From Zero to the Hero: A Collaborative Market Aware Recommendation System for Crowd Workers

Hamid Shamszare*
Stevens Institute of Technology
Hoboken, NJ, USA
hshamsza@stevens.edu

Razieh Saremi*
Stevens Institute of Technology
Hoboken, NJ, USA
rsaremi@stevens.edu

Sanam Jena
Stevens Institute of Technology
Hoboken, NJ, USA
sjena@stevens.edu

ABSTRACT

The success of software crowdsourcing depends on active and trustworthy pool of worker supply. The uncertainty of crowd workers' behaviors makes it challenging to predict workers' success and plan accordingly. In a competitive crowdsourcing marketplace, competition for success over shared tasks adds another layer of uncertainty in crowd workers' decision making process. Preliminary analysis on software worker behaviors reveals an alarming task dropping rate of 82.9%. These factors lead to the need for automated recommendation system for CSD workers to improve the visibility and predictability of their success in the competition. To that end, this paper proposes a collaborative recommendation system for crowd workers. The proposed recommendation system method uses five input metrics based on workers collaboration history in the pool, workers preferences in taking tasks in terms of monetary prize and duration, workers' specialty, and workers' proficiency. The proposed method then recommends the most suitable tasks for a worker to compete on based on workers' probability of success in the task. Experimental results on 260 active crowd workers demonstrate that just following the top three success probability of task recommendations, workers can achieve success up to 86%.

CCS CONCEPTS

• Software and its engineering → Software development process management; **• Information systems → Data analytics**; **Crowdsourcing**; **• Computing methodologies → recommendation systems**.

KEYWORDS

software crowdsourcing, worker performance, worker preference, worker success, dynamic decision making

ACM Reference Format:
Hamid Shamszare, Razieh Saremi, and Sanam Jena. 2021. From Zero to the Hero: A Collaborative Market Aware Recommendation System for Crowd Workers. In *Proceedings of The 36th IEEE/ACM International Conference on Automated Software Engineering (ASE 2021).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Both authors contributed equally to this research.

ASE 2021, 15 - 19 November, 2021, Melbourne, Australia

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Crowdsourced software development (CSD) integrates online and unknown workers' elements into the design. The success of such platforms relies heavily on a large crowd of trustworthy software workers who are registering and submitting for crowdsourced tasks in exchange of financial gains [36]. Crowd workers make their decision on participating in a task based on intrinsic values which result from workers' motivation and preferences [31] [23], and extrinsic values on the societal level such as educational background and household income[20].

In general, crowd workers choose to perform tasks based on some personal utility algorithm such as monetary prize and tasks' complexity and duration [37][29][26], their skills and some unknown factors [11]. However, they rather to work on the tasks with similar context in terms of task type, required technology and platform, and their previous experience [10][8] [39].

It seems that crowd workers analyze their probability of success based on the task's competition level and the number of more highly ranked opponents [11], and use their history of victories as a policy of assuring to win the registered tasks [11] [2]. But, crowd workers often overestimate their productivity [18], and they register for more tasks than they can complete. It is reported that 82.9% of active worker in topcoder[1] drop their registered tasks, and 55.8% of the submissions are not valid [36]. Therefore, understanding opponent performance and trustworthiness is essential for crowd workers' decision-making processes.

The objective of this study is to propose a collaborative recommendation system to support crowd workers dynamic decision making to explore options and improve their probability of success in CSD. To this end, we first present a motivational example to highlight workers relation in CSD platform. Then we propose a collaborative recommendation system . This system seeks to increase workers probability of success in competing on different tasks in the platform. The system takes as input a list of open tasks in the platform, workers' collaboration and attributes and previous performances. It then recommends the list of suggested tasks to workers based on workers' specialty, and proficiency and probability of success.

The input of the proposed system is conducted on more than one year's real-world data from topcoder, the leading software crowdsourcing platform with an online community of over 1.5M workers and 55k mini-tasks . We applied the proposed recommendation system to 260 active crowd workers in our database during the

[1] https://www.topcoder.com/

weeks of Jan 14th 2015 to Jan 30th 2015. For workers just following the top three task success probability, workers can achieve success up to 86%.

The remainder of this paper is structured as follows. Section 2 introduces a motivational example that inspires this study. Section 3 presents background and related work. Section 4 outlines our research design. Section 5 discuses the results of this research. Section 6 presents the conclusion and outlines a number of directions for future work.

2 MOTIVATING EXAMPLE

Figure 1 depicts a motivating example with task selection, task requirement, task type and competition information among four workers and six tasks. The information on top of each task are task ID and task type and the information under each task are technical requirements to perform the task, Monetary Prize associated with tasks and task duration. All of the six tasks opened for competition in the same week when the four workers (i.e worker I, II, II, and IV) were available to take tasks at the same time. Table 3 summarises the four workers profile and previous performance. As it is illustrated in figure 1, Worker II registered for four tasks to compete on (i.e. tasks 1,2,5 and 6). These tasks are chosen from different tasks types of First2Finish, Code, and UI Prototype. Also they required combination of different technologies such as Java, HTML, and Java Script. Worker III registered for five tasks of 1, 2, 3, 4, and 5. Two of First2Finish, two of Assembly and one of Code type. While four of these tasks can be performed with knowledge of Java, the other one (i.e. task 5) requires HTML or Windows Server to be done. At the same time Worker IV registered for tasks 1, 5 and 6. While tasks 5 and 6 can be performed only by knowing HTML, task 1 requires some knowledge of Java. Furthermore, the three tasks belong to three different task types: First2Finish, Code, and UI Prototype. Interestingly, Worker I only registered for task 2, which requires Java and is under First2Finish type.

Is there any other task that Worker I can confidently register for?

The relation among the four workers and six tasks creates a network. In this network, Worker II and Worker III are common neighbors for Worker I. This means tasks that these workers registered for can be a potential task for Worker I to take. Given the similarity in Worker I's *proficiency* in Java, according to his previous registration based on table3, most probably tasks 1,3, and 4 are good fit for him to work on. Given his *specialty* in First2Finish, task 1 could be number one match for Worker I to compete on.

To register for a task, workers make the decision based on there previous performances, types of tasks they are interested, skill match to those required by the task, affordability in terms of time and effort required to complete a task, associate monetary prize, and reputation of other competitors on the task.

In this example, task 1 meets worker I preferences in terms of monetary prize, duration, specialty and proficiency (table 3. However, taking task 1 means that worker I will compete against worker II, worker III, and worker IV. knowing competitors preferences and performance history would enable worker I to predict his/her success level before make a decision on taking task 1.

This observation motivated us to propose a worker recommendation method based on a combination of collaborative filtering systems and workers individual preference and performance. This method can increase the probability of workers' success in performing tasks in the platform.

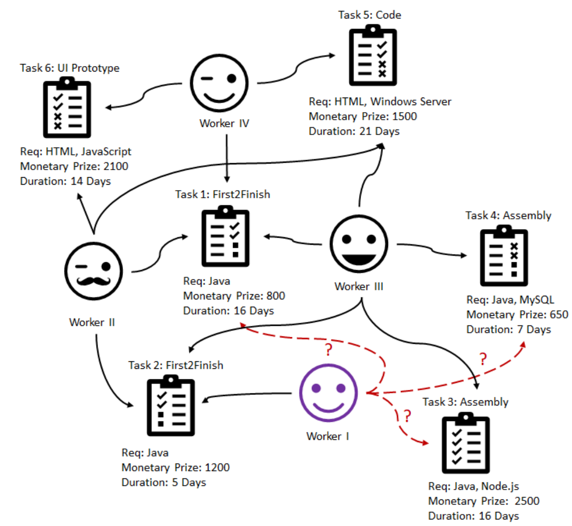

Figure 1: Overview of Motivating Example

3 BACKGROUND AND RELATED WORK

3.1 Workers Performance in Crowdsourcing

Software workers' arrival to the platform and the pattern of taking tasks to compete on, are factors that shape the worker dynamic in a crowdsourcing platform, however, the reliability in returning the qualified tasks creates the dynamic of the platform. Generally, not only would the monetary prize associated with the task influence the workers' interests in competitions[39], the number of registrants for the task, the number of submissions by individual workers, and certainly the workers' historical score rate would directly affect their final performance [22][30]. For newcomers or beginners, there is a time window required to improve and to develop into an active worker [11]. Therefore, it is typical that the workers need to communicate with the task owner in order to better understand the problems to be solved [21]. Existing studies show that over time, registrants gain more experience, exhibit better performance, and consequently gain higher scores [11] [3] [21]. Still, there are workers who manage not only to win but also to raise their submission-to-win ratio [10]. This motivate workers to develop behavioral strategies in topcoder [2] [3]. Moreover, the ranking mechanism used by topcoder contributes to the efficiency of online competition and provides more freedom of choice for higher rate workers in terms of controlling competition level [2].

3.2 Workers Competition in Crowdsourcing

To encourage contribution and engagement, crowdsourcing adopt the use of extrinsic rewards such as ranking as game elements for workers to compete in a non-gaming context [7]. Extrinsic rewards can increase the overall workers' engagement and commitment[4][5], motivation [6][32][33] and collaboration [13], since they address a type of social need for some community members[27].

In a CSD platform, a competitive environment not only influences the decisions of workers regarding which tasks to register and submit but also how they react to their peers. Such environment creates opportunities for workers to apply different strategies and assure their success and increasing their rank in the system. One primary example is rank-boosting [38][17] in Amazon Mechanical Turk, where workers mostly register for easy tasks or fake tasks that they themselves are uploading in order to increase their rating. Another example is detecting cheap talk phenomena [12][2] in topcoder. In CSD, higher rated workers have more freedom of choice in comparison with lower rated workers and can successfully affect the registration of lower rated workers. To assure a softer and easier competition level, higher rated workers are more likely to launch challenges against lower ones[16].

3.3 Workers Decision Making in Crowdsourcing

Online decision algorithms have a rich literature in operations research, economics, machine learning, and artificial intelligence, etc. Much of existing work on crowdsourcing decision making focuses on assigning reliable workers to existed task, such as learning worker quality and optimizing task assignment decisions [35], aggregating individual answers to improve quality [24], and worker incentives [20], developer recommendations [25][9], and understanding worker behaviors [37][2][40]. In software crowdsourcing, only a few studies have focused on decision support for software crowdsourcing market. Among them, Mao et al. [25] presented a content-based developer recommendation framework for CSD context, to recommend reliable workers based on static features extracted from participation history and winning history. Difallah et al.[9] propose a recommendation framework to pick a suitable crowd for a task. Yang et al. [36] introduces 'DCW-DS', an analytic-based decision support methodology to guide workers in acceptance of offered development tasks. Hettiachchi et al. [14] present 'Crowd-Cog',an online dynamic system that provides both task assignment and task recommendations, based on online cognitive tests to estimate worker performance across a variety of tasks. And Kumar et al.[1] proposed 'TasRec', a worker's fitment framework based on worker's preference, past tasks (s)he has performed, and tasks done by similar workers.

4 RESEARCH DESIGN

To develop a recommendation model from workers perspective, we use collaborative filtering method to increase workers proficiency and specialty in the suggested tasks. Then, we provide the probability of success per recommended task and report top three tasks with highest success probability per worker. This helps workers to have a higher confidence in making a decision to take the tasks. This architecture can be applied to any crowdsourcing platform; however, we focused on Topcoder as the target platform.

In this method, most suitable open tasks during two weeks from any point of time are suggested based on the degree of compatibility to workers' preferences in terms of monetary prize, duration, and workers' skill set in terms of proficiency, and specialty level;

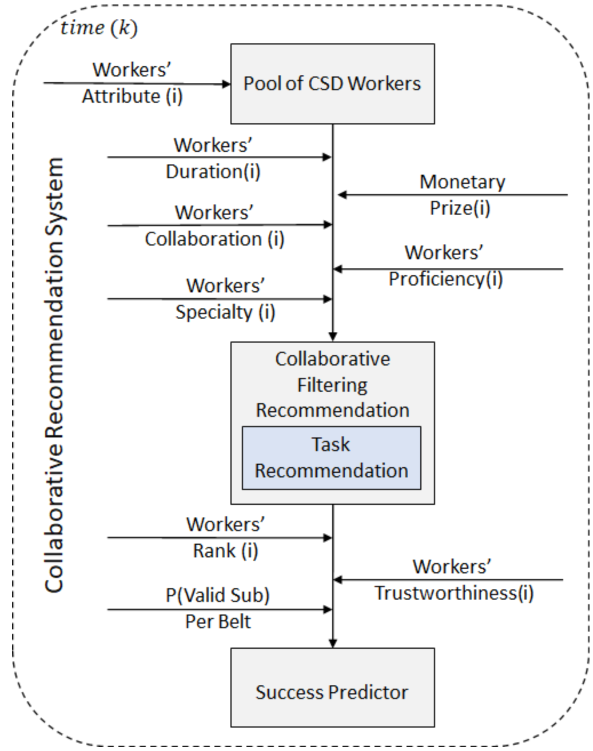

Figure 2: Overview of Collaborative Recommendation System Architecture

then, a worker could choose to compete in the proper tasks based on the probability of success in making a valid submission for the worker and the other registrants of the tasks. Figure 2 presents the overview of the collaborative recommendation system architecture. Workers' attributes are uploaded in the *pool of workers*. From this pool, *workers' rank* based on Topcoder definition[30], workers' preferences in terms of *monetary prize*, *duration*, *workers' collaboration*, *workers' specialty*, *workers' proficiency*, and *workers' trustworthiness* are calculated. The calculated metric is used as an input to *collaborative filtering system* which provides a list of *recommended tasks* to the workers as an output. Then, *worker success predictor* analyzes workers' probability of success per recommended task and ordered the top three recommendations.

4.1 Crowd Workers

According to Howe [15], crowd workers are a large and undefined group of skilled workers who have access to a task via the internet. In this research, we followed the definition of CSD workers in [28].

4.1.1 Workers' Attributes. Worker attributes are the subset of workers characteristics that dynamically influences successful task delivery per community. The list of Worker attributes applied in this study is summarized in Table 1.

4.1.2 Workers' Monetary Prizes. The minimum monetary prize, (bMP_i), associated to the list of tasks performed by Worker(i), W_i, is used as the base monetary prize for the recommended tasks to the worker.

Table 1: Summary of Metrics Definition

Type	Metrics	Definition
Workers Attributes	# Registration (R)	Number of registrants that are willing to compete on total number of tasks in specific period of time.
	# Submissions (S)	Number of submissions that a task receives by its submission deadline in specific period of time.
	# Valid Submissions (VS)	Number of submissions that a task receives by its submission deadline and passed the peer review and labeled as either completed or active.
Workers-Tasks Attributes	Worker registration date (WR)	Date and time that a worker registered for a task. (mm/dd/yyyy)
	Worker submission date (WS)	Date and time a worker submitted for a task. (mm/dd/yyyy)
	Duration (D)	total available days from worker registration date (TR) to submissions date (TS)
	Task Status	Completed or failed task.
	Task Type (Type)	Type of task competition.
	Task Competition Level (TC)	Total number of workers to register and are willing to compete on a task.
	Monetary Prize (MP)	Monetary prize (USD) offered for completing the task and is found in task description. Range: (0, ∞)
	Technologies (Tech)	Number of technologies used in task.
	Task Duration (D)	total available days from task registration start date (TR) to submissions end date (TS)

$$bMP_i = \min_{i=0}^{n} (MP_i) \tag{1}$$

4.1.3 Workers' Duration . Total available days from worker registration date (WR_i)to submissions date (WS_i) is the duration (D_i). In this research, the minimum value in list of duration periods worker i has in his/her profile is used as the base duration,(bD_i), for the recommended task.

$$D_i = (WR_i - WS_i) \tag{2}$$

$$bD_i = \min_{i=0}^{n} (D_i) \tag{3}$$

Table 2: Summary of Different Workers' Rank in Topcoder

Workers' Belt		Rating Range(X)	Workers%	p(VS)
Lower Experienced	Gray	$X < 900$	90.02%	0.25
Average Experienced	Green	$900 < X < 1200$	2.88%	0.45
	Blue	$1200 < X < 1500$	5.39%	0.39
Higher Experienced	Yellow	$1500 < X < 2200$	1.54%	0.6
	Red	$X > 2200$	0.16%	0.6

4.1.4 Workers' Rank. Topcoder adopts a numeric worker rating system based on Elo rating algorithm[2], as a result topcoder workers are divided into 5 groups. The 5 worker groups are defined as 5 belts of Red, Yellow, Blue, Green, and Gray, which corresponds to the highest skillful workers to the lowest ones[30]. Table 2 summarizes the distribution of workers belonging to different rating belts, (WB), in the Topcoder dataset used in this study. It is shown that among the total of 5062 active workers, more than 90% of the workers are in Gray belt, which is the non-experienced group. The other 10% of workers are more experienced solid workers.

4.1.5 Workers' Proficiency. To understand workers' proficiency in performing a task, we need to realize if the worker's skills match to those required by the task or not.

Worker proficiency level PL_i is indicative of the preferences of worker i for the set of technologies required by the recommended task. For a specific technology t, the proficiency is calculated as the ratio of the frequency of worker i's participation in the technology t, ($WTech_{i,t}$) , to the frequency of all the technologies worker i has used ($WTech_{i,j}$).

$$PL_{i,t} = \frac{WTech_{i,t}}{\sum_{j=1}^{n} WTech_{i,j}} \tag{4}$$

When Worker i's proficiency is calculated based on N different technologies required by task r, average value of $PL_{i,r}$ is utilized as workers' proficiency.

$$APL_{i,r} = \frac{\sum_{t=1}^{N} PL_{i,t}}{N} \tag{5}$$

4.1.6 Workers' Specialty. Workers' specialty level $SL_{i,r}$ is an indicator for worker i's preferences for a specific task type r. Therefore, workers' specialty level $SL_{i,r}$ is defined as the ratio of the frequency

[2]https://www.topcoder.com/member-onboarding/understanding-your-topcoder-rating

of worker i registration for the task type r , ($WType_{i,r}$) to the frequency of all the different task types, j, that worker i registered for ($WType_{i,j}$).

$$SL_{i,r} = \frac{WType_{i,r}}{\sum_{j=0}^{n} WType_{i,j}} \tag{6}$$

Algorithm 1 Collaborative Filtering Algorithm

I = set of all workers
WC_i = set of collaborators to worker i
WT_i = set of tasks registered by worker i
$Recom_i$ = set of tasks recommended to worker i
$poten_{ki}$ = set of tasks registered by worker k and not registered by worker i
$cond$ = set of conditions in eq()

for $i \in I$ **do**
  $WC_i = \emptyset$
  for $j \in (I/\{i\})$ **do**
    if $WT_i \cap WT_j \neq \emptyset$ **then**
      $WC_i = WC_i \cup$ j
    end
  end
  $Recom_i = \emptyset$
  for $k \in WC_i$ **do**
    $poten_{ki} = WT_k - WT_i$
    for $r \in poten_{ki}$ **do**
      $cnt = 0$
      for $condition \in \{condition_1, condition_2, condition_3\}$ **do**
        if $condition == TRUE$ **then**
          cnt = cnt + 1
        end
      end
      if $cnt \geq 2$ & $condition == TRUE$ **then**
        $Recom_i = Recom_i \cup r$
        $SL\_set_i = SL\_set_i \cup SL_{i,r}$
        $APL\_set_i = APL\_set_i \cup APL_{i,r}$
      end
    end
  end
  MX_SL_i = max(SL_set$_i$)
  MX_APL$_i$ = max(APL_set$_i$)
  for $r2 \in Recom_i$ **do**
    Proficiency = APL$_{i,r2}/MX_APL_i$
    $Specialty = SL_{i,r2}/MX\_SL_i$
    if *Proficiency > 0.5 & Specialty > 0.5* **then**
      Label$_{i,r2}$ = very strong recommend **end**
    if *Proficiency > 0.5 & Specialty < 0.5* **then**
      Label$_{i,r2}$ = strong recommend **end**
    if *Proficiency < 0.5 & Specialty > 0.5* **then**
      Label$_{i,r2}$ = recommend **end**
    if *Proficiency < 0.5 & Specialty < 0.5* **then**
      Label$_{i,r2}$ = low recommend **end**
  end
end

4.2 Collaborative Recommendation

Collaborative filtering recommendation [34] uses similarities between workers' profile and the history of their performances to provide serendipitous recommendations. To create the workers' collaborative filter, we use the workers collaboration as an input, then we add workers proficiency and specialty level, workers monetary prize and duration to narrow down the match between the recommended tasks and workers suitability.

In details as it is shown in Algorithm 1, system searches among the list of registrants of the tasks which was registered by worker i (W_j), if worker j (W_j) was registered for at least one of the tasks in the list then worker j is added to the set of worker i's collaborators, (WC_i). Automatically, all the tasks that were registered by worker j and not worker i is added to the list of potential tasks ($Poten_{k,i}$) for worker i. Then system analyzes the potential tasks based on worker$i$'s preferences in terms of monetary prize, task duration, proficiency (APL_i), and specialty ($SL_{i,r}$). Task z, t_z, should meet 3 out of 4 below conditions in order to be added to list of $workeri$'s recommended tasks, ($Recom_i$):

Condition 1. $bMP_i <= MP_z$
Condition 2. $bD_i <= D_z$
Condition 3. $APL_{i,r} > \alpha$
Condition 4. $SL_{i,r} > \beta$

α and β represent thresholds compared to which the minimum values of the proficiency ($APL_{i,r}$) and specialty ($SL_{i,r}$) should be higher. These information is chosen by the worker based on the characteristics of the task the worker has registered for. In this research we set both α and β to 30%. The recommended tasks will be mapped to two sets of values for specialty level ($SL - set_i$) and average proficiency level ($APL - set_i$) as an interval between 0 to 1. In next steps systems used the mapped data provides set of recommended tasks recommendation with four labels of *"very high recommend"*, *"high recommend"*, *"recommend"*, and *"week recommend"* per each active worker.

- *Very strong recommend*: is the list of recommended task that worker's proficiency and specialty level are greater than 50%;
- *Strong recommend*: is the list of task that workers has the specialty level less than 50% but the proficiency level is greater than 50%;
- *Recommend*: is the list of task that workers has the specialty level greater than 50% but the proficiency level is less than 50%;
- *Low recommend*: is the list of recommended task that worker's proficiency and specialty level are less than 50%.

4.3 Success Predictor

Making a submission by itself does not guarantee workers' success, in order to be successful in a CSD platform, it is important to make a valid submission. However, making a submission by higher ranked workers may impact on workers decision on attempting to submit[2][30]. It is important for a worker to be able to evaluate competitor's probability and trustworthiness in making a valid submission and as the result predict their own success.

4.3.1 Specialty Participation Ratio.

Specialty Participation, SP_i, represents worker i's registration frequency in tasks with the same type as the recommended task z. Therefore, specialty participation ratio, SPR_i, defines as the ratio of specialty participation per worker, sP_i, to the total registration frequency in tasks with a similar type by all the workers to whom the task z is recommended to $\sum_{j=0}^{N_w} TP_j$.

$$SPR_i = \frac{SP_i}{\sum_{j=1}^{N_w} SP_j} \tag{7}$$

4.3.2 Average Proficiency Experience Ratio $APER_i$.

Proficiency experience ratio of worker i, $PER_{i,k}$, represents the level of proficiency in technology requirement k to perform the recommended task z in compare with the list of opponents who compete on the task z. It defines as the ratio of the frequency which the worker i has utilized the technology k to the total utilization of technology$k$ by the workers to whom task z is recommended to, (N).

$$PER_{i,k} = \frac{PE_{i,k}}{\sum_{j=1}^{N_w} PE_{j,k}} \tag{8}$$

In the case that the recommended task z required multiple technologies, the average proficiency experience ratio $APER_i$ will be used.

$$APER_i = \frac{\sum_{j=1}^{N_{tsk}} PER_{j,k}}{N_{tsk}} \tag{9}$$

4.3.3 Workers' Trustworthiness.

To analyze workers' trustworthiness we expand topcoder definition of reliability. In topCoder, crowd worker's reliability of competing on the tasks is measured based on last 15 competitions workers registered and submitted. For example, if a worker submitted 14 tasks out of 15 last registered tasks, his reliability is 93% (14/15).

Therefore, worker trustworthiness level, TL_i, in this research measures ratio of number of valid submissions, VS_i, for worker i to register tasks up to the last 15 registered tasks $R15_i$.

$$TL_i = \frac{\sum_{i=1}^{15} VS_i}{\sum_{i=1}^{15} R15_i} \tag{10}$$

4.3.4 Probability of Valid Submission per Pelt.

Probability of valid submission per belt, $p(VS)_m$, measures the probability of worker i from belt b makes a valid submission.

$$p(VS)_b = \frac{\sum_{b=1}^{n} VS_b}{\sum_{b=1}^{n} R_b} \tag{11}$$

4.3.5 Probability of Success.

Probability of success for worker i, $p(Su)_i$, who registers for a task is a function of the worker's specialty participation ratio SPR_i, average proficiency experience ratio $APER_i$, Trustworthiness TL_i, and probability of valid submission $p(VS)_i$, as below:

$$p(Su)_i = \frac{SPR_i * APER_i * TL_i * p(VS)_i}{\sum_{i=1}^{N_w} SPR_j * APER_j * TL_j * p(VS)_j} \tag{12}$$

5 EXPERIMENT DESIGN

To evaluate the conceptual model introduced in Section 4, this section presents the experiment design and evaluation base line for this study.

5.1 Research Questions

To investigate the impact of workers' collaboration, proficiency and specialty in workers' success, the following research questions were formulated and studied in this paper:

RQ1 (Overall worker Performance): How distributed are crowd workers in terms of expertise?

This research question aims at providing general overview of workers distribution in the platform based on their proficiency and specialty in performing tasks;

RQ2 (Worker recommendation): How to strategically take a new task to ensure workers' success?

Understanding opponents performance pattern in a competition helps to provide better decision making for taking a task.

5.2 Dataset

The dataset from topcoder contains 403 individual projects including 4,907 component development tasks (ended up with 4,770 after removing tasks with incomplete information) and 8,108 workers from January 2014 to February 2015 (13 months). Tasks are uploaded as competitions in the platform, where crowd software workers would register and complete the challenges. When the workers submit the final files, it will be reviewed by experts to check the results and grant the scores.

The dataset contains tasks attributes such as required technology, platform, task description, task status, monetary prize, days to submit, registration date, submission date, and workers attributes such as registration date, submission date, valid submission, winning placement, winning status, rating score, and winning score. In this step, workers' rank, proficiency, specialty and trustworthiness metrics are not included. Then, we create attributes, such as workers skillsets (WTech), workers task type (WType), which are proxy by the number of technologies (#Tech) required to perform taken tasks by the worker, and different task type worker can choose to work on. We create binary variables for each technology (WTech) required in each task, and each task type (WType) worker chose to perform, where:

$$WTech(x,p) = \begin{cases} 1 & \text{worker x is an expert in technology p} \\ 0 & \text{otherwise} \end{cases}$$

and,

$$WType(x,s) = \begin{cases} 1 & \text{worker x is a specialist in task type s} \\ 0 & \text{otherwise} \end{cases}$$

The worker attributes used in the analysis are presented in top section of Table 1.

5.3 Implementation of The Collaborative Recommendation System

There are six steps in implementing the collaborative recommendation system: workers' collaboration, workers' monetary prize, workers' duration, workers' proficiency, workers' specialty and workers' success predictor.

5.3.1 Workers' Collaboration: Workers' Collaboration is analyzed based on the bipartite network of workers. If two workers registered on the same tasks, they are collaborating. In this research, first we grouped workers' attribute per month and labeled workers who had minimum one registration per month as active worker. This reduced the number of workers from 8180 to 2259 workers. Then we analyzed workers' collaboration based on the entire dataset (i.e 13 months from Jan 2014 to Feb 2015) explained in part 5.2.

5.3.2 Workers' Monetary Prizes: Worker's monetary prize is analyzed based on the minimum monetary prize in the history of tasks worker i registered for.

for example, worker I in part 2has the history of performing on tasks with the minimum monetary prize of 750$. Among available tasks that worker I can potentially take, tasks 1 and 3 are offering higher monetary prize than worker's monetary prize (i.e 800$ and 2500$ respectively) and can be labeled as potential recommended task.

5.3.3 Workers' Duration: In this research, duration among the tasks worker i compete on is used as the base duration for recommendation. For instance worker I in part 2 has the history of taking tasks with duration as short as 7 days. Therefore, in terms of duration, all the 3 tasks meet worker I's preference, figure1, and system will labeled them as potential recommended task.

5.3.4 Workers' Proficiency: Workers' proficiency is designed to analyze the match between workers' skill set and tasks requirement per technology. For example, if task (l), T_l, listed *HTML* as one of the requirements and worker(i), W_i, compete on a task/s with similar requirements out of total b tasks in workers' history, then W_i has a/b proficiency level to perform T_l.

In this research the workers' proficiency calculated based on binary metrics $Wtech_i$ and provide the percentage of match between workers and potential suggested tasks. According table 3 to register for task 1, Worker I has proficiency level of 0.307(i.e (100/(100+67+60+67+32))) in Java.

5.3.5 Workers' Specialty: Workers' Specialty measures the level of the match between a task and workers historical performance on similar task type. For example, if task(l), T_l, labeled as type *code* in competitions and worker(i), W_i, compete on c task/s with similar task type out of total b tasks in workers' history, then W_i has c/b Specialty level to perform T_l.

To analyze the specialty match this research used the introduced metric $WType_i$ in part 5.2. For example, based on table 3 to register for task 1, Worker I has specialty level of 0.30 (i.e 100/(100+42)) to perform First2Finish task type.

5.3.6 Workers' Success Predictor: To predict workers success in taking the recommended tasks , this research analyzed workers probability of success based on the registered opponent probability of making a valid submission for the task.

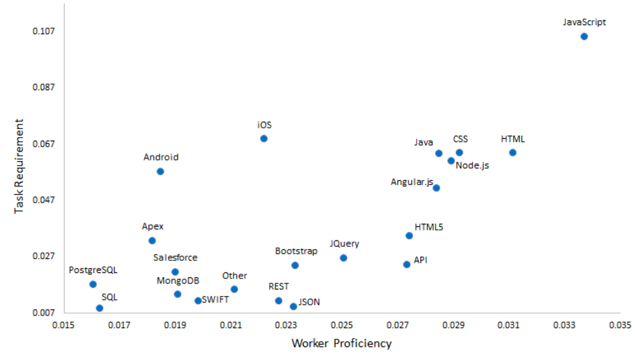

Figure 3: Overall Distribution of Workers' Proficiency per Technology

6 RESULT AND DISCUSSION

6.1 Overall Worker Performance

In order to have a better understanding of workers' overall performance, we studied overall distribution of workers' proficiency and workers' specialty based on required technology by tasks and tasks type in the platform.

6.1.1 Workers' Proficiency: We identified 115 technologies in the dataset. Mapping among technologies, task and workers showed that 50% of tasks can be done by knowledge of only 8 technologies, and 50% or workers are actively taking tasks with requirements of one of the top 21 technologies. Therefor in this part we visualize the workers' proficiency for the top 21 technologies in the dataset. According to figure 3, the most on demand technology is JavaScript with 3.4% of workers' proficiency and almost 11% of available tasks to compete on, the second technology is HTML with 3.1% workers' proficiency and 6% of tasks as requirements. CCS is the third technology that contains 3% workers proficiency. Interestingly, the technology with the least level of proficiency is PostgreSQL followed by SQL with almost 1.6% workers' proficiency. The rest of technologies attract workers with on average 2.3% proficiency and provide a pool of task with requirement technology between 3.2% to 6%.

6.1.2 Workers' Specialty: Topcoder introduced 14 task type to compete on. Mapping among technologies, task and workers showed that 50% of workers are interested in Assembly Competitions, assembling previous tasks, and Code, Programming specific task, while 50% of tasks are under First2Finish type, in which The first person to submit passing entry wins.

As it is illustrated in figure 4, workers' specialty for First2Finish task type is 26%, while for Assembly and Code is 29% and 28% respectively. The next task type is UI Prototype Competition with 7% workers' specialty and almost 6% task in the platform. The rest of tasks types are attracting workers with less than 3% specialty.

Table 3: Summary of Workers' Profile Overview in Motivating Example

Worker ID	# R	# VS	# Collab-orators	Belt	Min(MP)	Min(D)	Top 5 Proficiency	Top 2 Specialty
Worker I	244	80	61	Green	750	7	Java(100), SQL(67), Android(60), Apex(67), .NET(32)	UI Prototype Competition (100), First2Finish(42)
Worker II	150	87	80	Gray	890	23	Android (60), MongoDB (40), Node.js(32), JavaScript(30), Java(16)	First2Finish(96), Code(80)
Worker III	883	32	69	Yellow	1000	10	Java(300), IOS(210), MOngoDB(163), Angular.js(80), Node.js(65)	Assembly Competition (300), First2Finish (180)
Worker IV	195	14	85	Blue	1250	16	Java(70), IOS(56), MOngoDB(50), .NET (45), Node.js(45)	Assembly Competition (78), UI Prototype Competition (60)

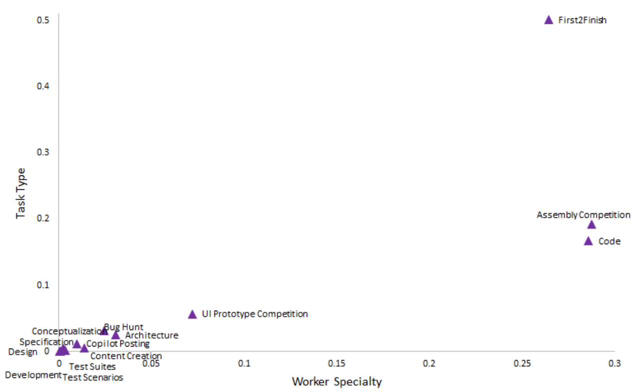

Figure 4: Overall Distribution of Workers' Specialty per Task Type

6.2 Worker Recommendation

We trained the proposed recommendation system based on 11 months data from our data set s (from Jan 2014 to Jan 2015). Then we recommended task to active workers during Jan 14^{th} 2015 to Jan 30^{th} 2015. According to our dataset there are 260 active workers and 311 available tasks in the platform during this period. The initial result of the recommendation system provide at least 1 recommended tasks for 237 workers, and no suitable tasks for 23 workers. Also workers would succeed with average probability of 23%. The maximum probability of success was 86% and minimum was 2%. 165 workers(i.e 57%) would receive success will probability of success less than average.

In order to understand how the system works, we picked workers from part 2 and explain all the steps system takes. As it is summarized in table 3, *worker I* is from Green belt, competed on 244 tasks with 61 collaborators. Worker II belongs to gray rank, with 150 competition in the profile and 80 collaborators. *Worker III* is one of high ranked workers from Yellow community, who competed 883 times with 69 collaborators. and finally *worker IV* is a blue belt with 195 competition and 85 collaborators.

6.2.1 *Collaborative Recommendation.*

As it will be explained in details in this part, recommendation system makes recommendation from 19 tasks to 4 chosen worker to register for next 14 days. Each workers received a list of recommended task between 11 to 18 task. Each task is labeled as one of the recommendation levels introduced in part 4.2.

Worker I: As figure 5 represents, the recommendation system recommends 12 tasks to worker I to take (i.e tasks 1,2,3,4,5,6,7,8,9,10,11,12). Task 1, 2 and 4 are under low recommended section, which mean worker I's proficiency and specialty in performing this tasks is under 50%. Task 5 is located in the recommended section. This means however worker I does not have high level of proficiency to perform this tasks but s/he has is specialized to do so. Task 3 is in strong recommended which relates to high level of proficiency of worker I to perform this task, there are 7 tasks (6,7,8,9,10,11 and 12) that are very strongly recommended to be taken by worker I.

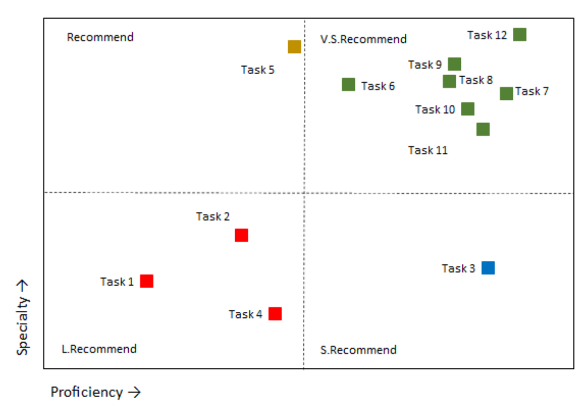

Figure 5: Magic Quadrant for Worker I Recommended Tasks

Worker II: As figure 6 presents, Worker II is recommended to register for 11 tasks (tasks 3,6,7,8,9,10,11,12,13,14,15). From these 11 tasks, tasks 6,13,14 and 15 are labeled as low recommended. Worker II does not received any recommended task under recommended section. S/he has tasks 8 and 9 under strong recommended with high proficiency and low specialty. And s/he has 5 suggested tasks in the very strong recommended section to take for next tow weeks(i.e tasks 3,7,10,11 and 12).

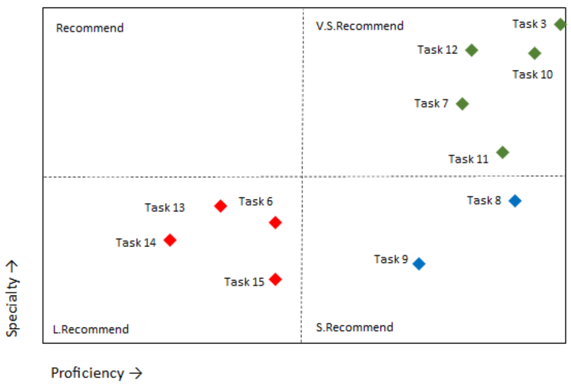

Figure 6: Magic Quadrant for Worker II Recommended Tasks

Worker III: Figure 7 illustrates the magic quadrant for worker III's recommended tasks. According to figure 7 , 17 tasks are recommended to worker III (tasks 3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19). 4 out of 17 tasks are under low recommended (task 3) or recommended (tasks 11,16,18) section. Tasks 5,7,9,8,and 12 are located under the strong recommended tasks and tasks 4, 10, 13, 14, 15, 16, 17 and 19 are very strongly recommended to worker III to take them based on her/his proficiency and specialty level.

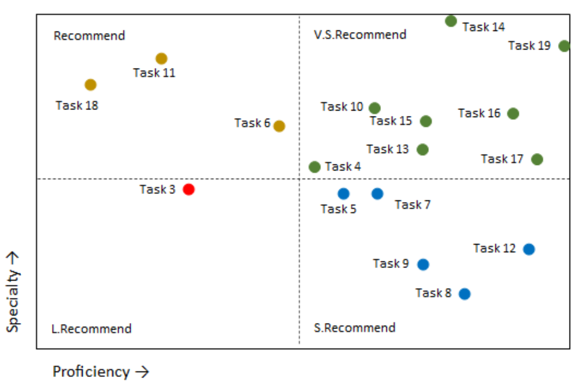

Figure 7: Magic Quadrant for Worker III Recommended Tasks

Worker IV: As it is clear in figure 8 The recommendation system provide 18 tasks to worker IV to potentially register for, tasks 1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18 and 19. 4 out of 17 tasks are under low recommended (task 3) or recommended (tasks 11,16,18) section. While 15 out of 18 tasks are labeled under recommended or low recommended, there is no task under strong recommended and only 3 tasks of 6, 12 and 13 are very strongly recommended to be taken by worker IV.

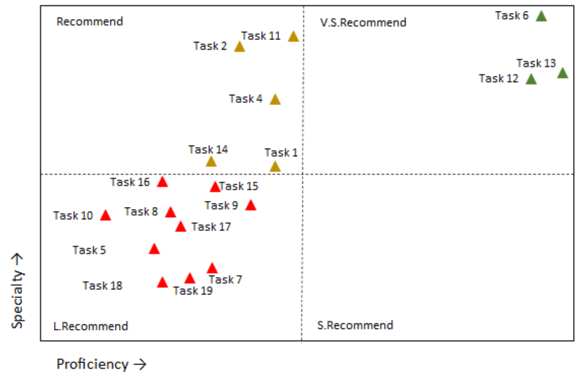

Figure 8: Magic Quadrant for Worker IV Recommended Tasks

6.3 Success predictor

It is not practical to register and compete on minimum 12 tasks during two weeks, therefore, system calculate the probability of success per recommended tasks for the workers based on registered opponents to compete on tasks and suggest the top three tasks to workers. Table 4 presents the order of tasks per worker.

As it is summarized in table4 worker I has the highest chance of success in taking task 12 among all the list of recommended tasks with 30% probability of success, the second highest chance happens when worker I takes either task 8 or task 9, both tasks provide 15% probability of success for worker I. For worker II task 8 provides the highest probability of success of 43%, Task 9 provides 40% probability of success and task 10 brings 39% probability of success. Interestingly, the top two highest probability of success for worker II come with tasks from strong recommended group, meaning worker II has high proficiency in technical term to perform the tasks, while the specialty in task type is not very high (i.e. under 50%). Worker III has 58% chance of success with registering for task 17 followed by 56% probability of success for task 16. both tasks are under very strongly recommended group. The probability of success drops to 50% for task 18 as the third suggested task from recommended cluster. The last worker (i.e worker IV) will have the highest probability of success by competing on task 6 for17% chance, followed by 16% probability of success for task 13. and the third task with highest probability of success is task 2 with 10% probability of success which is under recommended group of tasks for worker IV.

6.4 Validation

The last step was to understand the accuracy of the recommendation system for different workers. To do so, the mean relative error (*MRE*) of each workers' list of recommended task ($Recom_i$) based on the available actual workers' task registration (R_i) in the same

Table 4: Final Task Suggestion to Workers

Worker ID	Task Order	Recommended Level	p(success)
Worker I	Task 12	V.S.R	30%
	Task 8	V.S.R	15%
	Task 9	V.S.R	15%
Worker II	Task 8	S.R	43%
	Task 9	S.R	40%
	Task 10	V.S.R	39%
Worker III	Task 17	V.S.R	58%
	Task 16	V.S.R	56%
	Task 18	R	50%
Worker IV	Task 6	V.S.R	17%
	Task 13	V.S.R	16%
	Task 2	R	10%

period of time (Jan 14$^{\text{th}}$ 2015 to Jan 30$^{\text{th}}$ 2015) was calculated, where y is number recommended task to worker$i$, and n is number of actual tasks that worker i registered for as displayed below.

$$MRE_i = \frac{\sum_{i=0}^{n} R_i - \sum_{i=0}^{y} Recom_i}{\sum_{i=0}^{n} R_i}$$

The t-test was also applied to the prediction results in each state to confirm the models' accuracy.

The mean relative error (MRE) of the recommendation system was only 1.9% . The results of the t-test on 311 tasks that recommended to 260 workers revealed that the probability of error in recommendation system was 0.012 with zero hypothesized mean difference. This result shows that the proposed recommendation system is performing accurately.

6.5 Discussion

Recommending a task is dynamic in nature since new tasks are uploaded all the time, and some other tasks are getting finished. Consequently, a proper recommendation needs to be regularly updated. Different factors impact a worker performance in CSD, to accurately support workers dynamic decision making process it is important to understand and capture the variation of these factors. The proposed collaborative recommendation system in this paper addresses the measurement of varying amount of proficiency and specialty with respect to required technology and task type and preferences of individual workers using monetary prize and duration. From workers perspectives, success-driven task recommendation systems helps to: 1) ensures success and increasing expertise of workers in short time, and 2) guides the healthy growth of competition for tasks with respect to the new and emerging technologies. Even though crowd workers try to increase their proficiency level in new technologies, they may not pay enough attention to demanding technologies [19].

6.6 Threats to Validity

First, the study only focuses on competitive CSD tasks on the topcoder platform. Many more platforms do exist, and even though the results achieved are based on a comprehensive set of about 60,000 task-worker, the results cannot be claimed externally valid. There is no guarantee the same results would remain exactly the same in other CSD platforms.

Second, there are many different factors that may influence workers' preference, performance and decision in task selection and completion. Our worker collaboration approach is based on known task-worker attributes in topcoder. Different approaches may lead us to different but almost similar results.

Third, the result is based on network of tasks-workers only. Workers communication was not considered in this research. In future we need to add this level of research to the existing one.

6.7 Adaptability to Different Platforms

The overall presented collaborative recommendation system in this paper is adoptable to different crowdsourced platforms. However, based on the type of platform (i.e competitive, collaborative or coopetitive) the workers success predictor needs to be updated by different task validation and workers reliability algorithm adopted by the chosen platform. To make the presented system compatible for different crowdsourcing platforms, there is no need to update any part of the collaborative recommended system but the back end analysis in elaborating defined input metrics.

7 CONCLUSION AND FUTURE WORK

Crowdsourced Software Development (CSD) is an emerging paradigm that has been increasingly adopted. In a competitive crowdsourcing marketplace, competition for success over shared demand adds develops uncertainty in decision making process for crowd workers. Therefor, it is critical for a crowd worker to not only understand the suitability of available tasks to take, but also the sensitivity and performance of the opponents in taking the tasks and rate of success. This paper reports a collaborative recommendation system for workers to address that end.

The proposed collaborative recommendation system nominates tasks with high suitability to workers based on a set of workers attributes such as workers' proficiency and specialty in performing the task, workers' trustworthiness in make a valid submission, minimum monetary prize and minimum task duration, and workers' collaboration with other opponents. Then the system provide probability of success per recommended task based on other workers who received the task in their recommendation list(potential opponents).

The proposed collaborative recommendation system empowers crowd workers to explore different potential success strategies with respect to different available tasks and active opponents in the platform. This includes the probability of workers success, task duration, monetary prize, task type and required technology. Moreover, experimental results on Experimental results on 260 active crowd workers demonstrate that just following the top three success probability of task recommendations, workers can achieve success up to 86%.

In future, we would like to expand our system and add workers communication and relation to the system which develops facilitating techniques to support decisions during task scheduling phase.

REFERENCES

[1] Kumar Abhinav, Gurpriya Kaur Bhatia, Alpana Dubey, Sakshi Jain, and Nitish Bhardwaj. 2020. TasRec: a framework for task recommendation in crowdsourcing. In *Proceedings of the 15th International Conference on Global Software Engineering*. 86–95.

[2] Nikolay Archak. 2010. Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder. com. In *Proceedings of the 19th international conference on World wide web*. 21–30.

[3] Nikolay Archak and Anindya Ghose. 2010. Learning-by-doing and project choice: a dynamic structural model of crowdsourcing. *LEARNING* 1 (2010), 1–2010.

[4] Sanat Kumar Bista, Surya Nepal, Nathalie Colineau, and Cecile Paris. 2012. Using gamification in an online community. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. IEEE, 611–618.

[5] Huseyin Cavusoglu, Zhuolun Li, and Ke-Wei Huang. 2015. Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community. In *Proceedings of the 18th ACM conference companion on computer supported cooperative work & social computing*. 171–174.

[6] Dana Chandler and Adam Kapelner. 2013. Breaking monotony with meaning: Motivation in crowdsourcing markets. *Journal of Economic Behavior and Organization* 90 (2013), 123–133. https://doi.org/10.1016/j.jebo.2013.03.003

[7] Ruijia Cheng and Mark Zachry. 2020. Building Community Knowledge In Online Competitions: Motivation, Practices and Challenges. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–22.

[8] Matthew JC Crump, John V McDonnell, and Todd M Gureckis. 2013. Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research. *PloS one* 8, 3 (2013).

[9] 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*. 367–374.

[10] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2016. Scheduling human intelligence tasks in multi-tenant crowd-powered systems. In *Proceedings of the 25th international conference on World Wide Web*. 855–865.

[11] Siamak Faradani, Björn Hartmann, and Panagiotis G Ipeirotis. 2011. What's the right price? pricing tasks for finishing on time. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[12] Joseph Farrell and Matthew Rabin. 1996. Cheap talk. *Journal of Economic perspectives* 10, 3 (1996), 103–118.

[13] Mary L Gray, Siddharth Suri, Syed Shoaib Ali, and Deepti Kulkarni. 2016. The crowd is a collaborative network. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. 134–147.

[14] Danula Hettiachchi, Niels Van Berkel, Vassilis Kostakos, and Jorge Goncalves. 2020. CrowdCog: A Cognitive skill based system for heterogeneous task assignment and recommendation in crowdsourcing. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–22.

[15] Jeff Howe. 2008. *Crowdsourcing: How the power of the crowd is driving the future of business.* Random House.

[16] Zhenghui Hu and Wenjun Wu. 2014. A game theoretic model of software crowdsourcing. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering*. IEEE, 446–453.

[17] Panos Ipeirotis. 2010. Be a top mechanical turk worker: You need $5 and 5 minutes. *Blog: Behind Enemy Lines* (2010).

[18] Magne Jorgensen and Stein Grimstad. 2005. Over-optimism in software development projects:" the winner's curse". In *15th International Conference on Electronics, Communications and Computers (CONIELECOMP'05)*. IEEE, 280–285.

[19] Muhammad Rezaul Karim, Ye Yang, David Messinger, and Guenther Ruhe. 2018. Learn or earn?-intelligent task recommendation for competitive crowdsourced software development. (2018).

[20] Nicolas Kaufmann, Thimo Schulze, and Daniel Veit. 2011. More than fun and money. Worker Motivation in Crowdsourcing-A Study on Mechanical Turk.. In *Amcis*, Vol. 11. Detroit, Michigan, USA, 1–11.

[21] Aniket Kittur, Jeffrey V Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. 2013. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*. 1301–1318.

[22] Karim R Lakhani, David A Garvin, and Eric Lonstein. 2010. Topcoder (a): Developing software through crowdsourcing. *Harvard Business School General Management Unit Case* 610-032 (2010).

[23] Jan Marco Leimeister, Michael Huber, Ulrich Bretschneider, and Helmut Krcmar. 2009. Leveraging crowdsourcing: activation-supporting components for IT-based ideas competition. *Journal of management information systems* 26, 1 (2009), 197–224.

[24] Andrew Mao, Ariel Procaccia, and Yiling Chen. 2013. Better human computation through principled voting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 27.

[25] Ke Mao, Ye Yang, Qing Wang, Yue Jia, and Mark Harman. 2015. Developer recommendation for crowdsourced software development tasks. In *2015 IEEE Symposium on Service-Oriented System Engineering*. IEEE, 347–356.

[26] Denisse Martinez Mejorado, Razieh Saremi, Ye Yang, and Jose E Ramirez-Marquez. 2020. Study on Patterns and Effect of Task Diversity in Software Crowdsourcing. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–10.

[27] Ganit Richter, Daphne R Raban, and Sheizaf Rafaeli. 2015. Studying gamification: the effect of rewards and incentives on motivation. In *Gamification in education and business*. Springer, 21–46.

[28] Razieh Saremi, Ye Yang, Gregg Vesonder, Guenther Ruhe, and He Zhang. 2021. CrowdSim: A Hybrid Simulation Model for Failure Prediction in Crowdsourced Software Development. *arXiv preprint arXiv:2103.09856* (2021).

[29] Razieh Lotfalian Saremi and Ye Yang. 2015. Dynamic simulation of software workers and task completion. In *2015 IEEE/ACM 2nd International Workshop on CrowdSourcing in Software Engineering*. IEEE, 17–23.

[30] Razieh L Saremi, Ye Yang, Guenther Ruhe, and David Messinger. 2017. Leveraging crowdsourcing for team elasticity: an empirical evaluation at TopCoder. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, 103–112.

[31] Mei Si and Stacy C Marsella. 2014. Encoding theory of mind in character design for pedagogical interactive narrative. *Advances in Human-Computer Interaction* 2014 (2014).

[32] Wael Soliman and Virpi Kristiina Tuunainen. 2015. Understanding continued use of crowdsourcing systems: An interpretive study. *Journal of theoretical and applied electronic commerce research* 10, 1 (2015), 1–18.

[33] Osamuyimen Stewart, Juan M Huerta, and Melissa Sader. 2009. Designing crowdsourcing community for the enterprise. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*. 50–53.

[34] Loren Terveen and Will Hill. 2001. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium* 1, 2001 (2001), 487–509.

[35] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems* 22 (2009), 2035–2043.

[36] Ye Yang, Muhammad Rezaul Karim, Razieh Saremi, and Guenther Ruhe. 2016. Who should take this task? Dynamic decision support for crowd workers. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–10.

[37] Ye Yang and Razieh Saremi. 2015. Award vs. worker behaviors in competitive crowdsourcing tasks. In *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–10.

[38] Bin Ye, Yan Wang, and Ling Liu. 2015. Crowd trust: A context-aware trust model for worker selection in crowdsourcing environments. In *2015 IEEE International Conference on Web Services*. IEEE, 121–128.

[39] Ming Yin, Yiling Chen, and Yu-An Sun. 2014. Monetary interventions in crowdsourcing task switching. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

[40] Hui Zhang, Yuchuan Wu, and Wenjun Wu. 2015. Analyzing developer behavior and community structure in software crowdsourcing. In *Information science and applications*. Springer, 981–988.