if the answer is yes, we again use Algorithm 2 to determine whether worker s will be assigned to task t.

We use a simple necessary condition to check whether the quality requirements of other tasks can be fulfilled without worker s. In Algorithm 1, in line 11, we calculate ΔQ, the total quality requirement that has not been fulfilled yet. Note that $x^+ = x$, if $x > 0, x^+ = 0$, if $x < 0$. In line 12, we compute ΔR, the total quality level of the unassigned workers (including worker s). It is obvious that the quality requirements of other tasks cannot be fulfilled without worker s if $\Delta R - r_s$ is less than ΔQ.

In our proposed matching algorithm, we differentiate the cases when a task's quality requirement has not been satisfied and when a task's quality requirement has already been satisfied. In the former case, we need to prioritize the objective of meeting the quality requirement of the task in concern. Whereas in the latter case, we deem it more important to meet the quality requirements of other tasks than to add one more worker to this task for the sake of improving the success ratio of all crowdsourcing tasks. The stability of the task assignment is achieved through Algorithm 2, the algorithm to determine whether worker s will be assigned to task t.

As shown in Algorithm 2, when considering the task assignment between worker s and task t, the following situations may occur.

1) If the remaining budget of task t is greater than the quality level of worker s, s will immediately be assigned to task t.
2) If the remaining budget of task t is not enough to pay worker s, we will check whether some workers currently assigned to task t can be evicted to make room for worker s. We first single out the set of workers who are less preferred than worker s, denoted by $\mathcal{A}$. Then, we will check whether there exists a subset of workers in $\mathcal{A}$ whose aggregated quality level is less than worker s, i.e., $Q_{\mathcal{B}} < r_s, \mathcal{B} \subseteq \mathcal{A}$, and whose removal will save enough budget for worker s, i.e., $r_s + Q_{\mu(t) \setminus \mathcal{B}} \leq b_t$.
   a) If the answer is yes, we will find such a subset with the minimum total quality level, denoted by $\mathcal{B}_{min}$. The workers in $\mathcal{B}_{min}$ will be rejected by task t, and worker s will be assigned to task t.
   b) If the answer is no, worker s will be rejected by task t. The task assignment remains unchanged.

We now give a simple example to illustrate how this algorithm

Algorithm 1 Task Assignment Algorithm

Input: A set of workers $\mathcal{S}$, their quality levels $\{r_s\}_{s\in\mathcal{S}}$ and preference lists $\{\succ_s\}_{\mathcal{S}}$; a set of tasks $\mathcal{T}$, their preference lists $\{\succ_t\}_{\mathcal{T}}$, quality requirements $\{q_t\}_{t\in\mathcal{T}}$, and budget constraints $\{b_t\}_{t\in\mathcal{T}}$.
Ensure: The task assignment μ.

```
Initialization μ(s) = ∅, μ(t) = ∅, ∀s ∈ S, ∀t ∈ T.
for s ∈ S do
    p(s) := orderred list of tasks according to ≻_s.
end for
while ∃s, μ(s) = ∅ and p(s) ≠ ∅ do
    t = highest ranked task in p(s).
    Remove t from p(s).
    if Q_{μ(t)} < q_t then
        μ = ReDA(s, t, μ).
    else
        ΔQ = Σ_{t∈T} (q_t − Q_{μ(t)})^+.
        ΔR = Σ_{μ(s)=∅} r_s.
        if ΔR − r_s < ΔQ then
            μ(t) = μ(t), μ(s) = ∅.
        else
            μ = ReDA(s, t, μ).
        end if
    end if
end while
```

Algorithm 2 ReDA: Revised Deferred Acceptance Algorithm

Input: Task t, worker s, temporary task assignment μ.
Ensure: Updated task assignment μ

```
if r_s ≤ b_t − Σ_{s'∈μ(t)} r_{s'} then
    Assign worker s to task t: μ(t) = μ(t) ⋃ {s}, μ(s) = t.
else
    A = {s' | s' ∈ μ(t), s' ≺_t s}.
    if ∃B ⊆ A, Q_B < r_s and r_s ≤ b_t − Q_{μ(t)\B} then
        Find such B with the minimum total quality level.
        for ∀s' ∈ B_min do
            μ(s') = ∅.
        end for
        Assign worker s to task t: μ(t) = μ(t) ⋃ {s} \ B_min,
        μ(s) = t.
    else
        μ(t) = μ(t), u(s) = ∅.
    end if
end if
```

works. As shown in Table 2, we have 6 workers with their corresponding preference lists and quality levels, as well as two tasks with their corrresponding quality requirements and budget constraints. It can be easily derived that the preference list of all crowdsourcers will be $s_5 \succ_t s_2 \succ_t s_6 \succ_t s_3 \succ_t s_4 \succ_t s_1$. The task assignment algorithm runs as follows.

Round 1: $s_1 \to t_1$ (worker s_1 proposes to task t_1). The quality requirement of task t_1 is not fulfilled, and crowdsourcer t_1 has enough budget. Worker s_1 is assigned to task t_1, and we have $\mu(s_1) = t_1$ and $\mu(t_1) = \{s_1\}$.

Round 2: $s_2 \to t_2$. The quality requirement of task t_2 is not fulfilled, and crowdsourcer t_2 has enough budget. Worker s_2 is assigned to t_2, and we have $\mu(s_2) = t_2$ and $\mu(t_2) = \{s_2\}$.

Round 3: $s_3 \to t_2$. The quality requirement of task t_2 is not fulfilled, and crowdsourcer t_2 has enough budget. Worker s_3 is assigned to t_2, and we have $\mu(s_3) = t_2$ and $\mu(t_2) = \{s_2, s_3\}$.

TABLE 2
Workers and Tasks

	s_1	s_2	s_3	s_4	s_5	s_6
$\succ$	t_1 t_2	t_2 t_1	t_2 t_1	t_2 t_1	t_2 t_1	t_2 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 4: $s_4 \to t_2$. The quality requirement of task t_2 is not fulfilled, and crowdsourcer t_2 has enough budget. Worker s_4 is assigned to t_2, and we have $\mu(s_4) = t_2$ and $\mu(t_2) = \{s_2, s_3, s_4\}$.

Round 5: $s_5 \to t_2$. The quality requirement of task t_2 is not fulfilled, but crowdsourcer t_2 does not have enough budget. 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 level 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 $\mathcal{B}_{min} = \{s_3, s_4\}$ with the least total quality level. Therefore, we have $\mu(s_5) = t_2$ and $\mu(t_2) = \{s_2, s_5\}$.

Round 6: $s_6 \to t_2$. The quality requirement of task t_2 is fulfilled. Although crowdsourcer t_2 has enough budget for worker s_6, we won't assign worker s_6 to task t_2 because doing so will make it impossible to satisfy the quality requirement of task t_1. Hence, worker s_6 is rejected by task t_2.

Round 7: $s_3 \to t_1$. The quality requirement of task t_1 is not fulfilled, and crowdsourcer t_1 has enough budget. Worker s_3 is assigned to t_1, and we have $\mu(s_3) = t_1$ and $\mu(t_1) = \{s_1, s_3\}$.

Round 8: $s_4 \to t_1$. The quality requirement of task t_1 is not fulfilled, and crowdsourcer t_1 has enough budget. Worker s_4 is assigned to t_1, and we have $\mu(s_4) = t_1$ and $\mu(t_1) = \{s_1, s_3, s_4\}$.

Round 9: $s_6 \to t_1$. The quality requirement of task t_1 is not fulfilled, and crowdsourcer t_1 has enough budget. Worker s_6 is assigned to t_1, and we have $\mu(s_6) = t_1$ and $\mu(t_1) = \{s_1, s_3, s_4, s_6\}$.

We summarize the task assignment process in Table 3. The final task assignment is $\mu(t_1) = \{s_1, s_3, s_4, s_6\}$ and $\mu(t_2) = \{s_2, s_5\}$. It can be easily checked that this task assignment satisfies budget constraints and quality requirements of all tasks, achieving a 100% success ratio.

Theorem 1. *The proposed task assignment algorithm has a computational complexity of $O(|\mathcal{S}||\mathcal{T}|\tau)$, in which τ is the running time of finding $\mathcal{B}_{min}$ in Algorithm 2.*

Proof. The number of iterations in Algorithm 1 is at most $|\mathcal{S}|*|\mathcal{T}|$, since there are $|\mathcal{S}|$ workers, who may propose to no more than $|\mathcal{T}|$ tasks. The most difficult part is to find the set of workers $\mathcal{B}_{min}$ in Algorithm 2, which can be transformed into an integer programming problem. Integer programming is NP-hard, and there are many algorithms for solving the integer programming problem such as the cutting-plane and the branch-and-bound methods [10]. □

To prove that the proposed algorithm can reach a stable task assignment result, we first introduce the following lemma.

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

TABLE 3
Task Assignment Process

Round	Action	Available budget	Aggregate quality	Matching result
1	$s_1 \to t_1$	$b'_1 = 1.5; b'_2 = 1.3$	$q'_1 = 0.4; q'_2 = 0$	$\mu(t_1) = \{s_1\}; \mu(t_2) = \emptyset$
2	$s_2 \to t_2$	$b'_1 = 1.5; b'_2 = 0.75$	$q'_1 = 0.4; q'_2 = 0.55$	$\mu(t_1) = \{s_1\}; \mu(t_2) = \{s_2\}$
3	$s_3 \to t_2$	$b'_1 = 1.5; b'_2 = 0.45$	$q'_1 = 0.4; q'_2 = 0.85$	$\mu(t_1) = \{s_1\}; \mu(t_2) = \{s_2, s_3\}$
4	$s_4 \to t_2$	$b'_1 = 1.5; b'_2 = 0.25$	$q'_1 = 0.4; q'_2 = 1.05$	$\mu(t_1) = \{s_1\}; \mu(t_2) = \{s_2, s_3, s_4\}$
5	$s_5 \to t_2$	$b'_1 = 1.5; b'_2 = 0.15$	$q'_1 = 0.4; q'_2 = 1.15$	$\mu(t_1) = \{s_1\}; \mu(t_2) = \{s_2, s_5\}$
6	$s_6 \to t_2$	$b'_1 = 1.5; b'_2 = 0.15$	$q'_1 = 0.4; q'_2 = 1.15$	$\mu(t_1) = \{s_1\}; \mu(t_2) = \{s_2, s_5\}$
7	$s_3 \to t_1$	$b'_1 = 1.2; b'_2 = 0.15$	$q'_1 = 0.7; q'_2 = 1.15$	$\mu(t_1) = \{s_1, s_3\}; \mu(t_2) = \{s_2, s_5\}$
8	$s_4 \to t_1$	$b'_1 = 1; b'_2 = 0.15$	$q'_1 = 0.9; q'_2 = 1.15$	$\mu(t_1) = \{s_1, s_3, s_4\}; \mu(t_2) = \{s_2, s_5\}$
9	$s_6 \to t_1$	$b'_1 = 0.9; b'_2 = 0.15$	$q'_1 = 1; q'_2 = 1.15$	$\mu(t_1) = \{s_1, s_3, s_4, s_6\}; \mu(t_2) = \{s_2, s_5\}$

Proof. Let μ and $\widetilde{\mu}$ denote the task assignment before and after worker s proposes to task t, respectively.

- If worker s is rejected by task t, the task assignment remains unchanged, i.e, $\mu = \widetilde{\mu}$. Hence, $Q_{\widetilde{\mu}(t)} = Q_{\mu(t)}$.
- If worker s is assigned to task t without evicting any of task t's previously assigned workers, we have $\widetilde{\mu}(t) = \mu(t) \bigcup \{s\}$. It is obvious that $Q_{\widetilde{\mu}(t)} > Q_{\mu(t)}$.
- If worker s is assigned to task t by evicting some of task t's previously assigned workers, we have $\widetilde{\mu}(t) = \mu(t) \bigcup \{s\} \setminus \mathcal{B}_{min}$. According to Algorithm 2, $Q_{\mathcal{B}_{min}} < r_s$. Hence $Q_{\widetilde{\mu}(t)} = Q_{\mu(t)} - Q_{\mathcal{B}_{min}} + r_s > Q_{\mu(t)}$.

□

Theorem 2 (Individual Rationality)**.** *The proposed algorithm produces an individually rational task assignment.*

Proof. We have assumed that every task is acceptable to each worker. Therefore, every worker prefers the current task assignment to being unassigned. Under the budget constraint, a crowdsourcer prefers a larger set of workers as it improves the total quality level. Therefore, every crowdsourcer prefers the current set of assigned workers to any subset of these workers. The proposed algorithm is, therefore, individual rational. □

Theorem 3 (Nonwastefulness)**.** *The proposed algorithm produces a nonwasteful task assignment.*

Proof. We prove the nonwastefulness of the proposed 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 task t, but is rejected. Let $\widetilde{\mu}$ denote the task assignment at the time worker s proposes to task t. One of the following three situations must be true.

- The quality requirement of task t is not fulfilled, but the budget of task t is not enough for worker s, because otherwise worker s will be assigned to task t immediately. We have $Q_{\widetilde{\mu}(t)} + r_s > b_t$. According to Lemma 1, the budget of task t is non-decreasing, i.e., $Q_{\mu(t)} \geq Q_{\widetilde{\mu}(t)}$. Therefore, under the final task assignment, we have $Q_{\mu(t)} + r_s > b_t$, and worker s and task t cannot form a type II blocking pair.
- The quality requirement of task t is fulfilled, and without worker s, the remaining unassigned workers can satisfy the quality requirements of other tasks, but the budget of task t is not enough for worker s, i.e., $Q_{\widetilde{\mu}(t)} + r_s > b_t$. According to Lemma 1, the budget of task t is non-decreasing, i.e., $Q_{\mu(t)} \geq Q_{\widetilde{\mu}(t)}$. Therefore, under the final task assignment, we have $Q_{\mu(t)} + r_s > b_t$, and worker s and task t cannot form a type II blocking pair.
- The quality requirement of task t is fulfilled, but 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 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, either because task t will not have enough budget for worker s, or because worker s is essential to fulfill the quality requirement of her currently assigned task $\mu(s)$. □

Theorem 4 (Fairness)**.** *The proposed algorithm produces a fair task assignment.*

Proof. We prove that our proposed algorithm is fair by contradiction. Assume that under the final task assignment μ, there is a type I blocking pair (s, t). Since $t \succ_s \mu(s)$, worker s must have proposed to task t, but is rejected. Let $\widetilde{\mu}$ denote the task assignment at the time worker s proposes to task t. One of the following three situations must be true.

- The quality requirement of task t is not fulfilled, but there does not exist such a set of workers as $\mathcal{B}$ in Algorithm 2, who are less preferred than worker s, have a lower total quality level and can vacate enough budget for worker s. As the task assignment algorithm runs, the set of workers less preferred than worker s will shrink, and the remaining budget of task t will decrease (Lemma 1). Therefore, under the final task assignment, there will not exist a set of workers that satisfy the condition in Definition 3, thus task t and worker s cannot form a type II blocking pair.
- The quality requirement of task t is fulfilled, and without worker s, the remaining unassigned workers can satisfy the quality requirements of other tasks, but there does not exist such a set of workers as $\mathcal{B}$ in Algorithm 2, who are less preferred than worker s, have a lower total quality level and can vacate enough budget for worker s. As the task assignment algorithm runs, the set of workers less preferred than worker s will shrink, and the remaining budget of task t will decrease (Lemma 1). Therefore, under the final task assignment, there will not exist a set of workers that satisfy the condition in Definition 3, thus task t and worker s cannot form a type II blocking pair.
- The quality requirement of task t is fulfilled, but without worker s, the remaining unassigned workers cannot satisfy the quality requirements of other tasks. In this case, in the final assignment, worker s cannot shift to task t because the quality requirement of task $\mu(s)$ will be violated.

In summary, worker s and task t cannot form a type I blocking pair, either because worker s cannot displace any set of workers currently assigned to task t, or because worker s is essential to fulfill the quality requirement of her currently assigned task $\mu(s)$. □

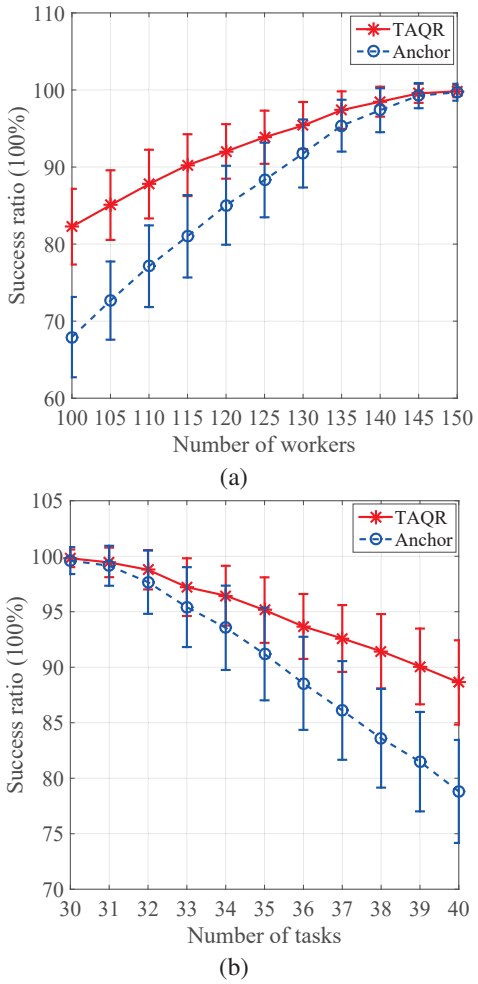


Fig. 2. Success ratio comparison. (a) $|\mathcal{T}| = 30$; (b) $|\mathcal{S}| = 150$.

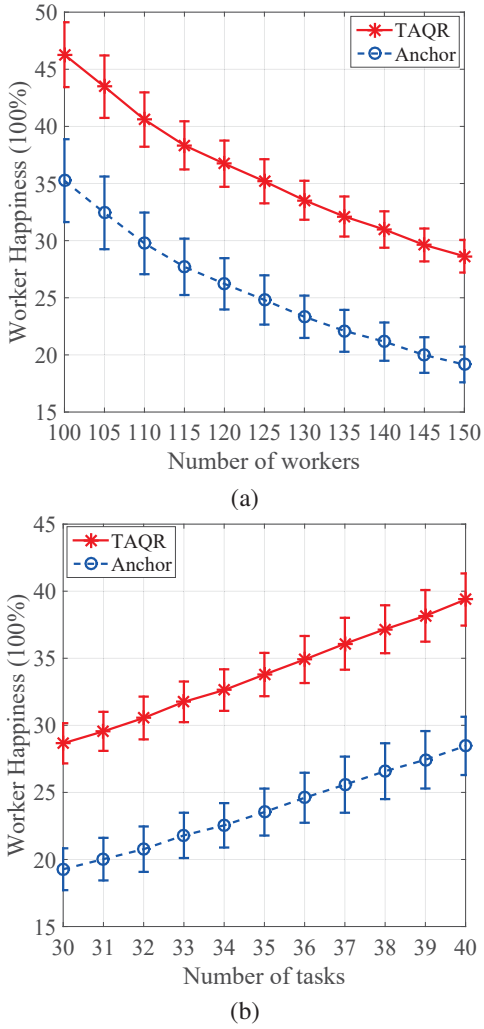


Fig. 3. Happiness comparison. (a) $|\mathcal{T}| = 30$; (b) $|\mathcal{S}| = 150$.

Theorem 5 (Stability)**.** *The proposed algorithm produces a stable task assignment.*

Proof. According to Theorem 2, Theorem 3 and Theorem 4, the final task assignment is individually rational, fair and nonwasteful. Therefore, the proposed task assignment algorithm is stable. □

4 Simulation

In this section, we evaluate the performance of the proposed algorithm, referred to as Task Assignment with Quality Requirement (TAQR) in the following figures. We implement Anchor [9], 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 non-wastefulness. We compare our proposed matching algorithm, i.e., TAQR, 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(R) Core (TM) i5-3230M CPU at 2.60GHz and 4.00GB RAM.

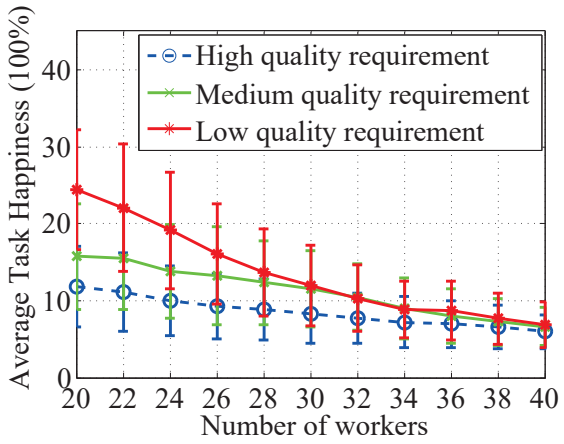
4.1 Success Ratio

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. 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, as shown in Fig. 2(a). Compared with the benchmark, TAQR can achieve up to 16% improvement when the number of workers are relatively small. With more and more workers, the gap between TAQR and the benchmark narrows down. As expected, the success ratio will reach 100% when the number of workers is large enough for all tasks to fill up their quality requirement. With insufficient workers, TAQR prioritizes tasks whose quality requirements have not been satisfied when assigning workers, leading to a higher success ratio than the benchmark.

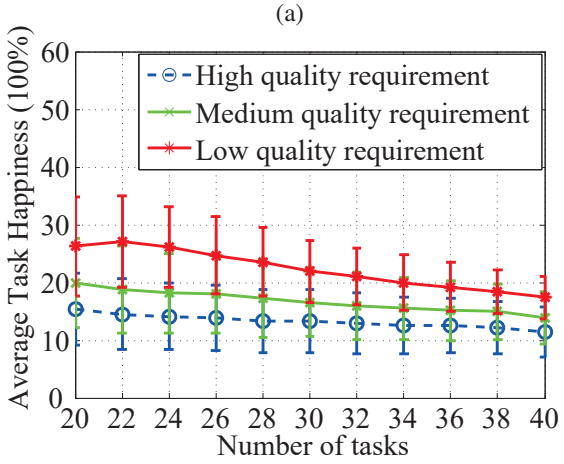
The success ratio will decrease as more and more tasks compete for workers, as shown in Fig. 2(b). TAQR has a slower decrease rate than the benchmark, and can achieve up to 18% gain in success ratio. The benchmark suffers from worker deficiency more severely as it aggressively rejects workers to ensure fairness without considering fulfilling the quality requirements of tasks.

4.2 Happiness of Workers and Tasks

We have the same definition of happiness as [9]. The happiness of a worker is the rank percentile of her assigned task. For

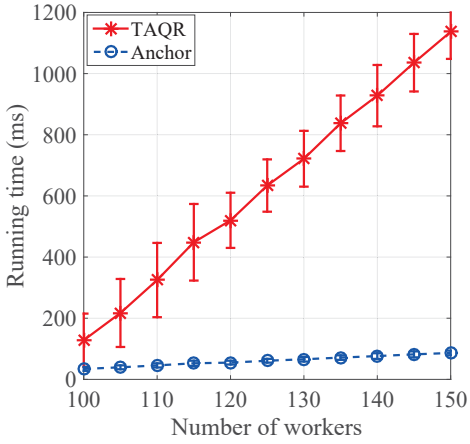


(a)

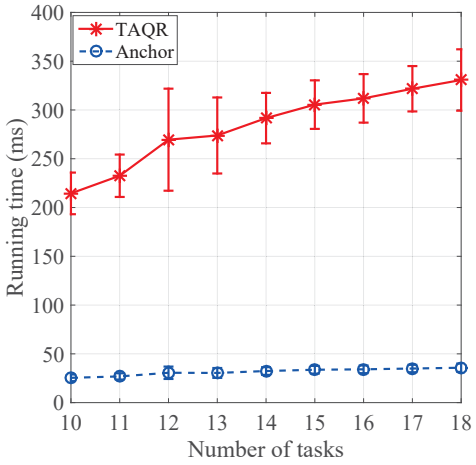


(b)

Fig. 4. Happiness vs quality requirements (a) $|\mathcal{T}| = 125$; (b) $|\mathcal{S}| = 30$.



(a)



(b)

Fig. 5. Running time comparison. (a) $|\mathcal{T}| = 30$; (b) $|\mathcal{S}| = 100$.

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; 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 $(100 + 50)/2 * 100\% = 75\%$. 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 $[1, 4]$, respectively.

As shown in Fig. 3(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 maintains a 11% gain over the benchmark because TAQR does not reject workers as radically as the benchmark and makes a better use of the budget to employ more workers. The worker 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. 3(b). The worker happiness of TAQR is around 11% higher than that of the benchmark.

4.3 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, but not the benchmark. In this set of simulations, the budget constraints $b_t, \forall t \in \mathcal{T}$ and worker quality levels $r_s, \forall s \in \mathcal{S}$ are randomly chosen from $[6, 10]$ and $[1, 4]$, respectively. We study the scenarios when tasks have high, medium and low quality requirements, with $q_t, \forall t \in \mathcal{T}$ randomly chosen from $[3, 4]$, $[2, 3]$, $[1, 2]$, respectively. As illustrated in Fig. 4, 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. 4(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, average task happiness gradually descends due to a lack of workers, as shown in Fig. 4(b).

4.4 Running time

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 $[0.8, 1.5]$, $[1.4, 2]$ and

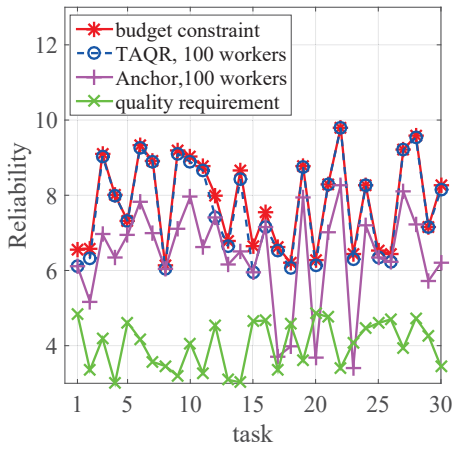


Fig. 6. Matching result in terms of the aggregate quality level.

$[0, 1]$, respectively. As shown in Fig. 5, though the running time of TAQR is linear in the number of workers and the number of tasks, it is much longer than that of the benchmark. Taking into account the results in Fig. 2 and Fig. 3, it is clear that there is a trade-off between the running time and the performance improvement. The benchmark rapidly culls out less-preferred workers to ensure fairness but not the quality requirements of tasks. In comparison, TAQR carefully examines every worker for potential assignment, and makes a better use of the budget to hire as many workers as possible, resulting in a higher success ratio and worker happiness at the expense of a longer running time.

4.5 Aggregate Quality Level Comparison

By taking a closer look at the task assignment result, we demonstrate that TAQR makes a better use of the budget and improves the success ratio of tasks. We consider 20 tasks and 60 workers, with the quality requirements $q_t, \forall t \in \mathcal{T}$, budget constraints $b_t, \forall t \in \mathcal{T}$ and worker quality levels $r_s, \forall s \in \mathcal{S}$ randomly chosen from $[3, 5]$, $[6, 10]$ and $[1, 4]$, respectively. We run the simulation once, and show the aggregate quality level of assigned workers to each task in Fig. 6. It is clearly that TAQR meets the quality requirement of every task and tries to use available budgets to attain the highest possible quality level. 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 TAQR can improve the success ratio (Fig. 2) and worker happiness (Fig. 3).

5 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 are 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 assesses worker skills while the exploitation phase performs the task assignment for profit maximization. In [3], Boutsis et al. presented CRITICAL, to determine the appropriate group of workers to every task under reliability and time constraints. In [4], Feng et al. 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. In [6], Xiao et al. proposed an offline task assignment algorithm and an online task assignment algorithm based on a greedy strategy. In [7], Karaliopoulos et al. devised a greedy heuristics worker selection algorithm to deal with uncertainty of worker mobility. In [11], Han et al. presented an online algorithm to achieve robust crowdsensing. In [12], by taking the geographical characteristics of sensing tasks and the spatial movement constraints of mobile workers into account, the problem of allocating location-dependent tasks is studied. In [13], Cheung et al. designed a distributed task selection algorithm to collect time-sensitive and location-dependent data from heterogeneous workers.

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. In [14], Gale and Shapley first proposed and then analyzed the problems of stable matching. According to [15], the deferred acceptance algorithm is used to achieve a stable matching. Variants of matching problems in economics have been examined [16], [17], [18], [19]. In [20], Hamada et al. focused on the hospital-residents matching problem with quota lower bounds, and proposed an exponential-time algorithm. The theory of matching has also been explored in computer science. In [8], Fragiadakis introduced two classes of strategyproof mechanisms for many-to-one matching with minimum quotas. In [9], Xu et al. proposed both online and offline algorithms to match virtual machines to heterogeneous sized jobs in the cloud. In [21], two generalized stable matching problems that exist in the higher education sector with lower and common quotas are studied. However, none of these matching frameworks can be applied to task assignment with quality requirements and budget constraints, where the heterogeneity in worker quality levels makes it challenging to reach a stable matching result.

6 Conclusion

In this paper, 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 towards 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 a stable matching algorithm, which can yield task assignment results that are individual rational, fair and nonwasteful. Through extensive simulation results, we verified

that our proposed algorithm can improve both the task success ratio and the happiness of both workers and tasks with acceptable time complexity.

7 Acknowledgment

This work was supported in part by NSFC under Grant Nos. 61202393, the CPSF under Grant No. 2012M521797, the International Cooperation Foundation of Shaanxi Province, China under Grant No. 2013KW01-02, and the International Postdoctoral Exchange Fellowship Program 2013 under Grant No. 57 funded by the office of China Postdoctoral Council. The co-authors would also like to acknowledge the generous research support from a NSERC Discovery Research Program and a NSERC Strategic Partnership Grant titled "A Cloud-Assisted Crowdsourcing Machine-to-Machine Networking Platform for Vehicular Applications" at the University of Toronto.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile Crowdsensing: Current State and Future Challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[2] C. J. Ho and J. W. Vaughan, "Online Task Assignment in Crowdsourcing Markets," in *The Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[3] I. Boutsis and V. Kalogeraki, "On Task Assignment for Real-time Reliable Crowdsourcing," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2014.

[4] Z. Feng, Y. Zhu, Q. Zhang, H. Zhu, J. Yu, J. Cao, and L. M. Ni, "Towards Truthful Mechanisms for Mobile Crowdsourcing with Dynamic Smartphones," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2014.

[5] Z. He, J. Cao, and X. Liu, "High Quality Participant Recruitment in Vehicle-based Crowdsourcing using Predictable Mobility," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[6] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu, "Multi-Task Assignment for Crowdsensing in Mobile Social Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[7] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User Recruitment for Mobile Crowdsensing over Opportunistic Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[8] D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof Matching with Minimum Quotas," *ACM Transactions on Economics and Computation*, vol. 4, no. 1, p. 6, 2016.

[9] H. Xu and B. Li, "Anchor: A Versatile and Efficient Framework for Resource Management in the Cloud," *IEEE Transactions on Parallel and Distributed System*, vol. 24, no. 6, pp. 1066–1076, 2013.

[10] R. S. Garfinkel and G. L. Nemhauser, *Integer programming*. New York: Wiley, 1972.

[11] K. Han, C. Zhang, and J. Luo, "BLISS: Budget Limited Robust Crowdsensing through Online Learning," in *IEEE International Conference on Sensing, Communication and Networking (SECON)*, 2014.

[12] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward Optimal Allocation of Location Dependent Tasks in Crowdsensing," in *IEEE Conference on Computer Communications (INFOCOM)*, 2014.

[13] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed Time-Sensitive Task Selection in Mobile Crowdsensing," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2015.

[14] D. Gale and L. S. Shapley, "College Admissions and the Stability of Marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[15] A. E. Roth, "Deferred Acceptance Algorithms: History, Theory, Practice, and Open Questions," *International Journal of Game Theory*, vol. 36, no. 3-4, pp. 537–569, 2008.

[16] M. Pycia, "Many-to-One Matching Without Substitutability," *MIT Industrial Performance Center Working Paper*, vol. 8, pp. 1–48, 2005.

[17] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, "Peer Effects and Stability in Matching Markets," *Algorithmic Game Theory*, pp. 117–129, 2011.

[18] A. E. Roth, "The College Admissions Problem Is Not Equivalent to the Marriage Problem," *Journal of economic Theory*, vol. 36, no. 2, pp. 277–288, 1985.

[19] L. Ehlers, I. E. Hafalir, M. B. Yenmez, and M. A. Yildirim, "School Choice with Controlled Choice Constraints: Hard Bounds versus Soft Bounds," *Journal of Economic Theory*, vol. 153, pp. 648–683, 2014.

[20] K. Hamada, K. Iwama, and S. Miyazaki, "The Hospitals/Residents Problem with Quota Lower Bounds," in *European Symposium on Algorithms*. Springer Berlin Heidelberg, 2011.

[21] P. Biró, T. Fleiner, R. W. Irving, and D. F. Manlove, "The College Admissions Problem with Lower and Common Quotas," *Theoretical Computer Science*, vol. 411, no. 34, pp. 3136–3153, 2010.