A Payment Scheme in Crowdsourcing

Xiao Chen[1], Kaiqi Xiong[2]
[1]Department of Computer Science, Texas State University, San Marcos, TX 78666
[2]Florida CyberSecurity Center and College of Arts and Sciences, University of South Florida, Tampa, FL 33620
Email: xc10@txstate.edu, xiongk@usf.edu

***Abstract*—Crowdsourcing coordinates a large group of workers online to do self-contained small tasks that are published by job requesters on a crowdsourcing platform. Many papers propose incentive strategies to motivate workers to participate in crowdsourcing. In this paper, we shift the focus from the workers to the job requesters by addressing two of their issues: how to design a good payment scheme to maximize profit and how to select qualified workers to do the job. We use a widely-adopted payment formula consisting of a base salary and extra bonus. We first formulate the problem as an optimization problem and then provide a general solution in which we show that the pay rate can be the same to all the workers. Next we instantiate the solution with a concrete example to derive more concrete results and propose a worker selection algorithm WS. In WS, we not only consider workers' workload demands but also their past working performance to guarantee crowdsourcing quality. Simulation results show that a job requester can pay much less to get the job done in a crowdsourcing environment and our worker selection algorithm is efficient in that it only searches a tiny space to find the solution to the optimization problem. Our effort here provides an evidence to support the benefits of using crowdsourcing in our daily lives.**

***Index Terms*—crowdsourcing, optimization, pay rate, workload**

I. Introduction

Crowdsourcing coordinates a crowd (a large group of people online) to do self-contained micro-work (small tasks) that solves problems that software or one user cannot easily do. Businesses use crowdsourcing to accomplish their tasks, find solutions to problems, or gather information. These include the ability to offload peak demand, access cheap labor and information, generate better results, access a wider array of talent than might be present in one organization, and undertake problems that would have been too difficult to solve internally [15]. Crowdsourcing is widely used in voting, information sharing, game, creative systems [19], and mobile crowd sensing (MCS) [7]. MCS is a new paradigm enabled by the widespread availability of smart phones equipped with a rich set of built-in sensors for collecting and sharing sensing data from surrounding environment over a large geographical region.

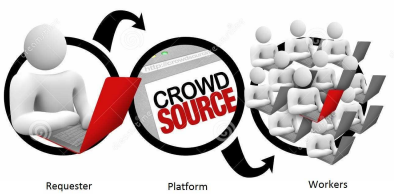


Fig. 1. Crowdsourcing components

There are three basic components in crowdsourcing as shown in Fig. 1: requesters who publish tasks on a platform, workers who carry out the tasks, and a platform such as Amazon Mechanical Turk [1] that manages jobs.

There are two classes of crowdsourcing applications: *participatory* crowdsourcing [16], [18] where participants are actively involved, and *opportunistic* crowdsourcing [5] where tasks are done automatically by mobile devices with minimal worker intervention. Malone *et al.* [14] define two variations of worker contribution: *collection* and *collaboration*. A collection occurs when different members of a crowd contribute independently of each other and a collaboration occurs when there exist dependencies between the contributions of a set of crowd workers such as when they work on different parts of a problem or improve a solution iteratively. In this paper, we address the class of participatory crowdsourcing and assume the worker's contribution is independent.

One important element to make crowdsourcing practical is how to motivate workers to contribute to the tasks, which has been extensively studied [6], [9], [17], [18], [20]. In this paper, we shift gears to look at the problems on the side of the requester which have not been discussed as much to supplement crowdsourcing research. We will address two issues of the job requester. One is how to design a good payment scheme to maximize profit and the other is how to select qualified workers based on the payment scheme. In our problem, we assume that the workers are motivated by the money paid for their work so their incentives are not a major issue here. In our model, both the requester and workers are rational. The requester wants to design a good payment scheme to maximize his profit and the workers will choose the corresponding workload to maximize their happiness which is defined as their income less the cost. Therefore, in our problem, the payment scheme is the driving force.

Different from the existing crowdsourcing payment schemes which pay a certain amount of money for each task [1], we use a payment formula $b + kx$ widely used in our daily lives. It is composed of two parts: a base salary b for base workload x_0 and an extra payment kx based on the pay rate k and extra workload x. That is, a worker will get an income of b if he finishes the base workload x_0 and an extra payment of kx if he takes extra x tasks. To help the requester decide b and k, we formulate this problem as an optimization problem on the requester's side to maximize his profit. And once the payment scheme is decided, the workers will react accordingly to choose their optimal workload demand to maximize their happiness. Correspondingly, the requester realizes this and will factor in the workers' reaction to find out the values for b and

k. To solve the optimization problem, we first give a general solution leading to the conclusion that the pay rate k can be the same to all the workers. Then we instantiate the general solution with a concrete example so that more concrete results can be derived.

Once the payment scheme has been decided, a worker selection algorithm can be designed accordingly. So far, very few worker selection algorithms for crowdsourcing have considered the issue of quality control. Quality control is a central issue for crowdsourcing because with the increase of workload and tasks' size, the quality of the work provided by workers raises many concerns as platforms are continuously challenged by workers with insufficient skillsets to solve a given problem [11]. In our proposed worker selection algorithm named *WS*, we take the workers' past working performance into account to guarantee the quality of the work to be done. The workers' previous working statistics can be obtained from the crowdsourcing platform as many crowdsourcing services on the web require workers to meet certain requirements before they can participate in a job [8]. For example, sites such as Amazon Mechanical Turk and GURU.com maintain detailed statistics tracking the performance of the workers. The former evaluates workers based on whether their work was accepted or rejected by job requesters [1] and the later keeps the technical skill, creativity, timeliness, and communication capabilities of a worker through a star-based rating system [2].

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

- We study the payment scheme design and worker selection algorithm for job requesters using a payment formula that has not been discussed in crowdsourcing.
- We formulate the payment scheme design problem as an optimization problem and provide a general solution to show that the pay rate can be the same for all the workers.
- We instantiate the solution framework with a concrete example and then derive a specific solution and thereafter a worker selection algorithm.
- We consider workers' previous working statistics in worker selection to guarantee crowdsourcing quality.
- We conduct simulations to show that the job requester can pay much less to get the job done in a crowdsourcing environment and our worker selection algorithm searches a tiny space to find a solution to the optimization problem.
- We provide an evidence through our theoretical analysis and simulations to justify the benefits of using crowdsourcing in our daily lives.

The rest of the paper is organized as follows: Section II references the related works. Section III gives preliminary information. Section IV defines the problem. Sections V and VI present a general solution and a concrete example, respectively. Section VII describes the simulations we have conducted, and the conclusion is in Section VIII.

II. Related Works

Crowdsourcing covers a wide spectrum. Here we survey the papers on incentive mechanisms in crowdsourcing.

Many existing incentive algorithms use auctions to model rewards. The work in [16] proposes ProMoT, a Profit Maximizing Truthful auction mechanism for mobile crowdsourcing system aiming to provide satisfying rewards to the smartphone users. Feng *et al.* propose TRAC [6] to stimulate smartphone users to join mobile crowdsourcing with a truthful auction mechanism that takes location information into consideration. And the works presented in [17], [20] propose auction mechanisms for mobile crowdsourcing, when a limited budget is assigned for sensing tasks and the platform performs a subset of tasks according to its budget constraint.

Some works provide monetary rewards to generally cooperative users [9], [10]. In [9], a subset of users are greedily selected according to their locations subject to the coverage and budget constraints. In [10], the authors design and evaluate a reverse auction-based dynamic price incentive mechanism focusing on minimizing and stabilizing incentive cost while maintaining adequate number of participants to contribute to the task. Different from these multi-winner mechanisms, Luo *et al.* in [12], [13] propose a winner-take-all mechanism, where a single best or designated user gets all the rewards. Chen *et al.* in [4] design incentive schemes by considering not only the extrinsic rewards such as money but also the intrinsic rewards such as a sense of satisfaction, social status, or honor.

Our work here is from a distinct perspective. We address the issues of a job requester to design a good payment scheme using a widely-adopted payment formula that has not been discussed before in crowdsourcing and then select qualified workers to work on a job accordingly.

III. Preliminary

We formulate the crowdsourcing optimization problem as follows: Suppose a job requester has a job containing W independent tasks and he wants to crowdsource it to N workers. There are two parties in this problem. On the job requester's side, he wants to maximize the profit of the job and on the workers' side, they want to maximize their happiness.

On the requester's side: Let us assume that each task of the job can bring in a revenue of P. Then the total revenue from this job is PW. The job requester needs to pay the workers to do the job. He decides to use the widely-used payment formula $b_i^t + k_i^t \cdot x_i^t$, in which the subscript i in each variable denotes worker i and the superscript t is the specified time period. The first part b_i^t represents the base salary for worker i at time period t if he finishes the base workload x_0^t and the second part is the extra payment with a pay rate of k_i^t if the worker takes x_i^t extra tasks over the base workload. In the payment formula, all the three variables b_i^t, k_i^t, and x_i^t are related to both time t and individual worker i. The first two variables b_i^t, k_i^t are controlled by the requester and the third variable x_i^t is chosen by worker i. The total cost of the job is the summation of the payments over time and individual workers, which is $\sum_t \sum_i (b_i^t + k_i^t \cdot x_i^t)$. Now, the goal of the job requester is to maximize the total profit by solving the following maximization problem for b_i^t and k_i^t.

$$\underset{b_i^t,k_i^t}{\text{maximize}} \quad PW - (\sum_t \sum_i (b_i^t + k_i^t \cdot x_i^t)) \tag{1}$$

Here, PW is a constant. So the problem is equivalent to minimizing the second part of the objective, which is

$$\underset{b_i^t,k_i^t}{\text{minimize}} \quad \sum_t \sum_i (b_i^t + k_i^t \cdot x_i^t) \tag{2}$$

On the worker's side: For each individual worker, his happiness is the income less the cost. Suppose worker i takes a load of $x_0^t + x_i^t$, where x_0^t is the base workload to receive the base income b_i^t and x_i^t is the extra load to earn the extra payment $k_i^t \cdot x_i^t$. Then the income of worker i at time t is $b_i^t + k_i^t \cdot x_i^t$. Obviously, x_i^t can be $zero$, meaning the worker does not want to take any extra load. The cost of a worker can be any expenditure on the worker's side to do the work. For example, it can be the energy consumed by the worker's device, or fatigue felt by the worker. The more workload the worker takes, the more the income but also the more the cost. Thus, the cost function of worker i is defined as $\alpha_i^t f_i(x_i^t)$, where f_i is an increasing function mainly related to the extra workload x_i^t worker i is willing to pick up and the added factor α_i^t makes the cost function to be dependent on time but allows the shape of f_i to stay the same over time. Now for each worker i, he wants to maximize his happiness by maximizing the following through picking the optimal amount of extra work x_i^t.

$$\underset{x_i^t}{\text{maxmize}} \quad b_i^t + k_i^t \cdot x_i^t - \alpha_i^t f_i(x_i^t) \tag{3}$$

IV. Problem Formulation

In this section, we analyze the conditions, make the requirements more explicit, and formulate the problem.

Let's first look at the workers' side. For each worker, he wants to maximize the expression in (3). To worker i, b_i^t and k_i^t are decided by the job requester, and α_i^t is a known variable. So the only unknown variable is x_i^t. To maximize the expression in (3), we can differentiate the expression with respect to x_i^t and set the result equal to zero. Thus, we have

$$k_i^t = \alpha_i^t f_i'(x_i^t)$$

So the optimal extra workload x_i^t to maximize worker i's happiness is:

$$x_i^t = f_i'^{-1}(k_i^t/\alpha_i^t) \tag{4}$$

Also, each worker wants his happiness to be greater or equal to zero. Thus, the expression in (3) should be greater or equal to zero. Furthermore, the total number of tasks including the base workload and extra taken by all the workers at any time t should be less or equal to the total number of tasks W.

Now to put the requester's objective and all of the constraints together, we can formulate the optimization problem with variables b_i^t, k_i^t, x_i^t as follows:

$$\begin{aligned} \underset{b_i^t,k_i^t,x_i^t}{\text{minimize}} \quad & \sum_t \sum_i (b_i^t + k_i^t \cdot x_i^t) \\ \text{subject to} \quad & \sum_i x_i^t + \sum_i x_0^t \le W, \ \forall t \\ & b_i^t + k_i^t \cdot x_i^t - \alpha_i^t f_i(x_i^t) \ge 0, \ \forall t, i \\ & x_i^t = f_i'^{-1}(k_i^t/\alpha_i^t), \ \forall t, i \end{aligned} \tag{5}$$

Here, the variables b_i^t and k_i^t are decided by the requester and x_i^t is the reaction from worker i to the payment scheme.

V. A General Solution

In this section, we give a general solution to problem (5).

First, in problem (5), $b_i^t + k_i^t \cdot x_i^t$ can be minimized if a worker's happiness is driven to zero, that is, $b_i^t + k_i^t \cdot x_i^t - \alpha_i^t f_i(x_i^t) = 0$. In that case, a worker does not feel very happy but he is still motivated to do the work because of income. Then $b_i^t + k_i^t \cdot x_i^t = \alpha_i^t f_i(x_i^t)$, which can be realized by any combination of b_i^t and k_i^t that can support an extra workload of x_i^t. Thus,

$$b_i^t = \alpha_i^t f_i(x_i^t) - k_i^t \cdot x_i^t \tag{6}$$

Next, we denote $f_i'^{-1}(k_i^t/\alpha_i^t)$ as $d_i^t(k_i^t/\alpha_i^t)$, where function d represents the demand function of worker i corresponding to pay rate k_i^t. From Equation (4), we have

$$x_i^t = f_i'^{-1}(k_i^t/\alpha_i^t) = d_i^t(k_i^t/\alpha_i^t) \tag{7}$$

Now problem (5) is reduced to an optimization problem with only one variable k_i^t:

$$\begin{aligned} \underset{k_i^t}{\text{minimize}} \quad & \sum_t \sum_i (\alpha_i^t f_i(d_i^t(k_i^t/\alpha_i^t))) \\ \text{subject to} \quad & \sum_i d_i^t(k_i^t/\alpha_i^t) + \sum_i x_0^t \le W, \quad \forall t \end{aligned} \tag{8}$$

This optimization problem can be dealt with using the Lagrange duality method [3]. We define the *Lagrangian* $\mathcal{L}(k_i^t, \lambda)$ associated with the problem as

$$\begin{aligned} \mathcal{L}(k_i^t, \lambda) &= \sum_t \sum_i (\alpha_i^t f_i(d_i^t(k_i^t/\alpha_i^t))) + \lambda(\sum_i d_i^t(k_i^t/\alpha_i^t) \\ &+ \sum_i x_0^t - W) \end{aligned}$$

The parameter λ is the Lagrange multiplier associated with the inequality constraint $\sum_i d_i^t(k_i^t/\alpha_i^t) \le W$.

We define the *Lagrange dual function* g as the minimum value of the Lagrangian over k_i^t:

$$\begin{aligned} g(\lambda) &= \inf_{k_i^t} \mathcal{L}(k_i^t, \lambda) \\ &= \inf_{k_i^t} (\sum_t \sum_i (\alpha_i^t f_i(d_i^t(k_i^t/\alpha_i^t))) \\ &+ \lambda(\sum_i d_i^t(k_i^t/\alpha_i^t) + \sum_i x_0^t - W)) \end{aligned}$$

The above function $g(\lambda)$ is the infimum of an affine function in terms of λ, so it is concave. According to the lower bound property [3], if $\lambda \ge 0$, then $g(\lambda) \le p^*$, where p^* is the optimal value of the original problem (8). That is, the Lagrange dual

function that depends on λ gives us a lower bound on the optimal value p^* of problem (8). Now we can work on the Lagrange dual problem that gives the best lower bound of p^*. That is,

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & g(\lambda) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{9}$$

This is a convex optimization problem, since the objective $g(\lambda)$ to be maximized is concave and the constraint is convex. So to maximize the objective, we differentiate $g(\lambda)$ with respect to λ and set the result to zero. Thus, we have

$$\sum_i d_i^t(k_i^t/\alpha_i^t) + \sum_i x_0^t = W \tag{10}$$

In Equation (10), d_i^t is a demand function of worker i at time t corresponding to the k_i^t set by the job requster, and α_i^t is a known variable. For k_i^t, it suffices for the job requester to set it the same to all the workers as long as the summation of all the workload taken by the workers is equal to W. Thus, we can drop i and make

$$k_i^t = k^t \tag{11}$$

Now the pay rate k^t is only related to time t and not each individual worker i any more. After k^t is set, $b_i^t = \alpha_i^t f_i(x_i^t) - k_i^t \cdot x_i^t$ from Equation (6). Since $x_i^t = d_i^t(k_i^t/\alpha_i^t)$,

$$b_i^t = \alpha_i^t f_i(d_i^t(k^t/\alpha_i^t)) - k^t \cdot d_i^t(k^t/\alpha_i^t) \tag{12}$$

From Equations (11) and (12), we can conclude that the job requester can use a uniform pay rate k^t for all the workers if they work extra hours, but can use a different base salary b_i^t for each individual worker i.

VI. A Concrete Example

In this section, we instantiate the general solution with a concrete example so as to better explain the problem and derive the worker selection algorithm for the job requester.

In this example, we assume that we look at the optimization problem during a time period in which we can remove the superscript t from each variable. In other words, $b_i^t, k_i^t, x_i^t, \alpha_i^t$ become b_i, k_i, x_i, α_i. From the above general solution, we know that k_i can be the same to all the workers. So $k_i = k$. We set a worker's cost function $\alpha_i f_i(x_i) = \alpha_i e^{x_i}$. Though a linear function can also be assumed, we use an exponential function to emphasize a worker's rapidly growing fatigue or expenditure as the extra workload x_i increases. The coefficient α_i reflects the different cost (e.g. fatigue) level of each worker.

Once the job requester decides the pay rate k, each worker can find his optimal extra workload demand x_i^* to maximize his happiness according to Equation (7). In this example, the demand function d_i is an $\ln$ function which is the inverse of f_i. So x_i^* is

$$x_i^* = f_i'^{-1}(k/\alpha_i) = d_i(k/\alpha_i) = \ln(k/\alpha_i) \tag{13}$$

We plug x_i^* into Equation (6) and get b_i as follows:

$$b_i = \alpha_i f_i(x_i^*) - k x_i^* = \alpha_i e^{x_i^*} - k x_i^* = k - k\ln(k/\alpha_i) \tag{14}$$

Now we plug b_i, x_i^*, and k into problem (2), the minimization problem becomes:

$$\text{minimize} \quad \sum_t \sum_i k \tag{15}$$

In this problem, k is not related to time t nor individual worker i. So if we focus on a period of time, then we can remove t. Suppose we want to recruit m out of a total of N workers, then problem (15) becomes:

$$\text{minimize} \quad mk \tag{16}$$

At the same time, it still needs to satisfy the constraint in Equation (10). Putting them together, the optimization problem in this concrete case is simplified to:

$$\begin{aligned} \underset{m,k}{\text{minimize}} \quad & mk \\ \text{subject to} \quad & \sum_i^m \ln(k/\alpha_i) + mx_0 = W \end{aligned} \tag{17}$$

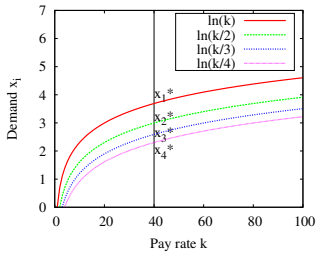


Fig. 2. Job demand x_i of different workers

Now we use Fig. 2 to explain the meaning of problem (17). Suppose there are four workers having a cost function $\alpha_i e^{x_i}$ and a coefficient $\alpha_i (1 \leq i \leq 4)$ as $1, 2, 3, 4$, respectively. The base workload of each worker to get the base payment is x_0. According to Equation (13), the extra workload demand of each worker i is $x_i = d_i(k/\alpha_i) = f_i'^{-1}(k/\alpha_i) = ln(k/\alpha_i)$, which is shown as a curve in Fig. 2. Now if the job requester sets k to be 40, then the corresponding extra optimal job demands of the workers are $x_1^*, x_2^*, x_3^*, x_4^*$, respectively. If some workers are chosen for the job, then their total workload should add up to W. For example, if all of these four workers are picked, then $x_1^* + x_2^* + x_3^* + x_4^* + 4x_0 = W$. When we select workers, we want to minimize the multiplication of the number of workers m and the pay rate k.

Next we derive the worker selection algorithm WS for the job requester. Please note that the idea of WS is not limited to this example. It is applicable to other parameter settings as well. We first come up with algorithm named *WSFK* in Fig. 3 to select workers when the pay rate k is fixed by the job requester. The coefficient α_i in the cost function can be obtained from the workers' past working performance recorded in the crowdsourcing platform. Algorithm WSFK first calculates a worker's optimal extra workload demand x_i^*, then sorts the workers according to the extra workload demands in a non-increasing order, and then picks the workers in this order until the total workload is equal to the total number of tasks W. Before we come up with a worker selection algorithm without fixing k, we need to figure out the lower and upper bounds of the pay rate k.

Upperbound of k

Assume α_{min} is the minimum value among all the coefficients $\alpha_i (1 \leq i \leq N)$ in the workers' cost functions. Then

Algorithm WSFK: Worker Selection with a Fixed k

Require: A pay rate k decided by the job requester, each worker's job cost function $\alpha_i f_i$, and base workload x_0

1: calculate each worker's optimal extra workload demand x_i^* according to Equation (13).
2: sort the workers according to their demand x_i^* in a non-increasing order.
3: pick the workers one by one in this order until the total of their workload equals W.

Fig. 3. The worker selection algorithm when k is decided

Algorithm WS: Worker Selection Algorithm

Require: A pay rate k decided by the job requester and each worker's job cost function $\alpha_i f_i$

1: $k = k_{lower}$ and $(mk)_{min} = \infty$
2: **while** $k \leq k_{upper}$ **do**
3: call algorithm WSFK to select m workers with the current k and calculate mk;
4: **if** the first solution to problem (17) is found **then**
5: $k_{upper} = m * k$;
6: **end if**
7: **if** $m * k < (mk)_{min}$ **then**
8: $(mk)_{min} = m * k$
9: **end if**
10: $k = k + increment$
11: **end while**

Fig. 4. The worker selection algorithm

for a given $k(k > 0)$, according to Equation (13), the optimal job demand corresponding to α_{min} is the largest. Therefore, k reaches its maximum value when $\ln(k/\alpha_{min}) = W$. Solving this equation, $k_{upper} = \alpha_{min} e^W$.

Lowerbound of k

Again assume α_{min} is the minimum value among all $\alpha_i (1 \leq i \leq N)$. Then for a given $k(k > 0)$, the job demand corresponding to α_{min} is the largest. If we consider all the N workers, then $N \ln(k/\alpha_{min}) \geq W$. Otherwise, the job will be too big for all the workers to finish. Solving this inequality, we get the lowerbound of k to be $k_{lower} = \alpha_{min} e^{W/N}$.

Now we have a worker selection algorithm WS shown in Fig. 4 to obtain the minimum mk with the constraint in problem (17) by searching in the range $[k_{lower}, k_{upper}]$ of k. The algorithm starts from $k = k_{lower}$ to select m workers out of N with the fixed current k using algorithm WSFK in Fig. 3. Then it calculates mk and compares it with the minimum mk. If mk is less than the minimum mk, then the minimum mk is updated to mk. And then the algorithm increments k by some amount and repeats the process. The incremented amount can be decided by the job requester. For example, some job requester likes to increase pay rate by \$1.0/hour each time. With each fixed $k \in [k_{lower}, k_{upper}]$, we have a multiplication of mk. After the whole process is over, we get the minimum mk, the pay rate k, and the selected workers.

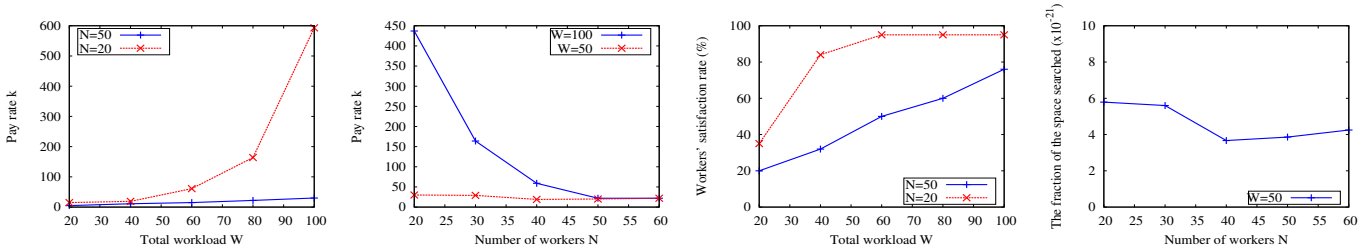
Time Complexity of Algorithm WS The loop in algorithm WS has $k_{upper} - k_{lower} + 1$ rounds. With each fixed k, we call algorithm WSFK to select m workers. The most time-consuming part in algorithm WSFK is to sort the N workers according to their work demands. If we use quicksort, the time complexity of sorting is $O(N \log N)$ on average. So the total time complexity of algorithm WS is $O(N \log N(k_{upper} - k_{lower} + 1)) = O(N \log N(\alpha_{min} e^W - \alpha_{min} e^{W/N} + 1))$.

VII. Simulation

In this section, we present the simulations we have conducted to evaluate the performance of our proposed worker selection algorithm WS. Since our algorithm is one of a kind, we wrote a customized simulator in Matlab to show its properties. In our experiments, we set both the base workload x_0 for each worker and the increment of k to 1. We randomly generated each worker's α_i in the range of $[1, 10]$. We ran algorithm WS 300 times and averaged the generated outputs.

In the first experiment, we wanted to check the relationship between the pay rate k and the total workload W when the number of workers N is fixed. We set the number of workers N to 20 first and then to 50. We increased the total workload W from 20 to 100. The results are shown in Fig. 5(a). From the results, we can see that when the number of workers is fixed, with the increase of the total workload, the pay rate k should also be increased. This is because with the fixed number of workers, the job requester has to improve pay rate to motivate the workers to pick up more workload. We can also see that when the number of workers is large ($N = 50$) relative to the total workload, the pay rate k is small and increased slowly. And when the number of workers is small ($N = 20$) relative to the total workload, the pay rate k is large and increased very quickly. This means that if there are a lot of workers, it benefits the job requester because he can use a lower pay rate to get the job done. On the other hand, if there are not many workers, the job requester has to pay a higher incentive. This proves that crowdsourcing allows a job requester to pay a low rate to recruit participants from a large number of workers.

In the second experiment, we intended to show the relationship between the pay rate k and the number of workers N when the total workload W is fixed. We set the total workload W to 50 first and then to 100. We increased the number of workers N from 10 to 60. The results are shown in Fig. 5(b). From the results, we can see that when the total workload is fixed, with the increase of the number of workers, the pay rate will be decreased for the same reason that more workers can make the pay rate lower. Also, when the workload is large relative to the number of workers (when $W = 100$ and $N = 20$), the pay rate is very high. With the increase of the number of workers, the pay rate is brought down quickly. When the workload is small relative to the number of workers (when $W = 50$), the pay rate is stable with the increase of the number of workers. This means that if there are enough workers for the job, the pay rate can be lower and if there are not enough workers, the requester has to increase pay rate to motivate the existing workers. Again, this justifies the use of crowdsourcing to get a job done.



(a) Pay rate vs. total workload (b) Pay rate vs. numbers of workers (c) Workers' satisfaction rate (d) Fraction of space searched

Fig. 5. Simulation results

In the third experiment, we aimed to check the satisfaction rate of the workers. The satisfaction rate is calculated as the ratio of the number of selected workers (m) versus the total number of workers (N). We fixed the number of workers N to 20 first and then to 50. We increased the total workload W from 20 to 100. The results of workers' satisfaction are shown in Fig. 5(c). From the results, we can see that when the number of workers is large ($N = 50$) relative to the size of a job, the workers' satisfaction rate is low and when the number of workers is small ($N = 20$) relative to the size of a job, the workers' satisfaction rate is high. This is because when the pool of the workers is small, most of them will be chosen. So more workers will get paid and be satisfied. If the pool is large, then only a few of them will be chosen. Thus, it is better for workers to participate in more crowdsourcing jobs to increase their chances.

In the fourth experiment, we evaluated the running time of our algorithm WS by checking the size of its searching space. In WS, if the total workload W increases, the upperbound of k will be increased exponentially because $k_{upper} = \alpha_{min} e^W$. But actually we do not search the full range of k_{lower} and k_{upper}. We stop the search long before that. When the first solution to the constraint $\sum_i^m \ln(k/\alpha_i) + mx_0 = W$ in problem (17) is found, we can substantially reduce k_{upper} by updating it to $m'k'$, where m' and k' are the number of workers selected and the pay rate, respectively in that solution. If there are more solutions when $k > m'k'$, the multiplication of k and m at that time must be greater than $m'k'$. The search efficiency following this idea is demonstrated in Fig. 5(d) where the percentage of the whole $[k_{lower}, k_{upper}]$ space searched is shown when $W = 50$ and N increases from 20 to 60. We can see that WS only searches a tiny fraction ($\times 10^{-21}$) of the whole searching space to find the solution.

VIII. Conclusion

In this paper, we have addressed two job requester's issues of how to design a good payment scheme to maximize profit and select qualified workers to do the job on an open crowdsourcing platform. We have adopted a widely-used payment formula that has not been discussed in crowdsourcing to formulate an optimization problem. We have first put forward a general solution and then instantiated it to get more concrete results and the corresponding worker selection algorithm WS. In WS, we have considered the workers' past working performance to guarantee crowdsourcing quality. Simulation results have shown that a job requester can pay much less to get the job done in a crowdsourcing environment and our worker selection algorithm is efficient in that it only searches a tiny space to find the solution to the optimization problem. Our effort here has provided an evidence to support the benefits of using crowdsourcing in our daily lives. In the future, we will further investigate workers' characteristics or performance indicators using data mining and machine learning techniques to refine the worker selection algorithm and thereafter provide more rigorous guarantee of crowdsourcing quality.

Acknowledgments

This research was supported in part by NSF under CNS1305302, ACI1440637, and CNS1065665, and NSF/BBN under CNS1346688 for project 1936.

References

[1] Amazon mechanical turk. http://mturk.com.
[2] Freelancer network. http://guru.com.
[3] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
[4] Y. J. Chen, B. C. Li, and Q. Zhang. Incentivizing crowdsourcing systems with network effects. In *IEEE INFOCOM*, 2016.
[5] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha. Understanding the coverage and scalability of place-centric crowdsensing. In *UbiComp*, pages 3–12, 2013.
[6] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos. TRAC: Truthful Auction for Location-Aware Collaboration Sensing in Mobile Crowdsourcing. In *IEEE INFOCOM*, 2014.
[7] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Comm. Magazine*, 49(11):32–39, 2011.
[8] J. Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6), 2006.
[9] L. G. Jaimes, I. Vergara-Laurens, and M. A. Labrador. A location-based incentive mechanism for participatory sensing systems with budget constraints. In *IEEE PERCOM*, 2012.
[10] J. S. Lee and B. Hoh. Sell your experiences: a market mechanism based incentive for participatory sensing. In *IEEE PERCOM*, 2009.
[11] C. Lofi and K. E. Maarry. Design Patterns for Hybrid Algorithmic-Crowdsourcing Workflows. In *IEEE Business Informatics*, 2014.
[12] T. Luo, S. S. Kanhere, S. K. Das, and H.-P. Tan. Optimal prizes for all-pay contests in heterogeneous crowdsourcing. In *IEEE MASS*, 2014.
[13] T. Luo, S. S. Kanhere, H.-P. Tan, F. Wu, and H. Wu. Crowdsourcing with tullock contests: A new perspective. In *IEEE INFOCOM*, 2015.
[14] T. W. Malone, R. Laubacher, and C. Dellarocas. Harnessing crowds: Mapping the genome of collective intelligence. Technical report, MIT, 2009.
[15] B. S. Noveck. *Wiki Government: How Technology Can Make Government Better, Democracy Stronger, and Citizens More Powerful*. Brookings Institution Press, 2009.
[16] H. Shah-Mansouri and V. W. S. Wong. Profit maximization in mobile crowdsourcing: A truthful auction mechanism. In *IEEE ICC*, 2015.
[17] J. Sun and H. Ma. A behavior-based incentive mechanism for crowd sensing with bedget constraints. In *IEEE ICC*, 2014.
[18] D. J. Yang, G. L. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *ACM MOBICOM*, 2012.
[19] M. Yuen, I. King, and K. Leung. A survey of crowdsourcing systems. In *IEEE PASSAT and IEEE SocialCom*, 2011.
[20] D. Zhao, X.-Y. Li, and H. Ma. How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint. In *IEEE INFOCOM*, 2014.