Crowdsourcing Multi-Objective Recommendation System[*]

Eiman Aldhahri
The University of Memphis
Memphis, Tennessee
aldhahri@memphis.edu

Vivek Shandilya
Jacksonville University
Jacksonville, Florida
shandilya@ju.edu

Sajjan Shiva
The University of Memphis
Memphis, Tennessee
sshivi@memphis.edu

ABSTRACT

Crowdsourcing is an approach whereby employers call for workers online with different capabilities to process a task for monetary reward. With a vast amount of tasks posted every day, satisfying the workers, employers, and service providers who are the stakeholders of any crowdsourcing system is critical to its success. To achieve this, the system should address three objectives: (1) match the worker with suitable tasks that fit the worker's interests and skills and raise the worker's rewards and rating, (2) give the employer more acceptable solutions with lower cost and time and raise the employer's rating, and (3) raise the rate of accepted tasks, which will raise the aggregated commissions to the service provider and improve the average rating of the registered users (employers and workers) accordingly. For these objectives, we present a mechanism design that is capable of reaching holistic satisfaction using a multi-objective recommendation system. In contrast, all previous crowdsourcing recommendation systems are designed to address one stakeholder who could be either the worker or the employer. Moreover, our unique contribution is to consider each stakeholder to be self serving. Considering selfish behavior from every stakeholder, we provide a more qualified recommendation for each stakeholder.

KEYWORDS

crowdsourcing, recommendation, task matching.

ACM Reference Format:
Eiman Aldhahri, Vivek Shandilya, and Sajjan Shiva. 2018. Crowdsourcing Multi-Objective Recommendation System. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3184558.3191579

1 INTRODUCTION

Crowdsourcing is a process whereby an employer outsources tasks to a large network of crowd workers for monetary reward. The advantage of crowdsourcing lies in the ability of employers to access a large pool of highly skilled workers to process the outsourced tasks in a reduced amount of time and cost compared to in-house workers [4, 9, 18]. Recently, there has been a significant trend towards crowdsourcing systems, and several major crowdsourcing platforms have emerged, such as ClickWorker, CloudCrowd, UpWork, and the well-known Amazon Mechanical Turk.

Crowdsourcing systems have three stakeholders: a worker, an employer, and a service provider. The employer posts the task to the crowd with a deadline and monetary reward. Workers apply to the tasks that could increase their reward and rating. The service provider's role is to provide a recommendation list that matches workers with tasks accurately in order to maximize the commission for the accepted tasks.

Due to the large number of tasks and workers available on crowdsourcing system, finding an appropriate task (or set of appropriate tasks) and a worker (or set of workers) is a strenuous and time-consuming process [8, 20]. An appropriate task depends mainly on two factors: interest and skills [8]. Interest is measured based on multidimensional factors that are weighted differently for each worker: the monetary reward and rating score. Moreover, selecting the most qualified worker is also a challenge even if we consider the worker's rating score. This score could reflect the worker's overall proficiency rather than the specialized rating. The aforementioned task-worker matching is an important factor to eliminate low-quality solutions, which is a major problem in crowdsourcing data management [12]. Another problem that could affect the stakeholders' goal is if a worker works on a large number of tasks at the same time, which could decrease the solution efficiency. As an alternative, part of these tasks could be assigned to less experienced workers who have more time, which could increase the solution efficiency. In another scenario, if we recommend tasks to the most efficient worker, the employer's goal will be satisfied. However, the worker may get busy processing low monetary tasks and miss some high monetary tasks. Therefore, a well-structured recommendation system, which satisfies all stakeholders and addresses the aforementioned difficulties, should be constructed. Such a system would entail workers finding their preferable task, employers getting a more qualified solution, and service providers increasing the accepted task rate to increase their platforms' income and popularity.

Crowdsourcing systems have four archetypes based on the platform's main function: Crowd Processing, Crowd Rating, Crowd Solving, and Crowd Creation [8]. Crowd Processing seeks micro-tasks that do not require specific skills such as Amazon Mechanical Turk. Crowd Rating seeks workers' perspectives on a given topic, which is what TripAdvisor does. Crowd Solving seeks a task that requires certain skilled workers, where solutions are acquired independently, as with InnoCentive. Crowd Creation seeks defined tasks from workers who have different skills, where the submitted solutions are aggregated to include the overall task solution, as with Wikipedia [8]. Crowdsourcing systems could also be classified based on the nature of their behavior, which can be competitive or hiring [1]. In competitive behavior, any worker may process the task without a permission. Then, the prize goes to one or more

[*]Produces the permission block, and copyright information

workers who provide the best solution. In hiring behavior, employers need to grant their permission to the worker before he/she can start processing the task. Then, the hired worker receives the rewards based on its correctness. Moreover, tasks in crowdsourcing systems can be classified as micro-task (e.g., labeling an image), which takes several seconds, and macro-task (e.g., creating of an analytical paper, web design), which takes more time [12].

In this paper, we consider hiring crowdsourcing with macro-tasks. As the majority of existing crowdsourced research has focused on micro-tasks, we choose to focus on macro-task which is considered an important research topic [12]. We assume that every stakeholder acts selfishly to maximize profit. Because of this assumption, we present a mechanism design based on a multi-objective recommendation system to reach holistic satisfaction through the following: matching the worker with a suitable task that fits the worker's skills, raising the worker's rewards and rating, giving employers more qualified solutions with lower costs without affecting their rating, and raising the rate of accepted task, which will increase aggregated commissions accordingly.

The main contributions of this paper are

(1) A model for quantitatively formulating the strategic interaction of the stakeholders (employers, workers, and the crowdsourcing service provider),
(2) Algorithms to compute the recommendation for both the employers and the workers, and
(3) A numerical simulation to evaluate the effectiveness of the recommendation.

The rest of the paper is organized as follows. Section 2 reviews related work, Section 3 describes the workflow, Section 4 presents the problem formulation, Section 5 describes the proposed recommendation model, Section 6 describes the experiment, and section 7 concludes the paper.

2 RELATED WORK

We have conducted a detailed survey and critical study of state-of-the-art recommendation systems that are ubiquitous among crowdsourcing and other online systems [2]. Our research shows that most general recommendation systems [5, 7, 10, 13, 14, 16] address one stakeholder. The main contribution of these studies was enhancing the collaborative filtering approach by utilizing different techniques, such as user-item subgroups [5], expert opinions [14], social media sentiment [16], and k-mean clustering [7]. The other main contribution was using revolutionary algorithms, such as a genetic algorithm [10].

Similarly, all crowdsourcing recommendation papers reviewed have addressed one stakeholder, either the worker [3, 6, 15, 19, 21] or the employer [22]. An important paper was by Yuen et al. [21], where the system recommends tasks on Amazon Mechanical Turk (Mturk) using a matrix factorization by extracting a worker's preferred tasks from both a worker's performance history and task search history. The other major contribution was by Lin et al. [15], who proposed a system that incorporates negative implicit feedback based on task availability. Yuen et al. [22] have considered dynamic scenarios to solve the cold start problem, which consists of new user and new item recommendation. Difallah et al. [6] have proposed different task recommendation approaches for crowdsourcing based on push methodology instead of the currently used pull methodology. The idea is to use the social media website Facebook to gather users' skills and interests from the pages they liked and the tasks they completed. Then, tasks are posted on the related worker's page.

To the best of our knowledge, no prior literature has considered satisfying the goal of all three stakeholders. Moreover, none has considered the other party's behavior to provide more qualified recommendations. Designing such a recommendation system would be a great opportunity for effective crowdsourcing as we have suggested in this study.

3 WORKFLOW

In this section, we provide an overview of the work flow as an interaction scenario for the stakeholders of the crowdsourcing platform.

- Employers $e_1, e_2, e_3, \dots$ register as members.
- Workers $w_1, w_2, w_3, \dots$ register as members.
- e_h posts tasks $a_j, a_{j+1}, .. : h, j = 0, 1, 2, ..$ at time $t_1, t_2, t_3, \dots$
- For each task, employers may specify:
(1) The required skills,
(2) Monetary rewards, and
(3) Time deadline.
- Workers w_i, w_{i+1}, w_{i+2} are qualified for the task a_j.
- At time t_1, t_2, t_3, t_4, workers w_i, w_{i+2} apply for the task a_j as long as $t_{j+n} > t_l : l = 1, 2, 3, 4$. t_{j+n} is the threshold time when the e_h must respond to the workers who accepted the task with a decision of hired / not hired.
- Employer e_h removes the task a_j from the available tasks after two conditions are met: 1) the task was allotted to the number of required workers, and 2) the employer accepted the task from one or more workers.
- Workers w_i, w_{i+2} completed and submitted their work for the task a_j.
- Employer e_h accepted the work for the task a_j from one or more workers, who submitted their work, and paid the associated rewards.
- In t_l seconds, the service provider S made C_l dollars as a commission for the task a_j.[1]
- Consider another case for task a_{j_1} a_j, which finally gives C_2 in t_2 seconds.[2]
- In a given duration of time T, maximize $\sum C_1 + .. + C_n$.
- The probability that the task is completed and the employer accepted the task is P_1.
- The probability that S will get the commission C_1 is P_1.

The recommendation system should order the task's recommendation list to the workers such that the expected cumulative commission is maximized, which means
$\sum P_1{}^1 C_1 + P_1{}^2 C_2 + \dots + P_1{}^n C_n$ is maximized.

[1]C_1 is the commission that the service provider grants when the task a_j is accepted.
[2]C_2 is the commission that the service provider grant when the task a_{j_1} is accepted.

4 PROBLEM FORMULATION

The actual system that solves this problem should consider k employers posting n tasks to m workers to maximize the commission. This is not mainly about the money but rather a complex and definitive matrix of overall platform success. In other words, maximizing the commissions means maximizing the rate of accepted tasks, which is a consequence of satisfying the employers by giving them a qualified solution and satisfying workers by giving them the associated rewards. Below, we identify the exact role for each stakeholder in the crowdsource platform.

4.1 Worker

- Lists skills on the profile
- $\langle Decision \rangle$ applies for n_{11} out of n_1 tasks that fits the profile.
- $\langle Decision \rangle$ completes $n_1{}^{`}{}_1$ out of n_{11}.
- $n_1{}^{`}{}_1$ <n_{11} <n_1 <n.

4.2 Employer

- Posts the task.
- $\langle Decision \rangle$ allots the task to the best m_{11} workers out of m_1 who applied for the task.
- $\langle Decision \rangle$ pays the $m_1{}^{`}{}_1{}^{`}$ workers out of $m_1{}^{`}{}_1$ who submitted the solution to the task.
- $m_1{}^{`}{}_1{}^{`}$ <$m_1{}^{`}{}_1$ <m_{11} <m_1 <m.

4.3 Service Provider

- Orders the recommended tasks for the workers.
- Sorts n_1 tasks in the order that leads to maximize $\sum c_1$.
  The ordering of n_1 is a list that looks like this:
  $j_{1_0}, j_{1_1}, j_{1_2}, j_{1_3},$ fi., $j_{1_{n_1}}$.
  The worker accepts tasks with probability
  $P(j_{1_0}), P(j_{1_1}), P(j_{1_2}),$ fi., $P(j_{1_{n_1}})$
  Where $P(j_{1_y}) \geq P(j_{1_{(y+z)}})$
  Where $z > 0, Z \in I$

5 THE RECOMMENDATION MODEL

This section describes the proposed model (Fig. 1). It is a multi-objective problem for both workers and employers. The worker's goal is to work on the tasks that maximize the reward and rating during a specified time. The employer's goal is to have more qualified solutions, pay less, and decrease the negative rating. In other words, if the employer hires a large number of workers for a task, the probability of getting more qualified solutions will increase. However, the employer in this case will have two choices. First, the employer could pay for all the workers who submitted qualified solutions, which will increase the cost. Second, the employer could pay for a subset of the workers who submitted qualified solutions, which will decrease the employer's rating from unsatisfied worker reviews.

The proposed model recommends the optimal choices for each worker and employer. Accordingly, the rate of accepted tasks will be maximized and the service provider's goal will be achieved.

There are two cases in the proposed model: Case-0, where workers can work only on one task at a time, and Case-1, where workers can work on multiple tasks at a time.

5.1 Worker's Objective

This section describes the worker objective in detail.

For each worker w_i,

Step 1: Find the set of tasks that fits his or her interests
For each task a_j there are required skills $Sk[a_j]$=$\{sk_1, sk_2, ..\}$, and each worker has a set of skills $Sk[w_i] = \{sk_1, sk_2, ..\}$. If $Sk[a_j] \subset Sk[w_i]$, then $a_j \in Tasks[w_i]$ where $Tasks[w_i]$ is the set that contains all the tasks that fit the worker's interests.

Step 2: Calculate the expected monetary rewards for each task in the list $Tasks[W_i]$ using Algorithm 1. Considering the history, with weighted consideration of the future expectations, we apply the Discount Factor equation Eq.(16).

Step 3: Calculate the expected rating for each task in the list $Tasks[w_i]$ using Algorithm 2.

Step 4: Calculate each task's type weight using Algorithm 3.

Step 5: Recommend tasks to the worker that will maximize the rewards and rating using Algorithm 4.

5.1.1 *Expected Payment (ExP).*

Each task a_j has a specified monetary reward, deadline, and required skills. The payment is not guaranteed unless the employer approves the work. Usually if the submitted work meets all the required specifications, the employer will approve the worker payment. However, there is no obligation for payment if the employer refuses to pay. Therefore, the employer's rating is an important factor to reflect the employer's trustworthiness.

From the worker's history, we can get an expectation of how likely the worker will be paid for each type of task in the set $Tasks[w_i]$. Moreover, based on the employer history, we can estimate how likely each employer will pay the worker.

Calculating the expected payment consists of two steps.
First, from the worker history, calculate the proficiency level of the worker in each skill or type of task using the following equations:

Q_j is the probability that worker w_i will complete tasks from type j

$$Q_j = \frac{\sum S[tasks_j]}{\sum H[tasks_j]} \tag{1}$$

where for each worker, $S[tasks_j]$ is the submitted or completed tasks from type j, $H[tasks_j]$ are the tasks that the worker was hired to process from type j.

Q_{j_1} is the probability that worker w_i will be paid for tasks from type j

$$Q_{j_1} = \frac{\sum Paid[tasks_j]}{\sum S[tasks_j]} \tag{2}$$

where $Paid[tasks_j]$ is the accepted tasks from type j.
The worker proficiency level in type j tasks is

$$Prof^j = Q_j * Q_{j_1} \tag{3}$$

Second, calculate the degree of employer trustworthiness, considering the worker's rating as a substantial factor to get more

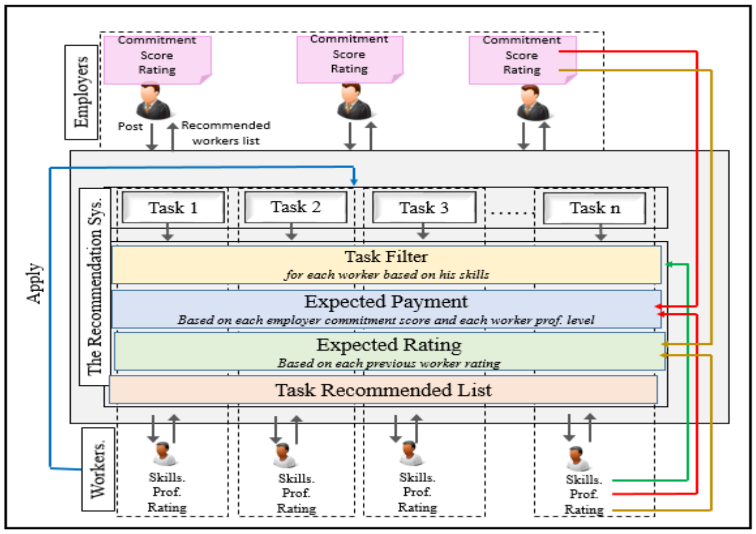

Figure 1: Recommendation Model

accurate results. For instance, a review from a five-star worker has more impact than a review from a two-star worker because the more highly rated worker is more trustworthy.

From the employer history:
Q_h is the probability that the employer e_h will pay the workers who submitted the solutions considering worker ratings $R[w_i]$ as a weight factor

$$Q_h = \frac{\sum Paid[w_i] * R[w_i]}{\sum S[w_i] * R[w_i]} \tag{4}$$

where for each task, $Paid[w_i]$ is the number of workers who got paid, and $S[w_i]$ is the number of workers who submitted the task.

Then from Equations 3 and 4, we can calculate the expected payment for each task in the worker's task list by the following equation:

$$ExP[a_j] = Prof^j * Q_h * Reward[a_j] \tag{5}$$

where $Reward[a_j]$ is the monetary reward for task a_j, and $Prof^i$ is the proficiency level of the worker in type j tasks.

Maximize $ExP(W_i) = \sum_{i=1}^{y} ExP[a_j]$
where y is the number of tasks in the $Tasks[w_i]$ set.

5.1.2 *Expected Rating (ExR).*

The rating system in crowdsourcing allows employers and workers to rate each other. Rating is a substantial factor, so it is important to optimize the rating score. From the employer's perspective, workers' ratings could help decide which worker should be hired. From the worker's perspective, ratings could help decide which tasks to apply for. As we have described in the employer rating Equation 4, the evaluator rating is considered to aggregate the overall rating score. To justify the evaluator rating factor needs, consider the following example. Because the rating system is mutual, as explained above, a dishonest employer could give workers a bad rating to decrease their overall rating. This lowered rating would result in workers' evaluations not having much effect on the employer's rating in Equation 4. However, if we consider the employer's rating in evaluating the workers' ratings, the rating score could be more trustworthy.

$$ExR[j] = \frac{\sum_{x=1}^{n} R[a_x] * R[e_h]}{\sum_{x=1}^{n} R[e_h]} \tag{6}$$

Where $ExR[j]$ is the expected rating for type j tasks, n is the total number of type j tasks that the worker has submitted before, $R[a_x]$ is the rating score for task x, and $R[e_h]$ is the employer e_h rating.

Algorithm 1 Expected Payment

```
1:  INPUT: Task set $Tasks[w_i] = \{a_1, a_2, ..., a_n\}$
2:  INPUT: Employers who posted the tasks $E = \{e_1, e_2, ..., e_h\}$
3:  Each task $a_j$ is a tuple of three values $\{rewards, deadline, skills\}$
4:  Task types in $Tasks[w_i]$ : $T_a = \{a_1, a_2, ..., a_m\}$
5:  OUTPUT: The expected payment for each task $a_j$ in $Tasks[w_i]$

6:  First: Calculate the proficiency level for each type of task
7:  for all $a_t$ in $T_a$ do
8:      Calculate $Q_j = \frac{\sum S[tasks_j]}{\sum H[tasks_j]}$
9:      Calculate $Q_{j_1} = \frac{\sum Paid[tasks_j]}{\sum S[tasks_j]}$
10:     Calculate $Prof^j = Q_j * Q_{j_1}$
11: end for
12: Second: Calculate the employer commitment
13: for all $e_i$ in $E$ do
14:     $Q_h = \frac{\sum Paid[w_i]*R[w_i]}{\sum S[w_i]*R[w_i]}$
15: end for
16: Finally: Calculate the ExP for each task $a_j$ in $Tasks[w_i]$
17: $ExP[a_j] = Prof^j * Q_h * Reward[a_j]$
```

Algorithm 2 Expected Rating

```
1: INPUT: Task set $Tasks[w_i] = \{a_1, a_2, a_3, ..., a_n\}$
2: INPUT: Employers who posted these tasks $E = \{e_1, e_2, ..., e_h\}$
3: Each employer has a rating score value $R[e_h]$
4: OUTPUT: The expected rating for each task $a_j$ in $Tasks[w_i]$
5: for all $a_j$ in $Tasks[w_i]$ do
6:     Calculate $ExR[j] = \frac{\sum_{x=1}^{n} R[a_x]*R[e_h]}{\sum_{x=1}^{n} R[e_h]}$
7: end for
```

5.1.3 Skill Based Workload. From the workers' history, we can calculate how many tasks each worker can handle successfully at the same time, i.e., the worker's appropriate workload based on the task type. For example, worker w_i could work successfully on an average of three tasks simultaneously when working on programming tasks, two when working on design tasks, and so on for each type of task.

For each worker, calculate the appropriate workload for each task's type j.

For each worker, if k_1 tasks were being done together during a given instance s_i, in which the given task type was present, find the average of the total number of tasks as follows:

$$L[j] = \frac{\sum_{i=1}^{S} Paid[a_j]}{S} \tag{7}$$

where $L[j]$ is the worker w_i workload for task type j, S is the total number of instances, and $Paid[a_j]$ is the number of the accepted tasks during an instance s_i, considering only the instances in which the given task type was present.

By applying Eq.(7) each worker w_i will have a different workload for each task type. Each workload will be converted to a weight score by the following equation:

$$Tw[j] = \frac{1}{L[j]} \tag{8}$$

where $Tw[j]$ is the worker's w_i weight score for task type j.

For example, if the workload of worker w_i for a programming task is 3, that means he or she could work efficiently on two additional tasks, and the weight score for the programming tasks will be equal to 0.33.

5.1.4 Worker Recommendation Task. The worker utility function is $Maximize[reward, rating]$
It is a multi-objective optimization problem (MOP) with two objectives: reward and rating. In the literature, researchers have studied MOPs from different points of view, so different solution philosophies and goals exist. There are three main classes for preference MOP, where preference information is needed to solve the problem. These classes are *a priori*,*a posteriori*, and interactive, where a preference information is involved from the decision maker (DM) in different ways. In the *a priori* method, the DM will first determine the preference information, and then the solution will be found. In the *a posteriori* method, the solutions will be found first, and then the DM will choose from among them. In the interactive method, the DM's preference information will be specified during computation.

Algorithm 3 Task Type Weight (Tw)

```
1: INPUT: Worker $w_i$ task type set $T_a = \{a_1, a_2, ..., a_m\}$
2: INPUT: Previous instances set $S = \{s_i, ..., s_n\}$
3: Each instance $s_i$ is a tuple of instance's id and the list of the
   worker's accepted tasks during this instance.
4: OUTPUT: $Tw$ set, which contains the weight score for each
   task type.
5: Int ins-counter = 0, task-counter = 0;
6: for all $a_i$ in $T_a$ do
7:     ins-counter = 0, task-count = 0;
8:     for all $s_i$ in $S$ do
9:         if $s_i$ contains task from type $a_i$ then
10:            ins-counter ++ ;
11:            task-counter + = the total number of tasks in $s_i$;
12:        end if
13:    end for
14:    $L[a_i]$ = task-counter / ins-counter ;
15:    $Tw[a_i] = 1/L[a_i]$
16: end for
```

To optimize the worker's objectives, the *a priori* method is used in this paper. Workers will specify their preference rating score first, which will be used as a constraint value to solve for maximizing the reward's value.

New workers could be more interested in building a robust history and set the rating constraint value to four or five stars in order to increase their future chances to compete with senior workers, who have a high rating score. However, each worker could set the rating constraint based on interest,

$$Maximize f(x) = \sum_{j=1}^{n} Reward[a_j] \tag{9}$$

Subject to $R[a_j] \geq R$
where R is a rating constraint value set by the worker.

There are two cases in the proposed model:
Case-0: Only one task at a time. It can be solved by sorting the tasks based on the expected payment considering the expected rating as a constraint value.
Case-1: Multiple tasks at the same time. If the worker wants to work on a set of tasks to maximize his/her objectives during a specified time, dynamic programming is used to solve the problem. It becomes a knapsack problem where we try to maximize the value within the time limit considering the weight score for each task from Algorithm 4, where the total weight score should be equal to or less than 1. The following is an illustrated example:
If the worker has the tasks set as in Table 1, by applying the *Tw* Algorithm 4, the tasks demonstration during the time period is shown in Figure 2.
The Worker Recommendation Task Algorithm uses dynamic programing for the knapsack problem [17], where $KnapSack(ExP, Tw, n, T)$ = recommended task set, ExP is the item value, Tw is the weight value, n is the number of items, and T is the total weight.

Table 1: Task List

Task	*Type*	*Deadline*	Tw_i	*ExP*
1	*Programming*	*30*	*0.33*	*300*
2	*Programming*	*90*	*0.33*	*600*
3	*Web design*	*120*	*0.5*	*500*
4	*Web design*	*60*	*0.5*	*400*
5	*Programming*	*30*	*0.33*	*250*
6	*Web design*	*90*	*0.5*	*300*

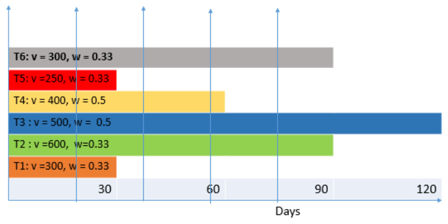

Figure 2: Task list

Algorithm 4 Worker Recommendation Task

```
1:  INPUT: Task set $Tasks[w_i] = \{a_1, a_2, ..., a_k\}$
2:  INPUT: $ExP = \{ExP[a_1], ExP[a_2], ...\}$
3:  INPUT: $ExR = \{ExR[a_1], ExR[a_2], ...\}$
4:  INPUT: $Tw = \{Tw[a_1], Tw[a_2], ...\}$
5:  INPUT: Minimum rating constraint $R$
6:  INPUT: Time to optimize $TL$
7:  OUTPUT: The recommended tasks
8:  $W_i = \emptyset$
9:  for all $a_j$ in $Tasks[w_i]$ do
10:     if $a_j deadline > TL$ then
11:         exclude $a_j$
12:     end if
13:     if $ExR[a_j] > R$ then
14:         add $a_j to W_i$
15:     end if
16: end for
17: if Worker works on one task at a time then
18:     Sort $W_i$ based on $ExP[a_j]$ in decreasing order
19: else
20:     TaskSet = $KnapSack(ExP, Tw, n, T)$
21: end if
22: Print TaskSet
```

5.2 Employer's Objectives

The employer's basic objective is to obtain more qualified solutions for the payment expended and rating awarded. Hiring more workers could increase the chance of receiving a better solution. Consequently, the two main objectives, which are cost and rating, will have a negative effect as described in Section 1.

The employer's objectives could be optimized by choosing fewer qualified workers. By doing so, we will decrease the number of unsatisfied (due to not being compensated as per their expectation) workers, which could cause negative ratings and decrease the rewards.

Another important factor is the worker's current workload. Some workers may apply for a large number of tasks and then choose a subset of these tasks to process. Some other workers may choose to work on a large number of tasks, which could decrease the quality of the task solution.

To address the employer's objectives, we will tackle the aforementioned factors as follows.

First, for each task a_j posted by employer e_h, k workers will apply. Each worker w_i will have a proficiency level for this kind of task from Equation 3. Moreover, each worker will have a rating score based on employer evaluations. Then, for each worker, we calculate the potential success (PS) by

$$PS = Prof^i * R[w_i]$$

Let x be the number of workers for task a_j. The expected expenditure for task a_j is

$$F(x) = Minimize_x[Maximize \sum_{i=1}^{k} Prof^i * R[w_i](Reward[a_j])] \tag{10}$$

The employer objective function will be
$Maximize\ [workdone - payment - negativerating]$

Second, from the workers' current processing tasks, we can get the current worker's workload based on each task's weight score in Section 5.1.3.

If the worker's w_i current workload (CW) is 0.75, that indicates the worker could still work efficiently on more tasks. However, if the CW of another worker w_j is 0.0, it indicates worker w_j could process the task better because that worker has more time than w_i considering an equivalent or comparable PS score for both workers.

Finally, to optimize the employer's objectives, the service provider needs to recommend workers with a higher PS score and a lower CW. To solve this MOP, the interactive method is used as follows:

(1) Employer e_h sets the number of required workers.
(2) The service provider finds all the non-dominated solutions as described in the Worker Recommendation List Algorithm.
(3) Based on this list, the employer e_h resets the number of required workers.

The service provider will recommend the employer to choose at least two qualified workers and some new workers who are willing to build a history. Hiring new workers could increase the chance of getting better solutions in terms of increasing the number of workers, but it does not have much negative effect on the employer's rating because the new workers do not have a sufficient rating score. Employers will set their own parameters to optimize their objectives based on the applicants' PS and CW. If there are two 0.9 applicant workers, the employer could set the number of required workers to two plus some new workers to help them in building a history. However, if the applied worker has a lower PS and higher CW, the employer could increase the number of required workers. The mutual rating system could help minimize employing unnecessarily large numbers of workers and wasting their time processing a task with a low chance of acceptance.

Algorithm 5 Worker Recommendation List

```
1:  INPUT: Workers set $W = \{w_1, w_2, w_3, ..., a_k\}$
2:  Each worker $w_i$ has a proficiency score $Prof_i$ and rating score $R[w_i]$
3:  Each worker $w_i$ has a CW
4:  $ND$ list = $\varnothing$
5:  OUTPUT: List of all the non-dominated workers
6:  for all $w_i$ in $W$ do
7:      Calculate the PS: $PS[w_i]$ by $Prof_i * R[w_i]$
8:  end for
9:  for all $w_i$ in $W$ do
10:     if $PS[w_i] > PS[w_{i-1}]$ then
11:         if $CW[w_i] < CW[w_{i-1}]$ then
12:             Add $W_i$ to ND
13:         end if
14:     end if
15: end for
16: for all $w_i$ in $W$ do
17:     if $CW[w_i] < CW[w_{i-1}]$ then
18:         if $PS[w_i] > PS[w_{i-1}]$ then
19:             Add $W_i$ to ND
20:         end if
21:     end if
22: end for
```

5.3 Service Provider's Objective

The service provider's objective is to maximize the aggregated commission by listing recommended tasks to the workers and the recommended workers to the employers.

S_h : Probability that worker w_i will apply for task a_j.

S_{h_1}: Probability that worker w_i will get the task a_j.

S'_{h_1}: Probability that worker w_i will complete task a_j.

S''_{h_1}: Probability that worker w_i will get paid for task a_j.

$$MaximizeTask(a_j) = \sum_{i=1}^{y} S_h S_{h_1} S'_{h_1} S''_{h_1}(C)$$

where C = the commission aggregated for task a_j.
The service provider utility function is

Maximize [commission - negative employ rating - negative worker rating]

5.4 Discount Factor

We get the value of n– the depth of history we are going to consider – from the demography in the system. We consider the history of similar tasks until the effect of that state becomes less than ϵ in terms of probability. In other words, the effect of that state is no more than random on the present state. If most workers completed three tasks one after another, we would get $n = 3$, which means we are going to consider three history records.

Once we have the n, we can calculate the discount factor β. The discount factor is needed because we are considering that what happened in the recent past is more influential on the worker's future attitudes.

$$\beta + \frac{\beta}{2} + \frac{\beta}{3} + ... + \frac{\beta}{n} \tag{11}$$

From Equation 11, we can get the value of β,

$$\beta = \frac{1}{1 + \frac{1}{2} + \frac{1}{3} + ..\frac{1}{n}} \tag{12}$$

We are looking for the probability of worker w_i getting payed for job j_i now.

$$P(\epsilon(j_{i-1}) = \beta_0 * P(\epsilon(j_i) + \beta_1 * P(\epsilon(j_{i_1}) + ..\beta_n * P(\epsilon(j_{i-n}) \tag{13}$$

where $\sum_{k=1}^{n} \beta_k = 1$

$$\sum_{k=1}^{n} \beta_k P(\epsilon(j_{i-K+1}) \tag{14}$$

where $\sum_{k=1}^{n} \beta_k = 1$

6 EMPIRICAL EVALUATION

This section describes an experiment that simulates the crowdsourcing paradigm. Our experiment is designed to address three questions:

(1) How does the proposed method compare with baseline and state-of-the-art approaches?
(2) What is the computational complexity of the proposed recommendation model?
(3) How scalable is the proposed model?

To demonstrate the superiority of our proposed model, we chose two models as a baseline for the comparison: the traditional model and the most recently published model.

6.1 Baseline Model

The traditional model uses a greedy algorithm to recommend the highest reward tasks that match workers' skills. The most recent recommendation system in crowdsourcing relies on matrix factorization based on worker performance history and worker task search history [22].

6.2 Dataset

The data needed to evaluate our proposed model requires the complete worker history and employer history. To the best of our knowledge, such data is only accessible by the crowdsourcing administrators and is not publicly available.

We evaluated our model with synthesized datasets. To make the datasets realistic and unbiased, we generated them from two distributions, binomial and uniform, with different scales. Table 2 shows the characteristics of the synthesized datasets. Binomial distributions were chosen because each submitted task has only two possibilities, accept or reject. The rating value was generated using discrete uniform distribution, yielding integers only. The datasets generated are implemented using numpy.random sampling module in Python [11]. With this module, the generated data can be customized randomly from any distribution with specified parameters.

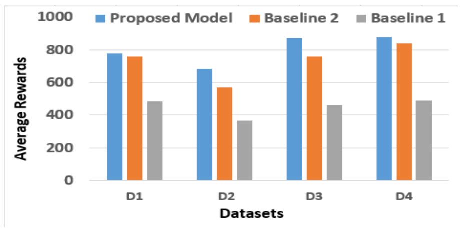

Figure 3: Evaluating the Workers' objectives

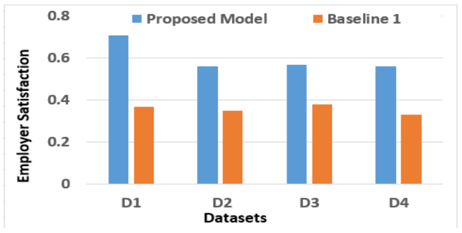

Figure 4: Evaluating the employers' objectives

Experiments were conducted on a standard desktop PC (Quadcore Intel i7 CPU@3.5 GHz).

Table 2: Characteristics of Datasets

Dataset	*Dist.*	*Task*	*Category*	*Worker*	*Employer*
D1	*binomial*	*1000*	*5*	*50*	*50*
D2	*binomial*	*5000*	*10*	*100*	*100*
D3	*uniform*	*1000*	*5*	*50*	*50*
D4	*uniform*	*5000*	*10*	*100*	*100*

6.3 Experimental Procedure

First, we evaluated the workers' objectives. For the comparison goals, we compared the reward average of five randomly selected workers in each model. Each worker had different proficiency and rating scores associated with each skill.

In our model, we calculate the expected rewards and rating. However, in the crowdsourcing paradigm, payment is not guaranteed as described earlier. To simulate the crowdsoursing paradigm, we designed a stochastic program that runs 10, 20, and 50 times, each time with a possibility of acceptance or rejection based on the employer's commitment score and the worker's proficiency score. In each run, a random number will be generated. If the number is between zero and the potential acceptance score, the task will be considered accepted; otherwise, it will be rejected. Then, the number of accepted times will be multiplied by the actual rewards. Finally, we calculate the reward's average. The potential acceptance is calculated by multiplying the employer's commitment score by the worker's proficiency score as described earlier in the algorithms.

We ran the simulation 10, 20, and 50 times on each dataset, and compared the average rewards for the selected workers in the proposed model with the average rewards of the same workers in the two baseline models. In Baseline 1, a greedy approach was used to choose the set of tasks that would maximize the worker's objectives. In Baseline 2, the worker's performance was considered and tasks would be recommended based on the worker's previous performance. To evaluate the potential acceptance in the baseline models, we considered additional information consisting of the employer's commitment score and the worker's proficiency score. Fig. 3 shows the average rewards for the selected workers in each dataset.

Second, we evaluated the employers' objectives. For the comparison goals, we randomly selected five employers and for each employer we randomly selected one task. To evaluate the employers' objectives, we compared their potential satisfaction with hiring each worker for the selected task.

To simulate the employer's role, we ran the simulation 10, 20, and 50 times on each dataset, the simulation calculate the probability of the potential satisfaction for each selected employer. Each time had a possibility of worker success or failure based on the worker's PS for task a_j. In each run, a random number was generated. If the number was in the interval of the worker's success range based on the $PS[a_j]$ score, the task was considered accepted and the worker succeeded. Then, the program calculated the average of the worker's successful outcomes. After running the program 10, 20, and 50 times, the success average for each worker was calculated and compared with the baseline for the employer recommendation system, which recommends workers with the highest rating scores. Fig .4 shows the probability of the employer satisfaction with the recommended workers in each dataset.

7 CONCLUSION

We have proposed a multi-objective recommendation model for the crowdsourcing paradigm. The computational complexity for the algorithms is $O(nW)$. The model has addressed the goals of all three stakeholders (the worker, the employer, and the service provider). The model is designed as an interactive system where every worker and employer can set the parameters that meet their goals. All previous crowdsourcing recommendation systems have been designed to address one stakeholder. Moreover, no crowdsourcing recommendation system has considered the other party's behavior to provide more qualified recommendations as we have done. The experimental simulation showed the superiority of the proposed model compared to two baseline models. The proposed model is a hybrid approach that combines content based and collaborative approach to overcome each approachâĂŹs limitation. However, the model still have some cases that faces the cold start problem. The common solution in the literature is based on matrix factorization which need workers' previous rating scores. To overcome this, we are working on different approach to solve this problem by adding a simple technique without any negative affect of the scalability. In the future, we plan to design a sequential decision recommendation model. The current model would use a one shot game where the decision is made simultaneously. However, in the sequential game, one player makes a decision and then based on that, the other player makes a decision. In this model, a recommendation decision will be provided in each stage.

REFERENCES

[1] Eman Aldhahri, Abdullah Abuhussein, and Sajjan Shiva. 2015. Leveraging Crowdsourcing in Cloud Application Development. *Software Engineering and Applications* (2015).

[2] Eman Aldhahri, Vivek Shandilya, and Sajjan Shiva. 2015. Towards an Effective Crowdsourcing Recommendation System: A Survey of the State-of-the-Art. In *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on.* IEEE, 372–377.

[3] Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell. 2011. Towards task recommendation in micro-task markets.. In *Human computation.* 1–4.

[4] Daren C Brabham. 2008. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies* 14, 1 (2008), 75–90.

[5] Jiajun Bu, Xin Shen, Bin Xu, Chun Chen, Xiaofei He, and Deng Cai. 2016. Improving Collaborative Recommendation via User-Item Subgroups. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2363–2375.

[6] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2013. Pick-a-crowd: tell me what you like, and i'll tell you what to do. In *Proceedings of the 22nd international conference on World Wide Web.* ACM, 367–374.

[7] Zhi Ming Feng and Yi Dan Su. 2013. Application of Using Simulated Annealing to Combine Clustering with Collaborative Filtering for Item Recommendation. In *Applied Mechanics and Materials*, Vol. 347. Trans Tech Publ, 2747–2751.

[8] David Geiger and Martin Schader. 2014. Personalized task recommendation in crowdsourcing information systems Current state of the art. *Decision Support Systems* 65 (2014), 3–16.

[9] Jeff Howe. 2006. The rise of crowdsourcing. *Wired magazine* 14, 6 (2006), 1–4.

[10] Chein-Shung Hwang, Yi-Ching Su, and Kuo-Cheng Tseng. 2010. Using genetic algorithms for personalized recommendation. In *International Conference on Computational Collective Intelligence.* Springer, 104–112.

[11] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. (2001–). http://www.scipy.org/ [Online; accessed <today>].

[12] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. 2016. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2296–2319.

[13] Xin Li, Mengyue Wang, and T-P Liang. 2014. A multi-theoretical kernel-based approach to social network-based recommendation. *Decision Support Systems* 65 (2014), 95–104.

[14] Chen Lin, Runquan Xie, Lei Li, Zhenhua Huang, and Tao Li. 2012. Premise: Personalized news recommendation via implicit social experts. In *Proceedings of the 21st ACM international conference on information and knowledge management.* ACM, 1607–1611.

[15] Christopher H Lin, Ece Kamar, and Eric Horvitz. 2014. Signals in the Silence: Models of Implicit Feedback in a Recommendation System for Crowdsourcing.. In *AAAI.* 908–915.

[16] Kevin Meehan, Tom Lunney, Kevin Curran, and Aiden McCaughey. 2013. Context-aware intelligent recommendation system for tourism. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on.* IEEE, 328–331.

[17] Paolo Toth. 1980. Dynamic programming algorithms for the zero-one knapsack problem. *Computing* 25, 1 (1980), 29–45.

[18] Paul Whitla. 2009. Crowdsourcing and its application in marketing activities. *Contemporary Management Research* 5, 1 (2009).

[19] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2011. Task matching in crowdsourcing. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing.* IEEE, 409–412.

[20] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2012. Task recommendation in crowdsourcing systems. In *Proceedings of the First International Workshop on Crowdsourcing and Data Mining.* ACM, 22–26.

[21] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2012. Task recommendation in crowdsourcing systems. In *Proceedings of the First International Workshop on Crowdsourcing and Data Mining.* ACM, 22–26.

[22] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. 2015. Taskrec: A task recommendation framework in crowdsourcing systems. *Neural Processing Letters* 41, 2 (2015), 223–238.