2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems

Combined Crowdsourcing Task Auction Mechanism Based On Stable Matching

Lanyu Zhang*, Mingjun Xiao*, Hui Zhao*, and Jianchun Liu*
*School of Computer Science and Technology / Suzhou Institute for Advanced Study,
University of Science and Technology of China

***Abstract*—Crowdsourcing allows requesters to allocate tasks to a group of workers on the internet to make use of their collective intelligence. In this paper, we consider the complementarity among different types of tasks, which means the cost of a worker to complete a task combination is less than the sum of the costs that the worker completes each task in the combination separately. In the crowdsourcing, task assignment is one of the most essential problems, we formulate the crowdsourcing task assignment as a many-to-many matching problem. However, most existing works ignore the preferences of crowdsourcers and workers. Since both of workers and crowdsourcers are selfish individuals who are not necessarily aligned with the system optimization. They can violate the task assignment to obtain a higher utility. Different from the traditional task assignment mechanisms which focus on social welfare maximization, we not only consider the preferences of crowdsourcers and workers, but also obtain a stable matching result for the many-to-many matching problem. Payment determination is also a essential problem in the crowdsourcing system, so we compute the corresponding payment profiles for both crowdsourcers and workers. Extensive simulations have been performed, and the result demonstrates that our proposed mechanism not only obtains a stable matching but also has a high individual utility for each worker and each crowdsourcer. Workers and crowdsourcers would comply with the allocation results.**

***Index Terms*—Crowdsourcing, complementarity, stable matching**

I. Introduction

Along with the proliferation of mobile internet, various crowdsourcing platforms have been developed in recent years. The integration of smartphones and crowdsourcing creates the mobile crowdsourcing [1], which enables us to harness the power of the crowd to share information or improve a service[2],[3],[4]. Crowdsourcing is playing an important role nowadays. First, a large population of service providers suggest that service demands can be satisfied with a low cost, then social cost can be effectively reduced. Second, service providers can obtain rewards by contributing services to those users.

There are many types of crowdsourcing tasks by now, such as labeling tasks, sensing tasks(e.g., perceiving the air quality in a certain area, observing the weather, reporting traffic conditions on the road etc.), takeaway crowdsourcing tasks etc. Some tasks require the professional skills to be completed, and some don't need.

In practice, complementarity may exists when a worker is assigned a set of tasks and is asked to complete them continuously. It can be divided into two situations. One is time complementary, which means that tasks can be completed in parallel. The other is spatial complementary, which means that the locations of the tasks are spatially similar. After completing a task, the costs of other tasks would be smaller. For example, a worker Bob is allocated a task set $C = (A, B)$, the task A is to ask the traffic condition of a certain road and task B is to take a picture of a specific building next to the road. Then the cost of Bob may reduce compared to accomplishing the task A and B respectively. This example explains the spatial complementary among tasks. The time complementary means that the two types of tasks A and B can be completed in parallel, which can help workers save time costs.

The assignment problem in our crowdsourcing system is different from the problem in [5], which is modeled as a many-to-one matching problem. We model task assignment as a many-to-many matching problem and it is more difficult to reach a stable matching. Workers can freely express their preferences for different combinations of tasks, and crowdsourcers also select some of workers. Considering the complementarity among different types of tasks, we can use the model of the combinatorial spectrum auction to solve this problem.

The main contributions of this paper are listed as follows:

1) We consider spatiotemporal complementarity among the tasks. The different combinations of tasks bring different costs for workers, which may be less than the sum of the costs of accomplishing the tasks separately.
2) We propose a stable many-to-many matching framework to model task assignment for crowdsourcing system, resolving conflicting interests between workers and crowdsourcers, and determine the payoff that crowdsourcers should pay for workers.
3) We have provided both theoretical analysis and extensive simulations which verify the effectiveness of our system. We also prove that our system satisfies the properties of convergence and stability.

The remainder of the paper is organized as follows. In the next section we present the system model. In Section III, we describe the stable combination matching algorithm. In section IV, we present the evaluation of the algorithm. In section V, we present the experiment results. We conclude the paper in section VI.

II. System Models

A. System overview

In this section, we present a system model for the task-distribution combination stable matching problem in crowdsourcing, and we also define the stability under such model.

We assume that there is a set of workers W on the crowsourcing platform and any worker $i \epsilon W$ could accomplish a combination of tasks. Due to workers' diversity in skills, proficiency and the currently status, workers will have different costs for different task combinations. We assume that once the task is completed by worker, the completed task's quality is the same. But workers who completed the same task combination may have different costs. Each worker can only complete one task of the same type.

We also assume that there is a set of crowdsourcers M on the crowsourcing platform. Since each crowdsourcer publishes one type task, we also can say the task set M. In our system, each crowdsourcer publishes one type and a certain number of tasks on the crowdsourcing platform with a budget. Each task only need to be completed once. When the task needs to be completed multiple times, we can treat this task as multiple tasks. The budget represents the most reward that the crowdsourcer can pay for the workers, which limits the number of workers that can be hired by the crowdsourcer. The number of tasks published by crowdsourcers also limits the number of workers that can be recruited by crowdsourcers. Each crowdsourcer wants to recruit as many workers as possible under his budget. Because he hopes his tasks to be completed as much as possible.

B. The utility of workers and crowdsourcers

Let p_{ij} denote the payment of crowdsourcer j to the worker i if worker i completed the task published by crowdsourcer j, and $p_i = (p_{i1}, p_{i2}, ...p_{im})$ is the payment profile of the worker i. Without loss of generality, we assume that worker i is allocated the combination of tasks Γ^i. Then the cost of worker i to complete this combination is $C^i(\Gamma^i)$. Obviously, if the worker is assigned an empty combination of tasks, which he is not willing to accomplish any task. His cost will be zero, i.e., $C^i(\varnothing) = 0$. Next, we define the worker's utility function as $U^i(C^i, p^i) = \sum_{j \epsilon \Gamma^i} p_{ij} - C^i(\Gamma^i)$.

As for crowdsourcers, we denote D^j as a worker set that matches the tasks published by crowdsourcer j. $p^j = (p_{1j}, p_{2j}...p_{nj})$ is the payment profile of the crowdsourcer j, and we denote p_{ij} as the payment that crowdsoucer j should pay for worker i. We denote $V_j()$ as the valuation function of crowdsourcer j. The function is proportional to the number of workers matching the crowdsourcer j. Obviously, if the worker set D^j is empty, then crowdsourcer $j's$ utility is zero, e.g., $V^j(\varnothing) = 0$. We define the utility function of crowsourcer as $W^j(D^j, p^j) = V^j(D^j) - \sum_{i \epsilon D^j} p_{ij}$.

In order to obtain a stable match of our proposed algorithm, we have the following reasonable assumptions regarding the utility function of workers and crowdsoucers:

- Each crowdsourcer declares a peak price, which is equivalent to the utility that a single task brings to the crowdsourcer when it is completed. It also represents the highest reward the crowdsourcer can give to a single worker. We denote the r_j as the peak price of crowdsoucer j.
- If a worker is accepted by a task in a certain round, the worker will continue to bid for the task in the next round. In other words, the task combination that brings the most utility to the workers in this round must include the task of accepting the worker in last round.

C. Stable task assignment

We formally define a stable combinatorial matching of workers and tasks as follows:

Definition 1 (matching of tasks and crowdsourcers). Given the set of tasks M and the set of workers W, we formally define the task assignment as a many-to-many matching problem with budget and quantity constraints. The matching is a mapping u from the set $M \cup W$ into the set of all subsets of $M \cup W$, such that:

1) For every crowdsourcer $m \epsilon M, u(m) \subseteq W$
2) For every worker $w \epsilon W, u(w) \subseteq M$
3) For every worker w and crowdsourcer m, $w \epsilon u(m)$ if and only if $m \epsilon u(w)$.

A task assignment is stable only if it has properties of individual rational, fair and non-wasteful. We next introduce these properties as follows:

Definition 2 (individual rationality). We define the property of individual rationality as follows:

1) Such as crowdsourcers $j\epsilon M$, he recruits a set of workers C under the payment profile p_j , and obtain a non-negative utility, e.g., $|C| * r_j - \sum_{i \epsilon C} p_{ij} \geq 0$.
2) Such as workers $i\epsilon N$ is employed by a crowdsourcer set T under the payment profile p_i and obtain a non-negative utility, e.g., $\sum_{j \epsilon T} p_{ij} - cost(T) \geq 0$.

Generally speaking, individual rationality is a extremely important property for a successful task assignment. Because it is the basis of a stable matching. It ensures that crowdsourcers are willing to publish the tasks and workers are willing to accomplish the tasks.

Before defining stability, we introduce the concept of type I blocking pair.

Definition 3 (Type I Blocking Pair). Given a feasible matching result μ and the payment profile P, we denote A as a non-empty subset of worker who is matched task t, i.e., $A \subseteq \mu(t)$. Worker s and task t form a type I blocking pair (s, t), if the following conditions are met:

1) Worker s is willing to accomplish task t under the profile p_s.
2) There exists a worker $s' \epsilon A$, crowdsourcer t is willing to replace s' with worker s under the p_s, and without violating the budget constraint of crowdsourcer t.

The type I block pair makes a task assignment unstable, because the crowdsourcer in concern is willing to replace a less-preferred worker who has been assigned with a more preferred worker.

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 his current task, because he cannot replace assigned worker to his more-preferred tasks. Next we define the Type II blocking pair as follows:

Definition 5 (**Type II blocking pair**). Given a feasible matching result μ and the payment profile P, we denote A as a non-empty subset of worker who is matched task t, i.e., $A \subseteq \mu(t)$. Worker s and task t form a type II blocking pair (s, t), if the following conditions are met:

1) Worker s is willing to accomplish task t under the profile p_s.
2) Crowdsourcer t is willing to add worker s under the p_s without violating the budget constraint and quantity constraint.

The type II blocking pair makes a task assignment unstable, because the type II blocking pair indicates that there is a crowdsourcer with enough budget to hire one more worker who is willing to work for him.

Definition 6 (**Nonwastefulness**). A feasible task assignment μ is Nonwasteful [6] if and only if there are no type II blocking pairs.

The property of nonwastefulness indicates that each crowdsourcer would make the best use of their budget to recruit workers. The main difference between type I and type II blocking pairs is whether the crowdsourcer is willing to abandon a currently matched worker for a more-preferred worker. If the crowdsourcer is willing to make this replacement, then it is type I blocking pair. Otherwise, it is a type II blocking pair.

Definition 7 (**Strong stability**). A feasible task assignment μ is strong stable if it meets three properties of individual rational, fair and nonwasteful.

III. Algorithm of stable combination matching

A. Stable matching algorithm design

The traditional deferred acceptance algorithm has been designed to solve the many-to-one stable matching problem. Nevertheless, in our matching problem, we can't direct applying the traditional deferred acceptance algorithm. First, our combination matching problem is a many-to-many problem, but the traditional deferred acceptance algorithm is design to solve the many-to-one matching problem. Second, in our matching problem, each crowdsourcer not only constrained by the number of tasks but also has a budget constrain, Third, we not only need a stable combination matching, but also need computing the corresponding payment that crowdsourcers should pay for workers, and the traditional acceptance algorithm is fair to achieving that.

In the matching of tasks and workers, starting with a set of free workers and a set of unassigned tasks, each worker would propose a task combination that could brings highest utility to her. Each worker has a bid for task which in task combination he proposed. Each crowdsoucer decides the winner set of workers based on the crowdsourcer's budget and number of tasks he published. During the matching process, each worker will gradually reduce their bid for the crowdsourcer who has rejected him. As the worker's bids fall, the crowdsourcer will accept the worker's proposal or worker gives up proposing to the crowdsourcer. We denote $b_{ij}(t)$ as the provisional payment of crowdsourcer j for worker i at the round t, $\mu(t)$ denoted as the provisional matching at the round t. We present the detailed process as show in Algorithm 1.

combination	worker 1	worker 2	worker 3	worker 4
A	6	3	4	8
B	6	7	10	7
C	9	3	4	6
AB	10	8	11	14
AC	9	9	7	12
BC	9	15	14	13
ABC	12	17	16	20

TABLE I: The cost of task combinations

Algorithm 1 Devised Deferred Acceptance Algorithm For Task Assignment

```
Input: $W(D,P)$,$U(C,P)$, $R = \{r_1, r_2, ...r_n\}$, $A$, $\sigma$, $B$;
Output: Matching result $\mu$; Payment $p_{ij}$,$\forall i \epsilon N$,$\forall j \epsilon M$ ;
1: t = 0, flag = 1;
2: for All $i \in \mathcal{N}$, $j \in \mathcal{M}$ do
3:     $\mu(i) = \phi, \mu(j) = \phi$;
4:     $b_{ij} = r_j$;
5: for All Crowdsoucer $j \epsilon M$ do
6:     initiate worker waiting list $L_j = \phi$;
7: while $flag$ do
8:     $t = t + 1$;
9:     for all worker $i \ \epsilon \ N$ do do
10:        Worker $i$ finds the crowdsourcer combination $C_i(t)$ that
           maximize $U_i(C_i(t), b_i(t))$
11:        for all crowdsourcer $j \ \epsilon \ M$ do do
12:            update the waiting list $L_j = L_j \bigcup i$
13:    for all crowdsourcer $j \ \epsilon \ M$ do do
14:        sort the bids in the $L_j$ in the ascending order;
15:        find the most smallest $k+1$ that $\sum_{i=1}^{k+1} b_{ij} \geq B_j$ and
           $k \leq a_j$, which denoted as $D_j$;
16:        crowdsourcer $j$ rejects workers $i \ \epsilon \ L_j \backslash D_j$ ;
17:    for all $i \ \epsilon \ N, j \ \epsilon \ M$ do do
18:        if crowdsourcer $j$ has rejected worker $i's$ proposal then
19:            $b_{ij}(t) = b_{ij}(t-1) - \sigma$
20:    $flag = sum_{ij}(b_{ij}(t+1) - b_{ij}(t))$
21: for all crowdsourcer $j \ \epsilon \ M$ do do
22:    for all worker $i \ \epsilon \ L_j$ do do
23:        $\mu(i) = \mu(i) \bigcup j, \mu(j) = \mu(j) \bigcup i$
24:        $p_{ij} = b_{ij}^t$
```

B. An Illustrative Example

In this section, we give a simple example to illustrate how the algorithm works. We first consider the limit on the number of workers for each crowdsourcer, then consider the budget constraint. As show in the TABLE I, there are four workers $1, 2$, 3 and 4, and three types of tasks A, B and C, we can also call three crowdsourcers A, B and C too. The number of tasks A, B and C all are 2, and the budgets of the crowdsourcers A, B and C are $10, 11$ and 14 respectively, and the peak price are $7, 9$ and 9. We set bid adjustment step size σ as 1. The table I presents the workers' cost for different combinations of tasks. As we can see, the cost of a task combination may be less than the sum of cost of individual task in the task combination due to the complementarity. In the Fig.1, on the right side of the crowdsourcer is the worker list he is willing to accept, and the left side of the worker represents his bid list to the corresponding crowdsourcers. At the round $t = 0$, as show in the Fig.1(a), worker 1 proposes to the task combination (A, B, C) at the crowdsourcers' peak price $7, 9$, and 9 respectively, which would gives him maximum utility. For worker 2, he proposes to the combination A and B at the crowdsourcers' peak price 7 and 9. The worker 3 proposes the combination (A, C) at the price $7, 9$. The worker 4 proposes the combination (B, C) at the price 9 and 9. Due to all workers propose the crowdsourcer A at the same price, the crowdsourcer A randomly accepts worker 1 and worker 2, and rejects the worker 3. Crowdsourcers B and C are the same. In the second round, as show in the Fig.3(b), the accepted bid is unchanged and the rejected bid minus the bid adjustment step size, and crowdsourcers select workers according the worker's bid. The same is true for the next few rounds. In the Fig.1(h), the worker 4 prefers the task combination (C) to the tasks combination (B, C). In the Fig.1(i), the algorithm begins to consider the budget constraints of crowdsourcers. According the algorithm 1, we can reach the final matching result as show in the Fig.1(j). All the proposals of workers are accepted by the crowdsourcers. The utility of workers $1, 2, 3$ and 4 is $3, 2, 2, 0$ respectively. The utility of crowdsourcers A, B and C is $6, 7, 6$ respectively.

IV. Evaluation

Theorem 8. *The algorithm we proposed will eventually converge.*

Proof: As we can know from algorithm 1, if crowdsourcer j keeps rejecting worker $i's$ proposal, then the b_{ij} keeps decreasing. However, the cost of worker i is a fixed value, the worker i will finally stoping proposing to the crowdsourcer j. If crowdsourcer j accepts the worker $i's$ proposal, then the bid b_{ij} would not be change. Finally, the crowdsourcer j would accept the worker $i's$ proposal or worker i gives up proposing to crowdsourcer j. This is true for all workers and crowdsourcers. So eventually, each crowdsourcer will accept all workers who still propose to him. The proposed algorithm has converged.

Theorem 9. *The matching result of proposed algorithm satisfies the property of individual rationality.*

Proof: For every worker, he will choose to propose a task combination which maximizes his utility. If utility of the task combination is less than zero, worker would rather do nothing. So the utility of a worker is not less than zero.

For every crowdsourcer, if crowdsourcer j accepts a worker $i's$ proposal, the bid b_{ij} is no more than the peak price r_j. So the crowdsourcer $j's$ utility that worker i brings is no less zero, then the total utility of the crowdsourcer j is not less than zero. In summary, the matching result of the proposed algorithm satisfies the individual rationality.

Theorem 10. *The matching result of the proposed algorithm satisfies the property of nonwastefulness.*

Proof: We prove the nonwastefulness of proposed algorithm as follows: if there is a type II blocking pair (s, t), worker s prefers $\mu(s) \cup t$ at the $p_s \bigcup p_{st}$ to $\mu(s)$ at the p_s. Due to each worker will choose a combination which maximize his utility, so worker s must has proposed the crowdsourcer t at the p_s but was rejected. Let $\overline{\mu}$ denote the task assignment at the time worker s propose the task t, and crowdsourcer t is willing to add worker s under the p_s without violating the budget constraint and number constraint in the matching result. So $\overline{\mu}$ must not be the matching result. According the lemma 1, the worker i would be accepted by the crowdsourcer j in the matching result μ. That is contradictory, so worker i and crowdsourcer t cannot form a type II blocking pair.

Theorem 11. *The matching result of the proposed algorithm satisfies the property of Fairness.*

Proof: We prove the fairness of our proposed algorithm as follows: if there is a type I blocking pair(s,t), which means there is a worker $s'\epsilon\mu(t)$, both of worker s and s' want to complete task t, crowdsourcer t is willing to replace s' with worker s under the p_s. So the p_s less than $p_{s'}$. According the algorithm 1 and the second assumption, the crowdsourcer t will select a smaller bid, so it is impossible to choose s and give up s'. That is to say, the worker i and crowdsourcer t cannot form a type I blocking pair.

Theorem 12. *The proposed algorithm produces a stable task assignment.*

Proof:According the Theorem 9, Theorem 11 and theorem 12, the final task assignment is individual rational, fair and nonwasteful. So the matching result of our proposed algorithm is stable.

V. Related work

Combination spectrum auctions : The spectrum auctions are widely used for spectrum resource allocation. Spectrum is an indispensable resource for wireless communication, but due to it is a limited resource and strained for supporting the ever-increasing wireless traffic. The transaction of spectrum is essentially a matching of spectrum sellers and buyers. Yanjiao Chen proposed a spectrum matching framework for spectrum transactions where buyers have maximum quotas and minimum requirements in [7].

Task assignment of crowdsourcing : Crowdsourcing leverages the collective intelligence of the massive crowd workers to accomplish tasks in a cost-effective way. Especially, task assignment is one of most essential problems in crowdsourcing. In [8], Xiao et al. proposed an offline task assignment algorithm and an online task assignment algorithm based on a greedy algorithm.

Stable matching : Gale and Shapley first proposed the deferred acceptance algorithm to solve the college admission problem in [9], and it indicates that deferred acceptance algorithm can obtain a stable matching result. In [10], K.Hamada mainly considered the hospital-residents matching problem with quota lower bounds, and proposed an algorithm to obtain a stable matching.

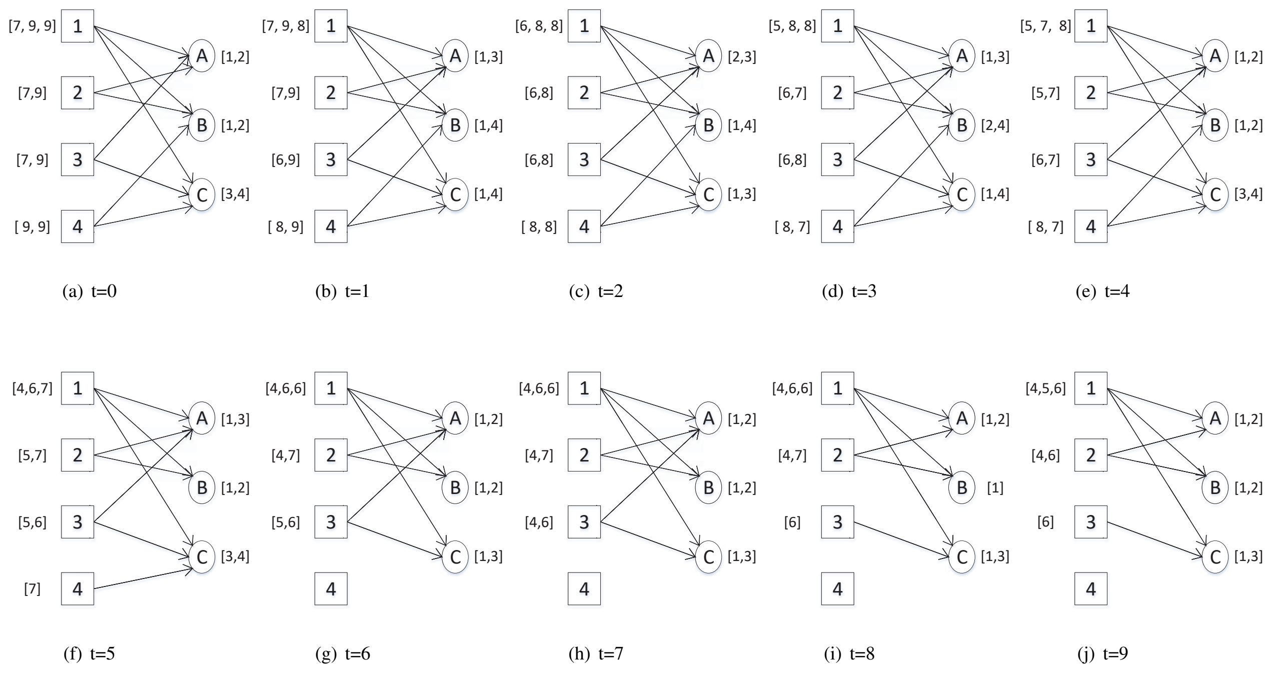

Fig. 1: An Illustrative Example

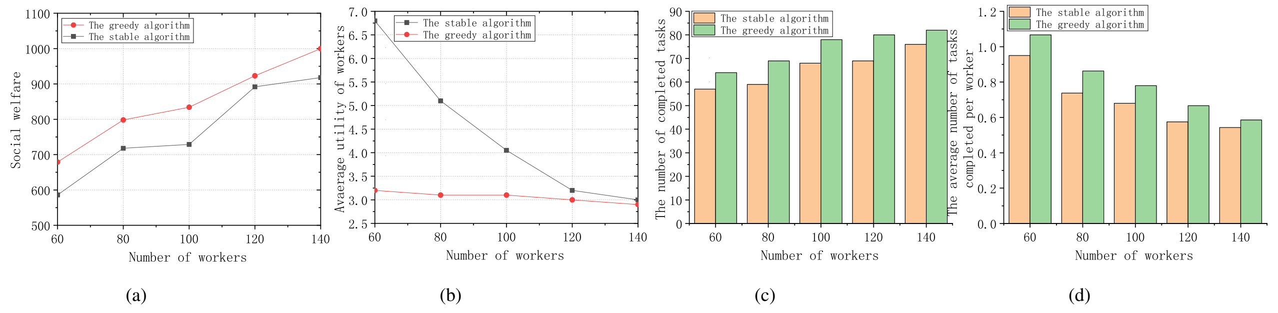

Fig. 2: The number of task types is fixed as 6, the number of worker changes from 60 to 140.

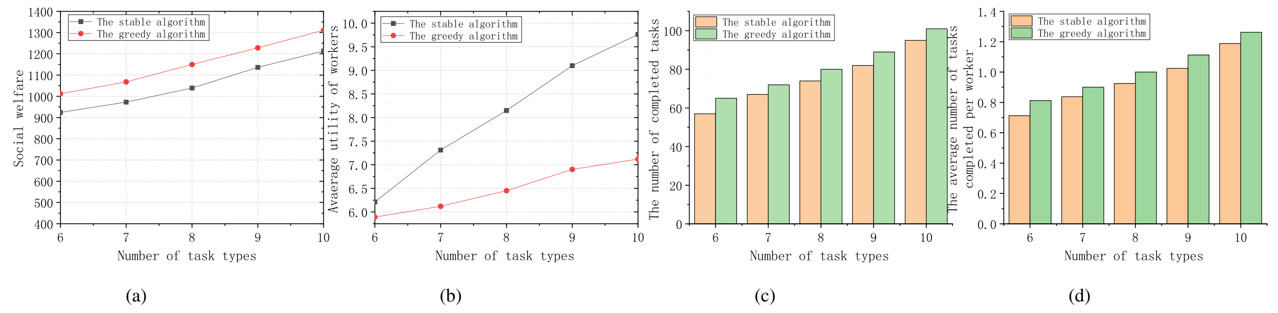

Fig. 3: The number of workers is fixed as 80, the number of types of task changes from 6 to 10.

VI. Simulation

In this section, we perform simulations to evaluate the performance of the proposed algorithm. We have four metrics for our algorithm's evaluation, and they are social welfare, average worker utility, the number of completed tasks and the average number of tasks completed by per worker. We also compare our mechanism with a greedy algorithm which is based on [11].

Greedy algorithm. In the greedy algorithm, workers will report their information truthfully, and they will propose one combination of tasks. Next, crowdsourcers will greedily choose workers according to worker's bids. Finally, the greedy algorithm will compute a critical value as the payment to the worker, and it also guarantees the honesty of workers.

Due to the number of possible task combinations grow exponentially with the number of task types, we set the number of task types from 6 to 10. A worker's cost of completing individual task is randomly chosen in the range $[1, 10]$, and the cost of a task combination is the sum of costs of completing the individual task in the combination minus random value in the range [0,8]. We set the default number of worker is 80, and the default number of task types is 6.

A. *Social welfare*

Social welfare is defined as the sum of the utility of workers and crowdsourcerws. As show in the Fig.2(a) and Fig.3(a), we compare the social welfare of the stability algorithm with the social welfare of the greedy algorithm. The goal of the greedy algorithm is to maximize social welfare, so greedy algorithm 's social welfare is a little bit higher than our proposed stable algorithm. It shows that some social welfare is sacrificed in exchange for a stable matching result. From the Fig.2(a), we can know as the number of workers increases, there will be more winning workers as the number of available workers increases, then the social welfare will also increase. From the Fig.3(a), we can know as the number of task types increases, workers are more likely to obtain their preferred tasks to complete. Then the social welfare will also increase.

B. *Average Worker utility*

A buyer's utility is calculated according the equation (1), the utility of the worker is the payment he gets minus his cost, we can get the average worker's utility by dividing the sum of the worker's utility by the number of workers. As show in the Fig.2(b) and Fig.3(b), we compare the average worker's utility of our algorithm and average worker's utility of the greedy algorithm. Individual worker's utility of our proposed stable algorithm is apparently higher than the individual worker's utility of greedy algorithm. As the number of workers increases, there will be more workers complete for the tasks, then the workers may be paid less and the workers may lose the qualification to complete the task. So the utility of workers will be smaller. As the number of type tasks increases in the stable algorithm, the utility of workers will also increases. However, the utility of workers in the greedy algorithm has not changed much.

C. *The number of completed tasks*

The number of completed tasks is the sum of the number of tasks completed by all workers. As show in the Fig.2(c) and Fig.3(c), as the number of workers increases, the number of completed tasks also increases, because there will more workers complete tasks. We also can see that, as the number of task types increases, then the number of completed tasks also increases in both of stable algorithm and greedy algorithm.

As for the average number of tasks completed by per worker, in the Fig.2(d) and Fig.3(d), as the number of workers and the number of types of tasks increases, the average number of tasks completed by per worker also increases in both of the stable algorithm and the greedy algorithm.

VII. Conclusion

In this paper, we not only consider the complementarity among different types of tasks in the crowdsourcing, but also investigate the stable matching in the task assignment problem. We consider the diverse preferences of individual workers and crowdsourcers towards each other and present a many-to-many matching framework with quantity constraint and budget constraint of crowdsourcing task. At the same time, we determine the corresponding payoff that crowdsourcers should pay. The algorithm we proposed in the paper can yield task assignment results that are individual rational, fair and nonwasteful, then we can say the task assignment result is stable. Experimental results show that our algorithm can achieve high utility and stable matching results.

VIII. acknowledgment

This research was supported by the National Natural Science Foundation of China (NSFC) (Grant No. 61872330, 61572457, 61379132, U1709217), the NSF of Jiangsu Province in China (Grant No. BK20131174, BK2009150), and Anhui Initiative in Quantum Information TechnologiesGrant No. AHY150300. The corresponding author of this paper is Mingjun Xiao.

References

[1] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 16(5):36–44, 2012.

[2] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[3] Jiangtao Wang, Leye Wang, Yasha Wang, Daqing Zhang, and Linghe Kong. Task allocation in mobile crowd sensing: State-of-the-art and future opportunities. *IEEE Internet of Things Journal*, 5(5):3747–3757, 2018.

[4] Jiangtao Wang, Yasha Wang, Daqing Zhang, Jorge Goncalves, Denzil Ferreira, Aku Visuri, and Sen Ma. Learning-assisted optimization in mobile crowd sensing: A survey. *IEEE Transactions on Industrial Informatics*, 15(1):15–22, 2019.

[5] Yanjiao Chen and Xiaoyan Yin. Stable job assignment for crowdsourcing. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[6] Daniel Fragiadakis, Atsushi Iwasaki, Peter Troyan, Suguru Ueda, and Makoto Yokoo. Strategyproof matching with minimum quotas. *ACM Transactions on Economics and Computation*, 4(1):6, 2016.

[7] Yanjiao Chen, Linshan Jiang, Haofan Cai, Jin Zhang, and Baochun Li. Spectrum matching. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 590–599. IEEE, 2016.

[8] Mingjun Xiao, Jie Wu, Liusheng Huang, Yunsheng Wang, and Cong Liu. Multi-task assignment for crowdsensing in mobile social networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2227–2235. IEEE, 2015.

[9] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[10] Koki Hamada, Kazuo Iwama, and Shuichi Miyazaki. The hospitals/residents problem with quota lower bounds. In *European Symposium on Algorithms*, pages 180–191. Springer, 2011.

[11] Yaron Singer. Budget feasible mechanisms. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 765–774. IEEE, 2010.