A Stable Task Assignment Scheme in Crowdsourcing

Xiao Chen
Department of Computer Science, Texas State University, San Marcos, TX 78666
Email: xc10@txstate.edu

***Abstract*—Crowdsourcing has become a popular business development strategy that outsources self-contained small tasks to a crowd of people to solve problems that an individual or an organization cannot easily do. There are two different task assignment types in crowdsourcing platforms: the worker selected task mode and the server assigned task mode. Right now, the crowdsourcing websites only use one of them and the satisfaction of the workers and the requesters are not fully addressed. Furthermore, it is not easy for the requesters to identify qualified workers quickly. In this paper, we propose a crowdsourcing model that considers the preferences of both the requesters and workers to improve their satisfaction and thereafter benefits the crowdsourcing platform. We first put forward a ranking formula for the requesters to identify qualified workers timely based on the Bayesian inference by considering two factors: the prices the workers charge and their online reviews, and then propose a stable task assignment algorithm STA that stably matches the workers and the tasks through the stable marriage approach. Simulation results show that our proposed task assignment approach greatly improves the satisfaction of the requesters and the workers compared with the existing Hungarian method and the STA variations that only consider one factor.**

***Index Terms*—Baysian inference, crowdsourcing, preference list, stable matching, task assignment**

I. Introduction

With the ever improving availability of social media platforms, the internet and mobile devices, crowdsourcing has become a popular business development strategy that outsources self-contained small tasks to a crowd of people to solve problems that an individual or an organization cannot easily do. Businesses use crowdsourcing to accomplish their tasks, find solutions to problems, or gather information. There are three basic components in crowdsourcing: requesters who publish tasks on a platform, workers who carry out the tasks, and a platform that manages jobs. There are many crowdsourcing platforms available today, such as Amazon Mechanical Turk [1] that outsources human intelligence tasks to the crowds, Fiverr [4] that allows workers to post tasks that they are willing to complete for a certain amount of money, Chegg Tutors [2] that finds tutors for the students online, and mobile crowd sensing apps such as Waze [5] and Weathermob [6] that rely on users' real-time mobile data to do navigation and find out the weather conditions, respectively.

In the crowdsourcing platforms, there are two different task assignment types: *worker selected task mode* and *server assigned task mode* [16]. In the worker selected task mode [9], [18], the server publishes the tasks and it is totally the workers' responsibility to choose any tasks that they are interested in. One drawback of this mode is that the server does not have any control over the allocation of the tasks. This may lead to some tasks never been assigned. On the other hand, in the server assigned task mode [8], [16], [22], the task is totally assigned by the server according to certain rules. The server has global pictures of the tasks and can achieve the global optimum of some objective functions.

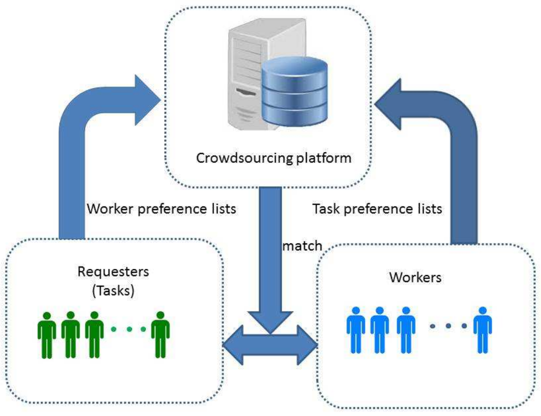


Fig. 1. Crowdsourcing model

Right now, the worker selected task mode and the server assigned task mode are totally separated, which causes the task assignment to be unilateral: either on the worker's side or on the server's side. And the goal of the assignment is usually to minimize or maximum some objective functions such as to minimize the cost of autonomous carsharing [13], or to maximize the number of location-related tasks assigned [17], etc. The satisfaction of the workers and the requesters of the tasks are not fully considered. Furthermore, crowdsourcing allows anyone to participate and the financial incentives cause workers to complete tasks quickly rather than well, allowing for many unqualified participants and resulting in large quantities of unusable contributions. Companies, or additional crowdworkers, then have to sort through all of these low-quality contributions. The task of sorting through crowdworkers contributions, along with the necessary job of managing the crowd, requires companies to hire actual employees, thereby increasing management overhead [3]. Verifying responses is time-consuming, so requesters often depend on having multiple workers complete the same task to correct errors. However, having each task completed multiple times increases time and monetary costs [15]. Therefore in this paper, we propose a crowdsourcing model that allows requesters to identify qualified workers and considers the interests of the workers and the requesters so as to improve the quality of the crowdsourcing platform.

On a crowdsourcing platform, the goal of the workers is to select those tasks that can maximize their rewards, enjoyment, and self-fulfillment while the interest of the requesters is to get the best workers to work on their tasks. If the workers and requesters are satisfied, more transactions can be done between them and the crowdsourcing platform can earn more profit. Based on these interests, we propose a crowdsourcing model as shown in Figure 1. On a crowdsourcing platform, each worker can submit his preference list of tasks and each requester of a task can also submit his preference list of workers to do the task. For the convenience of illustration, in the rest of the paper, we just call the preference list from a requester for a particular task 'the task's preference list' and the satisfaction of a requester on a particular task 'the task's satisfaction'. Then the objective of the platform is to stably match the workers and the tasks based on their preference lists to improve their satisfaction. Here 'stable' means that no worker will prefer another task than his matched one and no requester will prefer another worker than his assigned one. We are motivated to propose this model due to the following reasons: First, our model allows both the workers and the requesters to state their preferences so that the choice is mutual, not unilateral as in other papers. Second, we make the match between them stable so that their satisfaction scores can be improved, which can make them use the crowdsourcing platform more in the future.

To implement our crowdsourcing model, we adopt the following methods. First, for the workers, their selections of the tasks are usually subjective and based on their interests, skills, rewards, and self-fulfillment. The platform can adopt some recommender system [7] to suggest some tasks to them according to the interests in their profiles or their past finished tasks, but the best person to decide the eventual preference list is the worker himself. So we assume that it is the responsibility of each worker to provide a preference list of tasks. For the requesters, we will come up with a formula using the Beysian inference [10] to rank the workers by considering two factors: the workers' past evaluations and the prices they charge for the tasks. With the preference lists ready on both sides, we next propose an algorithm called STA to stably match the workers and the tasks based on their preference lists through the stable marriage approach [12].

The differences of our work from others and the key contributions of our work are as follows:

- We propose a crowdsourcing model that considers the interests of the requesters and the workers so as to improve the quality of the platform.
- We put forward a formula using the Beysian inference to help the requesters rank and identify qualified workers based on the considered factors.
- We propose a stable task assignment algorithm STA to match the requesters' tasks and the workers.
- Simulations are conducted to evaluate the proposed STA algorithm by comparing it with the existing method and the variations of STA.

The rest of the paper is organized as follows: Section II references the related works. Section III describes the model and defines the problem. Section IV presents the formula to rank the workers. Sections V matches the workers and the tasks. Section VI describes the simulations we have conducted, and the conclusion is in Section VII.

II. Related Works

Crowdsourcing covers a wide spectrum. Here we survey the related papers on task assignment in crowdsourcing.

There are two different task assignment types: worker selected task mode and server assigned task mode [16]. In the worker selected task mode [9], [18], the server publishes the tasks and it is totally the crowd workers' call to choose any tasks they are interested in. One drawback of this method is that the server does not have control over the allocation of tasks. This may lead to some tasks never been assigned while others do. Differently, in the server assigned task mode [8], [16], [22], the task is totally assigned by the server according to certain rules. This method has a global picture of the tasks so it can achieve the global optimum in terms of some objective functions. A typical example of the server assigned task mode is the Hungarian algorithm [20], which finds the minimum cost bipartite matching between the tasks and the workers. There are several other server assigned task algorithms in some specific applications. For example, in the car sharing industry, [13] provides a greedy method that assigns the passenger request to its geometrically nearest taxi. And in the spatial-crowdsourcing environment, [8], [23] assign tasks to workers according to their positions and then the workers will physically move to the specified locations to conduct tasks.

Right now, the worker selected task mode and the server assigned task mode are separated. The crowdsourcing platform uses one of them and the satisfaction of the workers and the requesters cannot be fully addressed. We argue that the satisfaction of the workers and the requesters (tasks) can be improved if we combine these two assignment modes by considering the preferences of both the tasks and the workers. We achieve that by adopting the idea of the Stable Marriage Problem (SMP) [19]. SMP aims to find a stable matching between two equally-sized sets of elements (i.e., men and women) given complete preference orders of each man and woman. Stability requires that a matched man and a matched women will not prefer each other over their existing partners. This is the task we do in this paper.

III. Model and Formulation

In our model, we assume that there are a set of workers $W = \{w_i\}$ looking for tasks and a set of tasks $T = \{t_j\}$ from the requesters. Each worker has submitted the price of each task and a list of preferred tasks as a result of recommendations from the platform and personal choice. We assume that the platform uses an evaluation system on the workers in the form of $X\%$ positive out of Y reviews. Different crowdsourcing websites [2], [4] may use different formats to evaluate the workers, but they can be converted to the format we adopt here. Our goal is to first propose a formula to help the requester who owns a task rank the preferred workers to work on the task and then match the workers with the tasks stably through the stable marriage approach.

IV. Ranking the workers

In this section, we work on the formula for the requesters to rank the workers they desire. We consider two factors: the price a worker charges to finish a task P_w and his past reviews by the requesters R_w. The price factor is straightforward. But the review part needs a little work. Nowadays, many crowdsourcing websites use a rating system for the workers. For example, worker A is 97% positive out of 1000 reviews and worker B is 98% positive out of 100 reviews. Then which worker is better? In term of the positive reviews, B is higher. But B gets much fewer reviews, which makes B's rating seem not as trustworthy as A's. So the number of reviews a worker gets matters. So to help a requester rank the workers based on their reviews, we need to design an aggregated scalar rating formula by including both the ratings and the number of reviews. We can start with the Beysian inference [10], which is a widely used and powerful tool.

In the Beysian inference, there is a prior and a posterior. In our case, if we know how people have rated the service of a worker, we can predict if the requester can get a satisfactory service from the worker in the future. So how people rate the worker is the prior and our prediction is the posterior in the Beysian inference. People's rating on a worker is a random variable that follows some distribution. We denote that as P and its distribution is unknown. In that case, an appropriate assumption is that it follows a beta distribution $\beta(a, b)$ [21] because beta distribution covers a broad range of distributions with the variations of its two parameters a and b. The PDF of the beta distribution is:

$$f(x) = Cx^{a-1}(1-x)^{b-1}, 0 < x < 1,$$

where the constant $C = \frac{(a+b-1)!}{(a-1)!(b-1)!}$.

Based on the current set of people's ratings X for a worker, e.g. 97% positive out of 1000 reviews, we observe that the likelihood function of the sample data follows a binomial distribution $B(n, p)$ [24], where n is the number of reviews and p is the probability that he gets a positive rating. By combining these factors, our problem is to solve the following:

We observe that $X|P \sim B(n, p)$, where the distribution of the prior $P \sim \beta(a, b)$. We want to find out the distribution of the posterior $P|X$.

Using the Bayesian rule, the PDF of the posterior can be written as:

$$f(p|X = k) = \frac{Pr(X = k|P)f(p)}{Pr(X = k)}$$

Here $Pr(X = k|P)$ is the PDF of the binomial distribution. It is equal to $\binom{n}{k}p^k(1-p)^{n-k}$. Item $f(p)$ is the PDF of the beta distribution. After plugging in the PDFs, the above formula becomes:

$$f(p|X = k) = \frac{\binom{n}{k}p^k(1-p)^{n-k}Cp^{a-1}(1-p)^{b-1}}{Pr(X = k)}$$

After ignoring all the items that are not related to p, we get:

$$\begin{aligned} f(p|X = k) &\propto p^{a+k-1}(1-p)^{b+n-k-1} \\ &\implies P|X \sim \beta(a + x, b + n - x) \end{aligned} \quad (1)$$

So the posterior is proportional to the items having p and is still a Beta distribution but with parameters $a + x$ and $b + n - x$. We interpret x as the new positive ratings on the basis of the original a positive ratings and $n - x$ as the new negative ratings on the basis of the original b negative ratings.

Let us explain this idea using the cases of workers A and B. First we need to determine the parameters a and b in the prior Beta distribution. In the beginning, we do not know how the requesters will rate the worker, so it is reasonable to assume that their ratings are uniformly distributed. That corresponds $a = b = 1$ in the Beta distribution. Thus we assume the prior has a distribution of $\beta(1, 1)$. Later we get some observed data: Worker A is 97% positive out of 1000 reviews and worker B is 98% positive out of 100 reviews. In other words, A has 970 new positive reviews and 30 new negative reviews and B has 98 new positive reviews and 2 new negative reviews. The posterior in expression (1) tells us that A and B's posterior distributions are $\beta(971, 31)$ and $\beta(99, 3)$, respectively. In Beta distribution $\beta(a, b)$, we know its mean and variance are as follows:

$$Mean = \frac{a}{a+b}, Variance = \frac{ab}{(a+b)^2(a+b+1)}$$

In our case, a good rating is a rating with high mean and low variance. So to obtain the scalar rating R_w for a worker w, we can simply use the ratio of the mean and the variance. That is,

$$R_w = \frac{Mean}{Variance} \quad (2)$$

After calculation, A's rating is: 32420 and B's rating is: 3502. For the value of R_w, the bigger the better. So A should be placed before B considering the number of reviews even though A's positive rate is lower than B's.

Now that we have a formula to rank the workers by the reviews, we add in the price factor and derive a combined formula. Since the price for a task charged by a worker P_w is known, we can use a weighted score formula in (3) to incorporate both factors.

$$S_w = g_1\widetilde{R_w} + g_2\widetilde{P_w} \tag{3}$$

The parameters g_1 and g_2 are the weights in the range of $[0,1]$ for the two factors and $g_1 + g_2 = 1$. The $\widetilde{R_w}$ and $\widetilde{P_w}$ are the processed values for R_w and P_w, respectively. The values of R_w and P_w need to be processed first because not only are they on different scales but also counted in opposite directions in the final score. As we know, for the value of R_w, the bigger the better. But for the value of P_w, the smaller the better. So for P_w, we first take $1/P_w$ to make it count in the same direction as R_w. Then since these two values are on different scales, we normalize them to the range of $[-1,1]$ using formula (4) before putting them together in formula (3). In formula (4), V is a vector which here represents either a price vector containing the reciprocals of the prices charged by a worker or his review vector containing the calculated reviews from formula (2). Notation V_i is the i's element in the vector and V_i' is its normalized value. Notations $mean(V)$, $max(V)$, and $min(V)$ represent the mean of all the elements, the maximum value, and the minimum value in the vector, respectively.

$$V_i' = \frac{V_i - mean(V)}{max(V) - min(V)} \tag{4}$$

In this paper, we consider two factors for the requesters to rank the workers, namely the price factor and the review factor. But formula (3) can be easily extended based on the needs of applications to include more parameters the requesters like to consider by simply adding the parameters and their corresponding weights.

V. Matching Tasks and Workers

In this section, we match tasks and workers according to their preferences so as to improve their satisfaction. We assume that there are n workers and n tasks. Each worker has a preference list of tasks and each task has a preference list of workers. To match these tasks and workers, we adopt the idea of stable marriage approach [11] and give a definition as follows.

Definition 1: A matching between tasks and workers is stable if there does not exist any task or worker which prefers each other more than their current match.

Suppose a worker w_i is assigned a task t_j' and a task t_j is given to a worker w_i', the matching is not stable if w_i prefers task t_j more than t_j' and worst yet, task t_j also prefers worker w_i more than worker w_i'. In our approach, after each worker submits his preferred list of tasks to the crowdsourcing platform and each requester of a task lists his preferred workers according to formula (2), we provide a stable task assignment (STA) algorithm in Figure 2 matching the workers and the tasks.

There are three parts in this algorithm. In the main control part, each task t_j calls Subroutine Proposal to get the next preferred worker on its list. After the loop is done, the matching A between the tasks and the workers is returned. In the Subroutine Proposal, we match t_j with its next preferred

Algorithm STA: Stable Task Assignment

```
1: Inputs: a set of tasks $T$, and a set of workers $W$.
2: Output: a stable task assignment $A$ matching a worker
   with a task.
3: Each worker $i \in W$ submits his task preference list to
   the crowdsourcing platform.
4: Each requester generates his worker preference list of
   each task according to formula (3).
5: Main control
6: Initialize each worker as unassigned.
7: for each task $t_j \in T$ do
8:     call Subroutine Proposal for $t_j$.
9: end for
10: return $A$ as the stable task assignment.
11: Subroutine: Proposal
12: Input: task $t_j$.
13: if $w_i$ is the next entry in $t_j$'s preference order then
14:     assign $t_j$ to $w_i$.
15:     call Subroutine Refusal for $A$, $t_j$, and $w_i$.
16: end if
17: Subroutine: Refusal
18: Input: assignment $A$, task $t_j$, and worker $w_i$.
19: if $w_i$ is assigned $t_j'$ but prefers $t_j$ over $t_j'$ then
20:     break up $w_i$ and $t_j'$ and assign $w_i$ to $t_j$; update $A$.
21:     reassign $t_j'$ by calling Subroutine Proposal.
22: else
23:     $t_j'$ and $w_i$ remain matched
24: end if
```

Fig. 2. The stable task assignment (STA) algorithm

worker w_i and then check if w_i will refuse this proposal or not by calling Subroutine Refusal. In the Subroutine refusal, the temporarily matched worker w_i can refuse the match. If w_i is matched to t_j' but prefers t_j over t_j', then we break up w_i and t_j', match t_j and w_i, and let t_j' go back to the market by calling Subroutine Proposal. Otherwise, t_j' and w_i remain matched.

We now use an example in Figure 3 to explain the stable task assignment algorithm. There are four tasks and four workers. Their preference lists are shown in the figure. For example, for task t_1, its preferred workers are in the order of w_4, w_1, w_2, and w_3. And for worker w_1, its preferred tasks are in the order of t_4, t_1, t_3, and t_2. We start from the Main control of the STA algorithm. Task t_1 first calls subroutine Proposal to propose to its first preference w_4. Then task t_1 and worker w_4 are temporarily matched. Then subroutine Refusal is called to see if worker w_4 would refuse the assignment. In Refusal, since w_4 is not assigned to any task yet, task t_1 and worker w_4 will remain matched. Next, in the Main control, task t_2 calls Proposal to propose to its first preference w_2. They are temporarily matched. Then Refusal is called to see if worker w_2 would refuse the proposal. Since w_2 is not matched to any task yet, t_2 and w_2 remain matched. Then, in the Main Control, task t_3 calls Proposal to propose to its first preference w_2. They are temporarily matched (see Figure 3(a)). Then

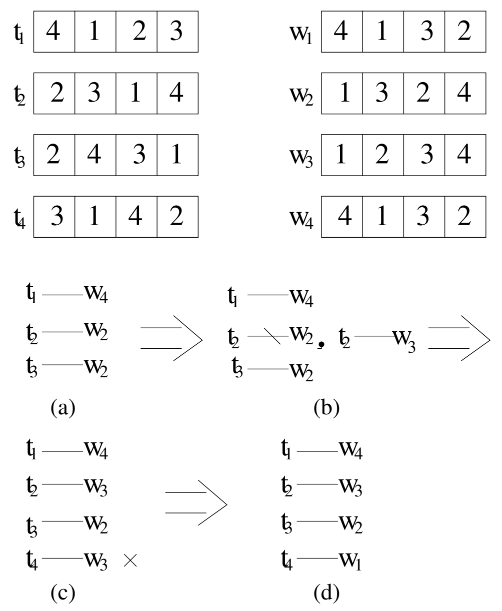


Fig. 3. An example of the STA algorithm

Refusal is called to see if w_2 would refuse the proposal. In Refusal, w_2 is already matched to t_2 but w_2 prefers t_3 more than t_2 according to its preference list. So t_2 and w_2 break up and w_2 is assigned to t_3. Now task t_2 is back on the market and calls Proposal to propose to its 2nd preference w_3. Task t_2 and w_3 are temporarily matched. Then Refusal is called to see if w_3 would refuse the match. In Refusal, since w_3 is not matched to any task, t_2 and w_3 remain matched (see Figure 3(b)). Next, in the Main control, task t_4 proposes to its first preference w_3. They are temporarily matched. Then Refusal is called to see if w_3 would refuse the match. In Refusal, since w_3 is already matched to t_2 and prefers t_2 over t_4, so w_3 is still matched to t_2 and task t_4 does not get w_3 (see Figure 3(c)). Then task t_4 proposes to its 2nd preference w_1. And since w_1 is not matched to any task, t_4 and w_1 remain matched. By now, all the tasks have obtained the matched workers and the algorithm terminates. Finally, tasks t_1, t_2, t_3, and t_4 get workers w_4, w_3, w_2, and w_1, respectively (see Figure 3(d)).

After applying the STA algorithm, all the workers and the tasks are stably matched. But there is a question: what if a worker does not like any of the tasks or a task does not like any of the workers? In that case, we allow them to insert a dummy in their preference lists. If finally someone is matched to a dummy, then he does not get a match this time and can wait for a better partner in the future. Furthermore, dummy relaxes the condition that the two matching sets should have the same size.

VI. Simulations

In this section, we evaluate the performance of our proposed STA algorithm with its variations and the existing algorithm using a customized simulator written in Matlab.

A. Algorithms Compared

The following algorithms are compared:

1) *The stable task assignment algorithm* (STA): our proposed algorithm based on stable matching and the consideration of both workers' reviews and their charging prices.
2) *The price-only algorithm* (Price-only): a variation of the STA algorithm based on stable matching and the consideration of only the workers' charging prices.
3) *The review-only algorithm* (Review-only): a variation of the STA algorithm based on stable matching and the consideration of only the workers' reviews.
4) *The Hungarian algorithm* (Hungarian): the existing Hungarian algorithm which achieves the lowest cost of assigning tasks but does not consider the requesters' and workers' satisfaction.

B. Comparison Metric

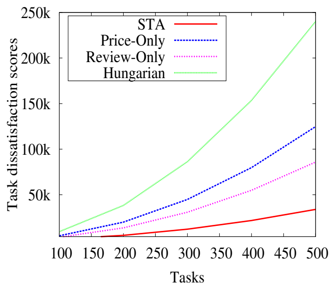
To compare the above algorithms, we define a metric called *dissatisfaction score*. It is calculated as follows. For a task t_j, suppose worker w_i's position in its preference list is k. If finally the task is matched to worker w_i', which is in the k'th position in its preference list. We define t_j's dissatisfaction score $D_j = k' - k$. If D_j is positive, the task is assigned to a worker it is less in favor of. If D_j is zero, the task is assigned to a worker exactly as it expects in its preference list. A negative value is not possible due to the nature of the stable matching problem. Then for the whole match, the task dissatisfaction score of a match is $\sum_{j=1}^{n} D_j$. Similarly, the worker dissatisfaction score of a match is defined in the same way. For the dissatisfaction scores, the smaller the better.

C. Simulation Setup

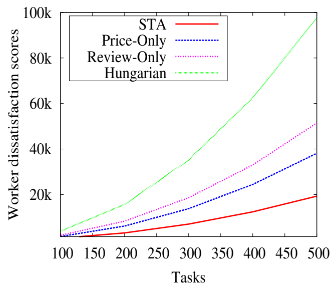
In our simulations, we randomly generated a set of tasks T and a set of workers W. For each worker, we randomly generated his task preference list and the associated price for each task in the range of $[1, 100]$. We also randomly generated each worker's evaluation scores in the range of $[10\%, 100\%]$ and the number of reviews in the range of $[1, 1000]$. For the STA algorithm, we used formula (3) to calculate a worker's preference list for each task by considering both the price and the review factors. For the Price-only and Review-only algorithms, we only included one factor in the calculation. Once the preference lists were ready, we applied algorithms STA, Price-only, and Review-only to match the workers with the tasks. For the Hungarian algorithm, we did not use the preference lists but just assigned tasks according to the prices of the tasks. We tried 100 to 500 tasks. After the matching results were obtained, we calculated the task and worker dissatisfaction scores. We ran each algorithm 1000 times and averaged the results.

D. Simulation Results

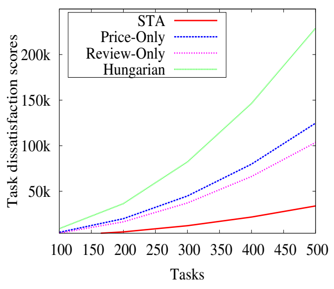
The simulation results are shown in Figures 4(a)-(f). We tried three weight values $g_1 = 0.2, 0.5, 0.7$ in the experiments. A weight of 0.2 means that the weight of the review factor



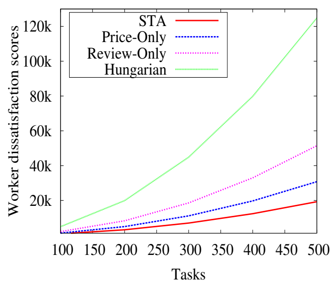
(a) Task dissatisfaction ($g_1 = 0.2$)



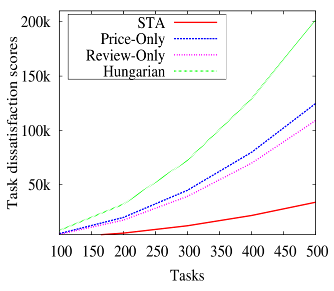
(b) Worker dissatisfaction ($g_1 = 0.2$)



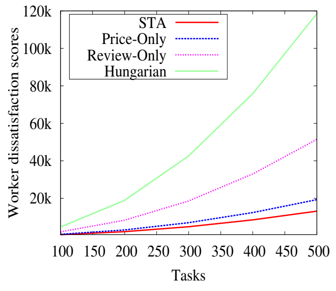
(c) Task dissatisfaction ($g_1 = 0.5$)



(d) Worker dissatisfaction ($g_1 = 0.5$)



(e) Task dissatisfaction ($g_1 = 0.7$)



(f) Worker dissatisfaction ($g_1 = 0.7$)

Fig. 4. Comparison of the algorithms with different weights

g_1 is 20% and the weight of the price factor $g_2 = 1 - g_1$ is 80%. For each weight, we obtained the task and worker dissatisfaction scores. Unanimously, the methods based on stable matching have substantial lower dissatisfaction scores than the Hungarian method in all of the parameter settings. In addition, the STA algorithm which considers both the price and the review factors has a lower dissatisfaction score than the Price-only and the Review-only algorithms which only consider one of the factors. In conclusion, the stable matching method can reduce the task and worker dissatisfaction scores compared with the existing Hungarian method and the tasks and workers can be more satisfied if both the price and the review factors are considered.

VII. Conclusion

In this paper, we have proposed a crowdsourcing model that considers the preferences of both the requesters and the workers to improve their satisfaction so as to benefit the crowdsourcing platform. We have first put forward a formula based on Bayesian inference for the requesters to identify qualified workers quickly by considering the price and the review factors. And then we have proposed a stable task assignment algorithm STA to stably match the tasks and the workers through the stable marriage approach. Simulation results have shown that our proposed scheme STA greatly improves the satisfaction of the requesters and the workers compared with the ones which only consider one of the factors and the existing Hungarian method which does not consider the requesters' and workers' satisfaction. In the future, we will explore more factors in the stable task assignment method.

Acknowledgments

This research was supported in part by NSF under CNS1305302 and ACI1440637.

References

[1] Amazon mechanical turk. http://mturk.com.
[2] Chegg Tutors. https://www.chegg.com/tutors/become-a-tutor/?from _header=1.
[3] Crowdsourcing. https://en.wikipedia.org/wiki/Crowdsourcing#Crowd sourcers.
[4] Fiverr. https://en.wikipedia.org/wiki/Fiverr.
[5] Waze. https://en.wikipedia.org/wiki/Waze.
[6] Weathermob-Social Weather Reporting, and Local and Global Weather Reports. https://itunes.apple.com/us/app/weathermob-social-weather-reporting-local-global-weather/id463729367?mt=8.
[7] E. Aldhahri, V. Shandilya, and S. Shiva. Towards an effective crowdsourcing recommendation system: A survey of the state-of-the-art. In *IEEE on SOSE*, 2015.
[8] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao. Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Transaction on Knowledge and Data Engineering*, 2015.
[9] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *21st SIGSPATIAL GIS*, pages 314–323, 2013.
[10] J-M Ernardo. *Reference analysis*, volume 25. Handbook of statistics, 2005.
[11] D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *American Mathematical Monthly*, 69:9–14, 1962.
[12] Y. A. Gonczarowski, No. Nisan, R. Ostrovsky, and W. Rosenbaum. A stable marriage requires communication. In *ACM-SIAM SODA*, 2015.
[13] J. P. Hanna, M. Albert, D. Chen, and P. Stone. Minimum cost matching for autonomous carsharing. In *IFAC-PapersOnLine*, 2016.
[14] C. J. Ho and J. W. Vaughan. Online Task Assignment in Crowdsourcing Markets. *AAAI*, 12:45–51, 2012.
[15] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *ACM HCOMP*, 2010.
[16] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *21st SIGSPATIAL GIS*, pages 189–198, 2012.
[17] L. Kazemi, C. Shahabi, and L. Chen. GeoTruCrowd: Trustworthy Query Answering with Spatial Crowdsourcing. In *ACM SIGSPATIAL*, 2013.
[18] Y. Li, M. L. Yiu, and W. Xu. Oriented online route recommendation for spatial crowdsourcing task workers. *Advances in Spatial and Temporal Databases*, pages 137–156, 2015.
[19] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 2002.
[20] J. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5:32–38, March 1957.
[21] K. Pearson. Mathematical contributions to the theory of evolution, XIX: Second supplement to a memoir on skew variation. *Philosophical Transactions of the Royal Society A*, 216(538-548):429–457, 1916.
[22] H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framewor. *ACM Transactions on Spatial Algorithms and Systems*, 1, 2015.
[23] H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems*, 1, 2015.
[24] G. P. Wadsworth. *Introduction to Probability and Random Variables*. New York: McGraw-Hill, 1960.