A Survey of General-Purpose Crowdsourcing Techniques

Anand Inasu Chittilappilly, Lei Chen, *Member, IEEE*, and Sihem Amer-Yahia, *Member, IEEE*

Abstract—Since Jeff Howe introduced the term *Crowdsourcing* in 2006, this human-powered problem-solving paradigm has gained a lot of attention and has been a hot research topic in the field of computer science. Even though a lot of work has been conducted on this topic, so far we do not have a comprehensive survey on most relevant work done in the crowdsourcing field. In this paper, we aim to offer an overall picture of the current state of the art techniques in general-purpose crowdsourcing. According to their focus, we divide this work into three parts, which are: incentive design, task assignment, and quality control. For each part, we start with different problems faced in that area followed by a brief description of existing work and a discussion of pros and cons. In addition, we also present a real scenario on how the different techniques are used in implementing a location-based crowdsourcing platform, gMission. Finally, we highlight the limitations of the current general-purpose crowdsourcing techniques and present some open problems in this area.

Index Terms—Crowdsourcing, incentive design, task assignment, quality control and result aggregation

1 Introduction

Crowdsourcing has gained a lot of attention since 2006 after Jeff Howe introduced this term in his article "The Rise of Crowdsourcing", published in *Wired* magazine [72]. According to Jeff Howe, crowdsourcing is the method of outsourcing a specific set of functions of a company to an undefined set of people, instead of assigning it to designated employees [71]. Unlike traditional office workers hired by the corporate and highly paid, with the help of crowdsourcing platforms like *Amazon Mechanical Turk (AMT)* and with the power of a crowd [1], a task can be completed by a set of workers called Internet workers for a small payment. These Internet workers can take up any job they want and finish it successfully to get the payment. Following the terms widely used in the crowdsourcing community, we term these Internet workers as turkers and crowdsourcing tasks as HITs, i.e., Human Intelligence Tasks.

Even though the buzzword crowdsourcing was coined in 2006, this technique was invented in early 18th century and has been in use for many years. In 1714, the British government offered a sum of £20,000 to those who can find a solution for so called *The Longitude Problem*, [23] which killed 1,000s of seamen a year because of the unavailability of the location coordinates to help sailors when the ship is damaged or when it gets stuck in an unknown island [115]. This is considered as the first crowdsourcing event and the winner was John Harrison, a self-taught English carpenter and also a clockmaker. In 1884, *Oxford English Dictionary* depended on a crowd to catalog words [94]. The current logo of Toyota was chosen from 27,000 logos designed in a logo designing competition held in 1936 [9]. We have various such crowdsourcing examples in the past. In the 21st century, the best example for crowdsourcing is *Wikipedia* which was started in 2003. *Wikipedia* acquires collective wisdom through crowdsourced knowledge. YouTube, started in 2005, is an example of crowdsourced entertainment. Similarly, there are a lot of other companies listed in Fortune 500, that have relied on a crowd for various tasks [110]. All these examples, show the power of the crowd and how revolutionary they can be.

One of the first, authentic and well-explained research articles about crowdsourcing was written by Prof. Daren C. Brabham in 2008. The paper titled "Crowdsourcing as a Model for Problem Solving: An Introduction and Cases", explains crowdsourcing, its potential uses by exploiting the crowd and also the cost involved. He also gives an insight to the research involved in this field [42]. The growth of the Internet and success of outsourcing are the main reasons for the attention to this human-powered problem-solving paradigm. Another reason for the tremendous growth of crowdsourcing within a short period is of course because of its inherited power of parallel processing and the fact that some intrinsic tasks like translation and labelling can be better done by humans than by machines. Each human task, even if it is quick, requires human time scales, and comes with some overhead to publish the corresponding HIT and tear it down. Comparatively, crowdsourcing enables issuing and solving multiple HITs in parallel, cutting down the wall-clock time. Labelling is an example for that where a number of people work on labelling images at the same time, which reduces the overall time required. As mentioned before, this method is much cheaper than the traditional method. For micro payments, there is a huge set of workers ready to do different types of tasks which not only decreases the overall expense but also helps to generate high-quality work. With the help of different quality assurance models and methods, crowdsourcing platforms ensure that the crowdsourced tasks are of high quality.

- *A.I. Chittilappilly and L. Chen are with the Hong Kong University of Science and Technology, Hong Kong. E-mail: {aicaa, leichen}@ust.hk.*
- *S. Amer-Yahia is with the National Center for Scientific Research, Grenoble Informatics Laboratory, Grenoble, France. E-mail: Sihem.Amer-Yahia@imag.fr.*

Manuscript received 8 Jan. 2016; revised 8 Apr. 2016; accepted 12 Apr. 2016. Date of publication 21 Apr. 2016; date of current version 3 Aug. 2016. Recommended for acceptance by J. Pei.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2555805

1041-4347 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

As mentioned earlier, crowdsourcing has been a hot research topic in Computer Science since 2006. Many crowdsourced applications have been introduced in the industry since then. Researchers have used the crowd to evaluate top-K and group-by queries [53], and sort and join [96], [129] operations. As the turkers are unknown to the requester, it is impossible to evaluate them before they take up a job. There could be genuine turkers as well as spammers in the workers' pool, from which they are chosen. So, it is necessary to have an additional step to filter out spammers from all crowdsourcing tasks to make sure that the quality of answers is good. Different research on crowdsourcing has led to the development of different technologies including [37], [66], [67], [92], and [113] to filter out cheaters. We can see some research done on data cleaning using crowdsourcing in [50], [120]. There is research conducted in the spatial crowdsourcing area including [48], [81]. Some research is conducted on entity resolution and schema matching [36], [55], [128], [130], [139]. There is also research conducted on query optimization [96], [103], [104], [105].

Different surveys have already been conducted on various topics of crowdsourcing which includes [134], [135], and [137]. Marcus and Parameswaran [95] has done a comprehensive study on different aspects of crowdsourcing. But these surveys merely mention about different research papers in a few sentence and do not compare any techniques used. The motivation to write this survey is that, as none of these work compare existing techniques in depth, it is necessary to have a comprehensive survey done which compares different techniques used with a detailed description.

2 Overview

Different steps involved in the process of crowdsourcing can be divided into three. They are: initializing, implementation and finalizing. Fig. 1 shows the control flow of this process with each step along with its sub-steps.

The first part is initializing, which includes all the steps done before a crowdsourcing task is given to the crowd. First, the task should be designed properly by the requester, who is like an employer in the traditional method. The requester also has to calculate the workforce needed. It is common in the field of Computer Science to follow a divide-and-conquer paradigm if the problem is too complex. Before giving a task to the public, if it is necessary, the requester has to break it down into sub tasks in such a way that the whole task is easily solved. The requester should also ensure that the execution of any task is not affecting any other tasks. Once the tasks are designed properly, incentives should be designed as the turkers will be signing up for the job based on the incentives. The importance of incentive design is discussed in Section 4.

The second part is implementation. Once the initializing step is over, the requester has to find a crowd for a specific task. The requester can use different crowdsourcing platforms like *Amazon Mechanical Turk* [1] or *CrowdFlower* [7] to find a crowd. Selection of the crowd is important as the quality of work depends on that. Based on the task requirement, the requester will select the turkers and the task will be assigned to them accordingly. In this paper, we use the terms turkers and workers interchangeably, which refer to the people who registered on a crowdsourcing platform and is willing to take crowdsourcing tasks.

Fig. 1. Framework of crowdsourcing.

The last step is finalizing, in which various steps are carried out to finalize the solution. After retrieving all the results from turkers, it is necessary to refine them by separating the answers given by cheaters from the answers given by genuine users, with the help of different quality control methods. After that process, answers are aggregated and submitted to find the correct answer which is the final output.

Depending on the tasks, these steps can be executed iteratively. After going through the above mentioned process once, the output will be the aggregated answers and that can be used as input in the next iteration. Amsterdamer et al. [32] and Sun et al. [119] are the best examples for using this method.

The most important challenges are summarized into four categories: retaining and motivating workers with proper incentives, recruiting workers based on task requirements, controlling the overall quality of responses and combining the answers given by the workers.

To address these four challenges, many models and methods have been proposed. This paper will give a brief study of different methods and technologies proposed by researchers in the field of crowdsourcing. The pros and cons of each technique are also compared and discussed. First we start by introducing some of the most used crowdsourcing platforms in Section 3. In Section 4, we discuss different types of incentives used and also the impact and need of each type. In Section 5, we discuss different techniques used for task decomposition and in Section 6, we discuss different methodologies used for task assignment. In Section 7, we discuss different quality control methods used to increase the effectiveness of crowdsourcing. Section also explains various techniques used to identify spammers from a pool of turkers. In Section 8, we have given a real case scenario of crowdsourcing and how different techniques are used to implement a crowdsourcing platform. Finally in Section 9, we have given some future research directions by showing the challenges still faced in crowdsourcing.

3 Crowdsourcing Platforms

There are many websites on the Internet which provide services for requesters to post different tasks or questions and for a crowd to complete them. These websites are called

Crowdsourcing platforms, and among all of them, *Amazon Mechanical Turk*(*AMT*) is the most popular one [1]. There are two types of crowdsourcing platforms, standalone platforms, which do not rely on any other platforms to execute the tasks and MetaPlatforms [27], where the systems rely on previously mentioned standalone systems. AMT is a standalone system and *CrowdFlower* depends on AMT by using a programmed layer. Each crowdsourcing platforms can be again classified as general purpose platforms and specialized platforms based on their purpose. AMT is a general crowdsourcing platform which allows the requester to post tasks of any type or from any background. Crowdsourcing platforms like *OpenStreetMap* and *Walkbase* have focused interests. In the following, we just highlight some of the popular crowdsourcing platforms. A detailed list of different platforms can be found at [3], [14], and [27]. Even though there are different platforms with minor differences, the overall process of these crowdsourcing platforms is the same as we discussed in Section 2.

3.1 Standalone Platforms

These are the platforms which do not depend on any other platforms for their functioning. A brief description of both general-purpose and specialized platforms are given below.

3.1.1 General-Purpose Platforms

Amazon mechanical turk. As mentioned earlier, *AMT* is the most popular crowdsourcing platform. It was launched in 2005 and is a part of *Amazon Web Services (AWS)*. It provides an on-demand, scalable and qualified workforce for a very small compensation. According to the AWS architecture, AMT can be accessed by three types of people, *Requesters*, *Workers* and *Developers*.

Requesters, those who post tasks, will have to use a software application to get AMT services like submitting questions, retrieving answers and others. They will be provided with a *Requester Console* to manage everything including the status of the 'questions they post. Requesters can also create qualification tests if the questions they post require some knowledge about a topic so that the job can be completed successfully. Requesters can access the console using this web address, *https://requester.mturk.com/*. With the help of a parameter (*QualificationRequirement*), which is a measure of the quality of a worker, the requester can filter out some workers and decide who can take up a task.

Workers are those who answer tasks posted by requesters. Tasks can be found at *http://mturk.amazon.com/*. Workers can take up any tasks they want based on the task type and the reward set by the task requester. Workers are rewarded only if the requester accepts the completed job [1].

Developers or even researchers can use different APIs provided by AWS to use different services of AMT to build their own application. Using AMT as a base, they can program a layer upon that to access specific services of the system.

Even though the total number of current workers on AMT is unavailable, according to *Wikipedia*, by January 2011, there was a total of 500,000 workers from 190 countries [2].

Crowd4U. Crowd4U is a non - profit platform for academics and general purpose crowdsourced task execution [99], [6]. Made available to public in November 2011, Crowd4U is an academic project powered by CyLog, a "Datalog-like language" to handle machine or human computations. Users do not have to register on the website to take part in different activities. But if registered, the data entered will be used for different purposes including academic or research work, without exposing their identity.

Operated by *FusionCOMP* projects, Crowd4U can be useful for different type of users [122]. For the public, it acts as a generic structure to solve problems. For academicians, Crowd4U can be used to conduct various research experiments. The system has implemented different strategies for recruitment, incentives and also for task assignment with a vision of improved efficiency. Crowd4U is used for different purposes such as damage identification of buildings, identifying the course of tornados and translation.

The developers of Crowd4U are mainly from the University of Tsukuba. A detailed list of ongoing and completed projects can be seen at [6].

gMission. gMission is a general crowdsourcing platform which uses the idea of spatial crowdsourcing, where the resources provided are bounded to a specific location and also the idea that a mobile phone can act as a moving sensor to collect data from different locations [48]. In *gMission*, a requester can ask different questions related to a specific location and workers who are at that location or moving towards that location can answer those tasks. *gMission* uses three modules, user interface module, function manager module and data manager module. While the user interface module takes care of requesters' and workers' interactions, function manager module takes care of location sensing, task recommendation and assignment, and quality control of the system. Data manager module takes care of all data storage and retrieval actions.

With different crowdsourcing techniques, *gMission* provides different real-time task solving services which can be useful in daily life for different purposes including the information about traffic of a location or crowd in a canteen. *gMission* was developed at the *Hong Kong University of Science and Technology* and currently most of the users are from the same university.

UpWork. UpWork is another crowdsourcing platform mainly used for complex tasks like interviewing or hiring [25]. It is different from *Amazon Mechanical Turk* as UpWork supports *macro tasks*, tasks which cannot be decomposed.

The platform has around ten million freelancers and four million clients who have registered in the system. This is considered as the world's largest freelancing market as around three million jobs worth $1 Billion USD is posted annually. A 10 percent cut from the total amount given to worker is taken by the platform. So in this platform, the requester, or the person offering a job need not pay anything, which makes this platform different from others. All the workers must provide a portfolio and attend the test to start working on the platform. The worker is considered eligible if he or she get at least 50 percent correct answer in the test [26].

The platform has won many awards including "TechCrunch50 DemoPit Winner (2010)".

Others. Other general-purpose, stand-alone crowdsourcing platforms include *clickworker* [4], *microtask* [15] and *Samasource* [19].

3.1.2 Specialized Platforms

InnoCentive. InnoCentive is a crowdsourcing platform founded in 2001 with a specific focus on commission research and development problems faced in different fields including engineering, science and business. All these challenging problems can be solved by anyone who signs up for the job and cash rewards are given to those who find the best solution.

Similar to other crowdsourcing platforms, InnoCentive has a requester-worker architecture and it has a user base of 355,000 users from 200 countries with most users from Russia, India, and China. Also, as the tasks are challenging, most of the users are well educated and the majority of them hold a Ph.D. [12].

Others. Metropolitalia, previously a research project, is a crowdsourcing platform used for linguistic [59]. *YourSpeech* was also a research project which was used for large-scale data collection [62]. *Gengo* and *Lingotek* are focused on translations [10], [13]. *OpenIDEO* is used to design solutions for various problems [16] and *Idea Bounty* is focused on idea generation [11]. *TopCoder* is a competition based crowdsourced computer programming [24].

3.2 MetaPlatforms

Meta platforms rely on the above mentioned standalone crowdsourcing platforms to complete tasks via a programed layer. A brief description of such platforms are discussed as follows.

3.2.1 General-Purpose Platforms

CrowdFlower. CrowdFlower was founded in 2007 by Lukas Biewald and Chris Van Pelt and the platform is mainly used by data scientists and enterprises for purposes of sentiment analysis, data collection, data categorization and transcription. It provides a trial account to get familiar with the system [7].

After uploading the data, a requester can design different tasks and publish them which can be monitored from the dashboard provided.

To contribute, one has to login through CrowdFlower's channel partners like *InboxDollars, Swagbucks*, or *Amazon Mechanical Turk*. After that, workers can see CrowdFlower tasks listed. They mainly use *Amazon Mechanical Turk* and *Samasource* for task execution.

Even though CrowdFlower provides different services, there are complaints registered for paying lesser compensation for the amount of work done [8].

3.2.2 Specialized Platforms

Quadrant of Euphoria: Quadrant of Euphoria is an academic research project which is a joint venture of Academia Sinica and National Taiwan University [45], [46]. It is a Quality of Experience (QoE) crowdsourcing platform with a very simple interface which facilitates quality assessments. Assessments are done for multimedia and network objects. The system contains diverse workers and provides easy-to-understand quality scores to the users for a very small compensation.

The tasks posted are treated as profiles in this system. Once the researcher or requester post a task in this system, a URL will be given to them for sharing on AMT. Upon the completion of tasks, the responses are collected for evaluation through a Flash application where one can vote the submitted answers.

3.2.3 Others

Crowd Guru is a general-purpose crowdsourcing platform which uses *clickworker* for task execution [5]. *Smartsheet* is an online project collaborating software and it uses Amazon Mechanical Turk and Livework for crowdsourcing services.

4 Incentives

Incentives are the main motivation to do a job. Without any incentive, it is unlikely that a person will be interested in doing anything specific. There are different types of incentives and different models to distribute them. A detailed description of both is given below.

4.1 Incentive Types

There are two types of incentives, monetary and non-monetary. Though both types have their own importance, different surveys have shown that monetary incentives are more preferred. In a survey conducted in February 2009, 91 percent of turkers preferred monetary reward over any other incentive. Out of everyone, only 42 percent of turkers were doing tasks just for fun. In another survey, it was reported that 25 percent of Indian turkers and 13 percent of US turkers were relying on different tasks in AMT as their primary source of income. These numbers are despite the fact that the requester has the entire right to reject completed work [134].

4.1.1 Monetary Incentives

Monetary incentives are the best and easiest way to motivate people [58], [65], [70], [79], [98], [114], [134]. A turker can choose any type of task from the available job pool and when the work is submitted the requester can either accept it or reject it. The worker is paid only if the requester accepts the work. Rejections have a strong impact on workers as more rejections make them vulnerable. This will lead to less acceptance rate of tasks.

4.1.2 Non-Monetary Incentives

It is not always the case that workers are looking for monetary compensation. We can find different people on the Internet who are ready to do different tasks for other intrinsic and some extrinsic incentives. These types of incentives are mostly used in knowledge sharing websites like *Wikipedia* [29] and *wikiHow* [28] and question answering websites like *Quora* [17] and *Stack Overflow* [20]. A brief description of different non-monetary incentive types are given below.

Natural incentives. Fun & entertainment: People prefer doing tasks which interest them more. So making a task more entertaining will help the requester to increase the user participation. Games with a Purpose (GWAP) is such an entertaining method to solve different complex tasks through games [123]. The ESP game was one such project where players were grouped into pairs and asked to label images. The task will be over when matching labels are found [125]. *Foldit* is another multi-player game used to

predict the structure of proteins [51]. In [44], authors have given a formal generic model for designing social games.

Personal development: Some people do tasks to achieve some satisfaction or to improve their skills. Crowdsourcing is a great platform where you can improve your skills and get good feedback. With the help of *Duolingo*, one can learn different languages along with contributing to the system by translating different articles [124]. With the help of [43], a set of stock photographs were developed along with learning photography. Other good examples are *Wikipedia* and *Quora* which help to acquire knowledge along with contributing to sites [17], [29], [100].

Competition: Competition is one of the main motivations to make people join for tasks [100]. In [121], the objective was to generate a 3D model of different buildings. To achieve that faster, authors ran a competition between students of two universities. As a result of competition, over 100,000 photos were submitted and most of them were relevant to 3D modeling.

Solidary incentives. Reputation & identity: As mentioned earlier, by doing many tasks, reputation increases. This point is very important as people do various things to get attention. It is also possible that people stop working when the attention, reputation and identity are lost. The importance of attention is discussed in [132].

Moral incentives. Humanity: During disasters, many people come forward offering help which is also a kind of motivation. [112], [118] are examples where people offered assistance on the spot.

Purposive incentives. Necessity: Sometimes people are requested to do tasks according to their employer's requirement or for the benefit of their organization [57]. von Ahn et al.[126] is a method where the task is done because of such requirements.

Material incentives. Points / Credits: There are several websites that use a point-based system for functioning. In [17] and [20], a turker is given more points when a question is answered or for participating in different activities. These points increase reputation among fellow workers. More points prove that a worker is better in that specific field. In [20], one can say whether a worker is good at programming or not based on the points. That reputation may help in job applications.

Others: There could be other incentives too. In [121], T-shirts were given for free to top students and in [66], snacks were given as incentives.

4.2 Incentive Models

It is imperative that the incentive design is well thought out as [58], [33] and [97] show how money increases participation but not necessarily the quality. Many workers try to finish up a job fast to earn money and maximize profit. These malicious users lead to the generation of ambiguous answers [116]. This clearly shows the importance of careful incentive design. Turkers should receive just the right amount of compensation [63]. Paying less could also lead to so-called task starvation problem, where tasks are not accepted by any workers [61]. Kazai [79] shows that the quality of answers is directly proportional to the payment. In fact, a payment increase from \$0.01 to \$0.10 for a task, showed quality improvement of 126 percent. Experiments also showed that the spam count was higher when the amount offered was smaller. In the experiment, 47 percent of the worker were unsatisfied with the \$0.01 pay condition while 72 percent were happy with \$0.10. Horton and Chilton [70] proposed a model to estimate the reservation wage for workers. Harris [65] showed that a better quality work was completed when the right amount of incentive was used. They also found out that the right amount of incentive helped people make more precise conclusions while attending a task.

4.2.1 Monetary Incentive Model

When it comes to monetary incentives, it is always difficult to keep both requesters and workers happy. If a task is underpriced, workers will not choose it and eventually that task will go unattended or might take a longer time to get it done. If it is overpriced, requesters will have to spend money on quality control as well and the total money spent will be higher. This is a disadvantage faced by requesters as this will give a competitive advantage to those having higher budget [140].

Another disadvantage is that workers with less background knowledge will generate low-quality answers which are of less benefit to requesters. That can be handled by accessing a worker's history and determining his/her reputation level so that high-quality solutions are guaranteed. But it is more difficult to design such a protocol integrating an incentive mechanism. Zhang and van der Schaar [140] and Xie et al. [133] have designed a model which integrate workers' history for task allocation and incentive design, which addresses this problem.

In most crowdsourcing platforms, a worker gets paid only if the requester accepts his/her work. Here, the whole control of payment, is in requester's hands, the same person who spends money. So even if workers do a good job, to get more profit, a requester may reject their job. In this case, workers will not get any payment and the requester can benefit from the job completed by the worker. Here, the requester is totally taking advantage of workers and this problem was addressed in [133] by proposing *Rating and Reward Dividing Method* for incentive design.

In some crowdsourcing platforms, payment is done before a worker starts a task. In this method, as the payment is given first, it is possible that the worker will make less effort to do a task which may result in low-quality answers. In [140], a punishing system was proposed to address this problem. Monetary incentive models can be divided into two, models which consider workers' history along with the reputation and models which do not.

Models considering workers' reputation. Reputation-based model: Zhang and van der Schaar [140] considers workers' reputation, with an assumption that all the payment will be done before taking up a task. The problem with giving money before the task execution is that the worker may not take the task seriously (called as *free-riding* problem) and the success rate might decrease. But the incentive protocol designed prevented this problem by assigning tasks, based on the workers' previous interactions with the crowdsourcing system. To improve the quality, authors set a scheme where more tasks are assigned to those with a higher reputation to increase their participation and improve overall

TABLE 1
Pros and Cons of Monetary Incentive Models

Category	Technique	Pros	Cons
Models not considering workers' reputation	Relevance-Based Model	• Easy to implement • Easy to keep track of payment	• Does not consider workers' rating or expertise level.
	Survival Analysis Method	• Given the time t, algorithm generates min wage for the task.	• Does not consider workers' expertise level. • High computation cost
Models considering workers' reputation	Reputation-Based Method	• Address free - riding problem • Integrates payment and reputation elements • Maximizes website's revenue	• Assuming tasks are homogenous and workers have equal expertise
	Rating and Reward Dividing Model	• Considers heterogeneous skill sets of workers • An administrator in the system makes sure that the requester is not taking advantage.	• Depending on the expertise of workers, there is a possibility that the requester might get only low quality answers

quality. The interaction between a worker and a requester is captured repeatedly like games. This interaction is modeled as an asymmetric gift-game and as mentioned before, payment is done before the task. In order to avoid *free-riding*, they consider users' action a as a binary set and $A = \{H, L\}$, where H represents "High level of effort" and L represents "Low level of effort". With the help of a utility matrix and two conditions, when $a = H$ and $a = L$, the involved cost is determined. They use *social norms*, to control workers' behavior. The system also has a punishing scheme for those who returns low-quality answers. This will control the above mentioned *free-riding* problem.

Rating and reward dividing model: In [133], they consider the fact that workers have heterogeneous skill sets. Some workers will be experts in certain fields and others might be rookies. Their model has a reward dividing system, a rating scheme and has considered other important crowdsourcing elements including task assignment process. As novice workers cannot contribute more than a limit, and as their solutions could be of lower quality compared to experts, this model has incorporated a reputation protocol to avoid low-quality answers.

The system uses probabilistic modeling to assign tasks. A worker's action can be defined as $\{H, L, D\}$, where H is when the worker agrees to put high effort, L is when the worker agrees to put low effort and D is when the worker rejects a question.

In the incentive model, a requester has to give a service charge or transaction fee T, along with the reward r, to the administrator. Once received, the administrator will assign m workers to do the job and will collect the answers, which will be given to the requester. The requester has to evaluate all these answers and should return a rating called "*feedback rating*" $\in \{-1, 1\}$, to the administrator. A rating -1 represents an unsatisfied answer and 1, for a satisfied answer. After getting all ratings, the administrator will equally divide reward r to all those who received rating 1. If no one has rating 1, the administrator will divide reward r, equally among all m workers. This is to make sure that the requester will not take advantage of workers. In other methods, the reward will be given back to the requester if a proper answer is not received.

Models not considering workers' reputation. Relevance-based model: In [65], authors used an incentive model, based on the relevance of a job. The task was to review resumes for three jobs in a company. Authors designed two HITs, first to evaluate the resumes on a scale of 1 to 5 to know its relevance to the job, based on its description. The second HIT was to mark each resume as either relevant or irrelevant. A golden standard was used for testing the second HIT.

Four variants of relevance based incentive models are used. In the first variant, they fixed a price for reviewing each resume. In this mode, no importance weight was assigned to the ratings given by a worker. In the second variant, called as positive incentive, a specific amount was set for reviewing a resume. The difference in this variant is that the requester informed workers that an expert had already reviewed it and the worker will be paid a bonus amount if his or her rating is same as the expert's. In the third variant, called negative incentive, workers were informed that a specific amount will be deducted from the base pay if the rating is different from the expert's. The fourth variant, called combined incentive, is the combination of the second and the third variants. The bonus is given if the rating is same as expert's and money will be deducted if the rating is different.

These four variants were used for both HITs. After experiments, in the first HIT, it was clearly shown that the positive and combine incentive variants gave a better performance than the other two. In the second HIT, it was shown that all the variants except the first one gave a similar performance.

Survival analysis method: In [61], they use *Survival analysis* method used in epidemiology and *Cox proportional hazard (CoxPH)* model to design a pricing policy. As CoxPH variables are assumed time - independent, authors used R's CoxPH implementation and derived a survival curve using the available data. They calculated the expected time completion by changing the reward from 1 cent to 1 dollar and implemented a recursive algorithm to calculate the price. The reward generated is the minimum wage required to complete the tasks in desired time.

Pros and cons of monetary incentive models: Table 1 on page 6 gives pros and cons of above mentioned monetary incentive models.

4.2.2 Non-Monetary Incentive Model

As different crowdsourcing systems use different actions to achieve points or credits (non-monetary incentive), in this section, we have to do a case-by-case analysis considering different crowdsourcing systems. Some of the famous question-and-answer systems which use points or credits are described below. Even though workers can get points and

accumulate it, in the end of the day, what they really achieve through these points is reputation.

Yahoo! Answers. In *Yahoo! Answers*, to increase the activity, they use Levels. A user begins from Level 1 with 100 points when they sign up. There are 7 levels, 1 to 7, and a user's level is based on the points they collect. The system deducts (awards) points for asking (answering) a question. The number of questions (answers) a user can ask (provide) per day depends on the user's level [30].

Stack overflow. Unlike *Yahoo! Answers*, *Stack Overflow* has a reputation measurement, which defines the community trust of a user. That measurement convinces other people that the worker knows what he or she is talking about. Therefore, if the reputation count of a worker is high, people will treat his or her answers more seriously and he or she will get more points when others vote up.

Reputation can go down as well. A user will lose points if his or her question (answer) is voted down. To ensure that the users do not vote on answers randomly, they will also lose one point when they vote down others' answers [21].

Quora: Even though Quora removed its point system, it is worth mentioning as it had high impact. Quora was using *Quora Credits*, "a virtual currency system on Quora that were earned via things like answer upvotes, and spent on things like Ask to Answer." In Quora, users can ask questions for free and they will be awarded 10 credits if someone follows their questions. Similar to asking questions, answering is also free and a user will be awarded 10 credits for each upvote he or she gets for answers from other users except the questioner. Another feature in Quora is "Ask to Answer(A2A)", in which a user can ask his or her questions to specific people or experts, by giving them some of his or her credits. In Quora, they also allow to transfer credits among users if they are out of credits [18].

5 Task Decomposition

The Tasks to be done could be sometimes complex which will be tough to solve. As it is important to finish tasks on time, the requester has to adopt a different method to solve complex tasks. It is common in the field of computation to decompose a task to reduce complexity. One main task is split into different easier subtasks which will be processed individually and the results from sub-tasks will be combined to get the final output. Depending on the complexity of the main task, different methods are adopted for task decomposition. In Sequential Implementation (Section 5.1), the tasks are divided into small tasks and are executed sequentially, taking the output of a task as input to another task. In Parallel Implementation (Section 5.2), a set of divided tasks are run in parallel and merged later to form a final output. In Divide and Conquer Implementation (Section 5.3), the main problem is recursively split into smaller problems which can be easily solved. Once the smaller problems are solved, they are merged back to generate the final output. Below, the above-mentioned implementation techniques are explained in detail.

5.1 Sequential Implementation

In this method, the sub-tasks will be executed serially. The system will choose the first task, execute it and then move on to the second task and so on until it executes the final task and generates output. It is important to note that in this method that the output of one phase is taken as the input for the next phase. Bernstein et al. [38], Hirth et al. [67], Zhu et al. [143], and Noronha et al. [102] are the examples for sequential execution of subtasks. In [38], for creating the tail answers it has to go through three stages, Identify, Filter and Extract. In [67], in control group method, the model splits the job to many subtasks first. All the subtasks from phase one are collected and assigned to the workers in the second phase for evaluation, and in the third phase, the final output is generated. In [143], the model first goes through training, then refinement and finally it classifies the quality of the results for submission. In [102], all the images of food are tagged first. In the second stage, each image is identified by describing, matching and voting. In the final stage, the quantity of each food item identified in the previous stages is estimated.

5.2 Parallel Implementation

Parallel implementation is a technique in which a set of independent jobs will be executed in parallel to save execution time. SCRIBE is a peer to peer system which facilitates instantaneous captioning for deaf people in real time [89]. When the worker starts recording, different captionists will try to capture different parts of recording and those partial captions caught through Flash Media Server will be merged together and sent back to the user. Captionists are recruited from AMT [1] or quickTurk [39]. In this model, workers are used in parallel by dividing the main task into sub-tasks and assigning them to different workers. In [121], users are encouraged to take pictures of different locations in parallel and merge all those photos in the final step to develop a 3D model. In [126], they made a model to transcribe old books with the help of CAPTCHAs done by various users around the world. In [38], [67], [143], and [102] even though the main tasks are executed sequentially some of the sub-tasks were done in parallel. For example, in [67], the sub-tasks of phase one are executed parallel to reduce the time cost.

5.3 Divide and Conquer Implementation

The above two methods explain simple tasks which can be solved easily in sequential or parallel implementation. Divide and Conquer method is used when there is a complex task which can be executed efficiently by decomposing the main task into smaller tasks until it cannot be split further. Every small task will be executed and the results from decomposed small tasks are merged recursively together to get the final output. The divide and conquer method makes the task execution not only simple but also efficient and accurate. In *Turkomatic* [31], [87], and [101], the divide and conquer method is used in various tasks such as data generation and labelling.

5.4 Macro Tasks

Some tasks such as hiring a web developer or a designer are complex and non-decomposable. These tasks are referred as Macro Tasks and [64] has done a real-life case study on such tasks. UpWork is a good crowdsourcing platform which deals with *macro tasks* [25].

TABLE 2
Pros and Cons of Task Decomposition Techniques

Technique	Pros	Cons
Sequential Implementation	• Easy implementation • Better visibility of procedure after each step • Tangible output after each process	• More execution time • Dependencies may delay the output. • Problems in one step will affect the subsequent steps and eventually the entire process.
Parallel Implementation	• Less execution time	• Complex implementation
Divide and Conquer Implementation	• Less execution time • Efficient algorithm formation	• Complex implementation • Slow recursion

5.5 Pros and Cons of Task Decomposition Techniques

Pros and cons of different techniques for task decomposition are given in Table 2 on page 8.

6 Task Assignment

Choosing workers for a specific task is always difficult. For crowdsourcing tasks, it is important to choose workers carefully as the quality of output is directly dependent on those who work on it. As mentioned earlier, workers recruited for crowdsourcing tasks come from different backgrounds possessing diverse sets of skills with different objectives. So the tasks can be assigned to different people based on different factors. This diversity of workers has a direct impact on the success rate of tasks [109].

When recruiting workers, two important criteria need to be considered, which are job requirements and human attributes. For every job, there will be some requirements, like a specific skill set that workers should have. Kazai [79] shows the importance of choosing workers based on the knowledge required to carry out some specific tasks. Kazai [79] demonstrates that the resulting quality drops when workers are chosen randomly. In [39], workers need to have some prior knowledge about the camera parameters to carry out a task. In [73] workers are assigned based on the difficulty and the total time required to complete the task.

Another important factor is human attributes. Among workers, there could be people who are spammers, cheaters and proper turkers[80]. While proper workers provide good results, others just play around with different objective in mind. Clearly, cheaters and spammers will reduce the success rate of tasks.

However, in crowdsourcing, as it is online and no interview can be done to recruit workers, it is quite challenging to figure out whether workers are cheaters or not. The current solutions rely on examining the workers' completed tasks. Specifically, when a task is completed, a worker is paid only if the requester approves the result [114]. Thus, if the worker did not get paid for a task, it means that the submission was wrong or at least did not satisfy the requester's need. By counting all the unsuccessful tasks, the crowdsourcing platform could rate the worker. Based on these ratings and task requirements, the crowdsourcing platform can pick up the workers for the specific tasks. Though this solution is reasonable, however, it is quite challenging to implement it in real practice. First, to rate a worker, we are required to have enough data for analysis. For a new comer, even though he or she is a good worker, as their rating is zero (since there is no history about them), he or she may not be recruited for any tasks. However, considering the fact that there are a lot of people on the Internet, many of them will be new comers when they work for any crowdsourcing tasks. Another issue is that this solution can be easily tricked by playing the role of requester and worker by the same person [74]. All one has to do is to be a requester and create a simple task in the platform. After creating, logout from the platform and login as a worker and take up the task that he or she just created. After that login again as the requester and approve all the tasks. By doing these steps repeatedly, his or her rating will be increased.

The main problem faced in task assignment is the unavailability of data. Some crowdsourcing platforms will not publish the information about their workers, but the system is still required to be able to pick up workers to maximize the profit. Based on the availability of workers' data, task assignment can be mainly classified into offline and online task assignment methods. Task assignment methods in which the workers' data are available in advance are considered as offline, while those methods where data are not available in advance are treated as online. Assadi et al. [34], Karger et al. [78] proposed solutions to deal with offline task assignment. Ho and Vaughan [69], Ho et al. [68], Yuen et al. [138], Boim et al. [40], and Zheng et al. [141] targeted on online task assignment.

6.1 Offline Task Assignment

In offline task assignment, a requester will have all necessary information about workers in advance. The requester has access to the skill set and also the bid of every worker. According to the skill set, bids of workers and other factors including the number of questions asked and the total budget specified by the requester, the system will select the workers for the tasks to maximize the profit. Several typical offline assignment solutions are presented as follows.

Offline approximation algorithm: In [34], the workers bid for the tasks and the amount is known before. All the tasks are denoted as j and workers as i. The total number of tasks available is denoted as m. The model addresses two sequences of workers arriving in two different ways, an adversarial setting and a random permutation model, respectively. The requester has a total budget of B and each worker can pick a numeric bid of $b_{i,j}$. When a worker takes up a job j, the requester has to pay at least $b_{i,j}$. The method starts with modeling tasks and workers in a bipartite graph with tasks and workers on opposite sides. If the worker is ready to accept any task, an edge will be drawn connecting those two nodes with a cost of $b_{i,j}$. There is a source node connected to all the workers and a target node connected to

all task nodes with the capacity of all edges assigned to 1. All the connections between these two opposite nodes show the task assignment, and the minimum cost of a flow denotes the minimum budget required for the assignment. It is proved that the minimum cost flow will give the optimal solution. To compute the final result, the system uses a sub routine named *Fixed Threshold Policy*. This function is provided with a threshold value p. Once the threshold value is available, the algorithm goes through the total set of workers and a task is assigned to an idle worker, if the bid value is lesser than the threshold value p. The remaining budget will be updated after assigning each task and finally, the algorithm stops arbitrarily when an unassigned task, for which a worker bid p or less, is found.

Iterative learning: In *Iterative Learning* [78], instead of hiring a lot of workers and assigning the job randomly, jobs are assigned in a specific pattern with the help of a set of skilled workers and the estimated accuracy can be obtained. *Iterative Learning* model also uses a bipartite graph G denoted $\left(\{t_i\}_{i\in[m]}\cup\{w_j\}_{j\in[n]}, E\right)$. Here t represent tasks and w represent workers. m and n denote task and worker nodes respectively. Edges between the tasks and workers are denoted by E . A requester specifies left degree l denoting the number of workers needed for a task and right degree r denoting the number of tasks assigned to a worker. Due to the requirement of consistent, $ml = nr$, where m and n are tasks and workers respectively. For task allocation, as a random graph generation is sufficient to achieve an optimal performance, the model uses random graph generation scheme. This scheme is called a *configuration model* [41]. Here the optimal performance refers to the situation where a minimal budget is used for task assignment.

After the task allocation, an inference algorithm is applied on the graph to improve the results. The inference algorithm uses real valued task messages denoted as $\{x_{i->j}\}_{(i,j)} \in E$ and worker messages denoted as $\{y_{j->i}\}_{(i,j)} \in E$. These messages are regularly updated in every iteration. Messages of a worker, $y_{j->i}$, represents the reliable factor of that worker j. The final estimate is given as $\hat{S}_i = sign(\sum_{j\in\ \partial i} A_{ij} y_{i->j})$ where ∂_i are the neighbors of task t_i and A denotes the answers. The average quality of the crowd is given as $q = E[(2p_j - 1)^2]$ in which p_j denotes the probability that a worker completes job j. The optimality of the algorithm is proved.

Collaborative method: This algorithm [35] focuses on collaborative crowdsourcing where a different set of skills possessed by a set of workers is used to solve one complex problem. As it is collaborative, apart from considering default human constraints like wage and skills, the system has also considered team-based factors such as *worker-worker affinity*, which represents the comfort level of a worker with other workers working on the same task, and *upper critical mass*, which denotes the maximum allowed size of a team after which the collaborative effect reduces.

The expertise of an individual in a domain d_i is $u_{d_i} \in [0, 1]$ and $w_u \in [0, 1]$ represents the minimum compensation needed for a worker to do one task. Here d is the domain,u is the worker and w is the wage. The affinity between two workers is represented as $aff(u_i, u_j)$. But to fit in $[0, 1]$, affinity value is normalized and uses the distance between workers, $dist(u_i, u_j) = -= aff(u_i, u_j)$, to represent the same. Lesser distance indicates a better collaboration. Another factor is *intra-group affinity* which refers to the working effectiveness between workers in the same group and is measured using *Diameter Distance*, which is the largest distance, meaning lesser affinity, between any worker pair:

$$DiaDist(\mathcal{G}) = \max_{u_i,u_j\in\mathcal{G}} dist(u_i, u_j). \quad (1)$$

The system uses one more factor, *inter-group affinity*, to deal with the increase in group size. When the size goes beyond *upper critical mass*, it should be decomposed into smaller groups. The smaller groups will be assigned a part of the main problem and the contribution from each sub groups will be later aggregated to solve the main problem. Hence, the effectiveness of collaboration of a group is the sum of the distance between the workers in its subgroups. If G_1 and G_2 are two groups split from main group G, their combined effectiveness is calculated using:

$$SumInterDist(G_1, G_2) = \sum_{u_i\in G_1, u_j\in G_2} dist(u_i, u_j). \quad (2)$$

The algorithm uses a two-step method, *Grp*, where a group is formed satisfying the required skill and wage, but without considering upper critical mass, and *Splt*, where the *upper critical mass* constraint is enforced. In the first step a sorted list, L is made based on the worker distance in ascending order and does a binary search. For each user u_1, a star graph is made with other workers as nodes and edges as the distance. The edges should be less than α, which is the maximum allowed distance between workers. Then a function is used to pick users who satisfy the wage and skills mentioned.

This method can be used for cooperative projects like product designing, editing or translation.

6.2 Online Task Assignment

In online task assignment, the requester will have no prior information about the worker. The requester can specify a total number of questions, final budget and the number of times he or she wants each task to be answered. Tasks will be assigned online when turkers arrive [69]. The main challenge of this method is that as turkers are unknown, the requester does not have access to the information regarding the skill sets of each turker. But these skill sets will be later explored by observing the answering behavior of turkers. That could lead to a problem of data uncertainty where data available is not enough for processing. Also, the system developed should be able to handle heterogeneous tasks. The ultimate goal is again to assign turkers efficiently and achieve high productivity.

AskIt!: Boim et al. [40] determines which question to be directed to whom in order to reduce the uncertainty of the data collected. The system works in real time with U users and Q questions. The matrix M is formed by $|U|$ x $|Q|$ which provides the answer given by user u, for the question q and the entry will be null if the user has not answered that question. A_q denotes the set of possible answers for the question and ϕ denotes an unknown answer, means the answer to

that question is not yet given by user u. The uncertainty in the current answers for some questions q is quantified with the help of entropy equation. This method uses collaborative filtering to predict the missing answers and also to predict which questions will be answered. This system reduced the sparsity and also gave high-quality prediction with the help of collaborative filtering method. The system was also able to reduce the uncertainty in data distribution and the number of questions asked.

TaskRec: Yuen et al. [138] is an unified probabilistic matrix factorization based task recommendation system for dynamic situations in which the user rating is explored through the interaction with workers. One of the main problems in recommendation systems is the cold state problem, a problem faced by different items which are never rated by any user. Latent factorization is an alternative way to address this problem and it is best done by matrix factorization [86]. As the matrix factorization does not scale well, they use the PMF model [111] as it can be scaled with observations.

TaskRec is divided into three parts. First, the system merges workers' preferences of tasks with workers' category preferences through shared worker latent feature space. Then it will connect workers' task preferences with task's category grouping preferences through shared task latent feature space. Third, task's category grouping information is merged with workers' category preference information through shared category latent feature space. With the help of the above-mentioned methods, they can discover the interests of workers. To address the cold state problem, where ratings are missing, they use a matrix factorization method to predict those ratings based on user behavior. Once the ratings are known, it can be ranked and directed to the workers.

Dual task assigner: Ho and Vaughan [69] solves the online task assignment problem faced by an individual requester who has a set of heterogeneous tasks to be done. The system assigns workers of various skill sets to the set of tasks introduced. In this method, the skill set is not known in advance. But is discovered by going through the set of work done by an individual through an iteration process. In this model, there are two steps, exploration and exploitation, through which the skill set is discovered. The whole idea is based on primal-dual formulation.

The model assumes a total of n tasks submitted by the requester. The requester also has to specify the total number of times he or she wants each task to be executed with a budget of b_i, where i represents each task. The total size of workers is represented in k and j represents each worker. The skill set of each worker is defined as $U_{i,j}$. This value, utility value, is additive and will be updated b_i times. The requester should give a job to each worker. The worker will finish up the job and return the results to the requester. Upon receiving results, new utility value will be added to the old utility value. The evaluation is done using competitive ratio and is analyzed using random permutation model [56] along with *Learn Weights Algorithm* with minor modification. As mentioned before, LW phase is divided into two steps, exploration phase and exploitation phase. In exploration phase, along with estimating $U_{i,j}$, the optimal task baseline weight is calculated for i tasks and is passed on to exploitation phase where the dual objective is optimized.

Adaptive task assignment: Ho et al. [68] is closely related to [69] but there are some key differences. One is that unlike [69] this model does not assume that the requester will evaluate the quality right away. This model also needs repetitive labeling in order to estimate the worker quality.

Just like Dual Task Assigner model, there is a total of n tasks and the label of each task is denoted as l_i which has a value of either 1 or -1, which is not known to the requester. The worker has a capacity of M_j where j is the worker and that is the only thing known to the requester. Like DTA, the skill level is denoted as $p_{i,j}$. When a work i is assigned to worker j with a probability of $p_{i,j}$, ATA can return a label $l_{i,j}$, such that $l_{i,j} = l_i$ and with a probability of $1 - p_{i,j}$, ATA can return a label such that $l_{i,j} = -l_i$. In every repetitive step, the algorithm decides whether to assign a work or to move on. As the workers are strangers, they cannot be assigned to any work once the algorithm moves on. The results are depended on the quantity, $q_{i,j} = (2p_{i,j} - 1)^2$. The label is considered as informative if the value is closer to 1.

Online heterogeneous algorithm: This algorithm [34] is motivated from [142]. The system uses a potential function, $\Phi : [0, 1] - > [1, R]$ which is based on the fraction of budget spent so far. Here Φ is the threshold value and R is the upper bound of workers' bid. A task is assigned if the bid of the worker is less than a threshold for any upcoming unassigned task. In other words, the algorithm becomes more selective about workers when the budget decreases. A greedy task assignment is done when a suitable worker is found. This model uses the random permutation model and the performance is measured with competitive ratio $O(R \epsilon\ ln(R))$ when $R \leq \epsilon B$.

QASCA: QASCA stands for A Quality-Aware Task Assignment System for Crowdsourcing Applications [141]. It is another task assignment method which is better than AskIt. It makes use of the fact that other models do not consider the application type for task assignment. They showed that consideration of evaluation metrics can significantly improve the results. This is an application driven method which makes use of accuracy and F-Score metrics. With the help of two distribution matrices (Current Distribution Matrix & Estimated Distribution Matrix), where the probability distribution of answers is captured, they addressed the ground truth problem which helped to maximize the quality of result of each question and the cost of execution is reduced with the help of two linear algorithms which is used for online assignments.

The requester has to set n questions having l labels along with the number of questions asked in each hit, a total budget and the evaluation metrics. There are two actions triggered from the workers' side named HIT completion, where the worker finishes the job and HIT request, where he or she requests a HIT. When the worker requests, first they calculate the Estimated Distribution Matrix, Q^w of each worker w, with the help of Current Distribution Matrix, Q^c. Those questions which the worker has attempted will be excluded from the new set of questions he or she will be possibly assigned. With the set of remaining questions, the system finds the combination of questions and calculates Q^{x1}, Q^{x2}, $Q^{x3} \ldots Q^{xn}$ where n is the total number of possible combinations. The system analyzes all the above-mentioned

TABLE 3
Pros and Cons of Task Assignment Models

Category	Technique	Pros	Cons
Offline Task Assignment	Offline Approximation Algorithm	• Provides the optimal solution. • Simple implementation	• Assumption regarding bid to budget ratio may not be feasible all the time.
	Iterative Learning	• Minimizes resource usage • Efficient execution	• High computation cost
	Collaborative method	• Works very well in collaborative environments. • With group decomposition, system makes sure that the integrity of the group is maintained.	• Scarcity of the skills in subgroups could lead to problems.
Online Task Assignment	AskIt!	• Maximizes uncertainty reduction extent	• The algorithm is not optimal and has a possibility that some questions will not be answered by any worker.
	TaskRec	• Addresses cold start problem • Scalable to large datasets	• The performance degrades dramatically when training data are not enough.
	Dual Task Assigner	• Focuses on heterogeneous task • Works with unknown skill set • Works with large number of incoming workers	• Less frequency of workers leads to less information, hence the method could be inefficient.
	Adaptive Task Assignment	• Works efficiently for diverse workers	• Poor efficiency with uniform workers
	Online Heterogeneous Algorithm	• Addresses heterogeneous task assignment problem	• Picky in case of worker hiring, which may hurt works' motivation
	QASCA	• Workers quality is accurately determined	• High computation cost

values and based on that estimation metric picks the maximum value generating combination and those questions will be assigned to the worker. Once the worker returns answers, the system updates the answer sets and workers' quality, which is stored in Confusion Matrix, is used to update Current Distribution Matrix, Q^c.

As mentioned before, this system uses estimation metrics like accuracy and F score. As accuracy is in need of ground truth, this system uses expected accuracy to measure the quality of the results. Expected accuracy is defined as:

$$Accuracy^*(Q, R) = \frac{\sum_{i=1}^{n} Q_{i,r_i}}{n}. \quad (3)$$

Here R denotes the result vector. But the values of F-Score is difficult to calculate as the numerator and denominator are random variables. F-Score is calculated with the equation:

$$F - Score = \frac{1}{\alpha.\frac{1}{Precision} + (1-\alpha).\frac{1}{Recall}}. \quad (4)$$

In above equation, *Precision* is the fraction of correct results returned and *Recall* is the fraction of answers returned with the same ground truth. In the above equation, α acts as an emphasis control variable, which decides the emphasis is given to recall or precision. The challenge faced is that the denominator contains random variables. Hence, they could only calculate the estimate F-Score.

6.3 Pros and Cons of Task Assignment Models

Table 3 on page 11 gives pros and cons of above mentioned task assigning techniques.

7 Quality Control

The main objective of crowdsourcing system is to maximize the productivity with the minimal cost. Turkers are humans and they will have different characters. Like mentioned before, there could be genuine turkers, who answer questions properly and also there could be spammers, who have different types of intentions from earning money to sabotaging the system. Spammers will reduce the productivity of the whole system which ends up in wasting more resources. As the turkers are online and unknown, it is impossible for the requester to identify the characters of each turker. So it is necessary to implement different methods to guarantee the quality of system output.

Quality control is used to achieve the objective of crowdsourcing platforms and can be done in two phases, pre-task execution and post-task execution. All phases are discussed below in detail.

7.1 Pre-Task Execution

We call the phase before the issue of tasks as pre-task execution. In this phase, many mechanisms can be taken to avoid the bad quality outcome of crowdsourcing tasks, including incentive design, task planning, prior knowledge training and worker skill estimation. We discuss each factor in details as follows.

7.1.1 Incentive Design

Different types of incentives are the major motivation of crowd workers. Unlike offline work, factors such as responsibility, management etc. will not be counted in the list of motivating factors for online tasks. Just like any other jobs, people will leave the task and go for other tasks if they are not given the right incentive. So it is

really important to design incentives properly before the task is published. Incentives could be monetary or non-monetary. A detailed description about incentives is discussed in Section 4.

7.1.2 Task Planning

It is important to plan the questions or tasks properly. Like mentioned before, reputation or reward of a worker depends on the acceptance rate of a requester [114]. If the requester rejects an answer, it will make an impact on the workers' profile which prevents him or her from choosing some tasks. Sometimes, it is possible that the workers are provided with wrong instructions or do not make themselves clear. This will lead to the generation of a junk of wrong answers and wasted jobs, which will be rejected by the requester as he or she did not get what they expected. It is possible that the turker was genuine but the wrong planning of questions led to the wrong answers which later got rejected. Such kind of buggy task or wrong information of tasks should be avoided.

It is also important to give enough time to workers for them to finish a task. Sometimes it is possible that the task gets expired before it is completed, which may be caused by two factors. 1). The turker left the job in between and 2). Sufficient amount of time was not given to finish a specific task. If some tasks need thinking for solving, giving enough time will produce more accurate results compared to getting some random answers in a short time.

Furthermore, it is important to select the right number of workers [92] before issuing a task. More money has to be spent if we give the same task to many workers. It will also generate a lot of data which increases the probability of useless answers. At the same time, allotting tasks only to a few will end up getting less data which leads to inaccuracy due to insufficient data. So a proper approach should be followed in choosing the right number of workers.

7.1.3 Prior Knowledge Training

As mentioned in Task Assignment, prior knowledge is also an important factor in controlling the quality of answers [92]. Given some complex tasks, turkers might need a good knowledge in some specific fields or need to possess a different set of skills. Lack of these prior knowledge will lead to wrong answers. To address the issue of lacking prior knowledge, current solutions try to provide enough knowledge to workers in order to carry out the task or ask workers to pass certain qualification tests before taking the job. Le et al. [90] solved this problem by proposing *Gold Standards*.

Gold standards: Le et al. [90] considered the fact that even though the turkers are genuine, generation of invalid results may have caused by the misunderstanding of the problem. To address this issue, before a task is executed, we can set a training period. In the training period, a worker will be asked to answer a set of qualification tests. It is necessary to pass the tests to get the payment. Once they attempt all these questions, they are subjected to screening and a feedback will be given. In this feedback, they can discuss why they choose their answer and the requester can explain why the worker should have chosen the right answer over the others. With this method, the worker can be trained to have a better idea of answering the questions. The spammer can be identified and can be banned from doing the task. Also, a regular feedback about the performance is a better way to improve the quality of results [90].

7.1.4 Workers' Skill Estimation

In [107], authors tried to estimate the workers' skill, based on the team work they have participated in the past. Here, a team was often formed by workers with different expertise to solve a problem. They used *Sum* and *Max*, two famous techniques for skill aggregation and the workers' skills are represented either as a probability distribution or a deterministic value. Specifically, as input, authors take a set of workers in a team who has worked on at least one task together. The expertise level for the task will be known before the task is executed, it is evaluated based on the quality of the previously completed team work. As the whole point is to estimate workers' skill, they use *Sum* and *Max* to aggregate skills of individuals to calculate team's expertise or skill level. However, they assumed that the workers' expertise levels are known in advance, which in fact is not true, especially when the workers arrive online.

7.2 Post-Task Execution

This section includes different models and frameworks used to control the quality after a task is executed. There are different methods to control quality, which include identifying the cheater and worker selection. We discuss these two different models as follows.

7.2.1 Cheater Detection

The success rate of a task will be improved if the cheaters can be eliminated. However, in the current crowdsourcing systems, it is hard to identify cheaters since the requester cannot interview the turkers before they post the tasks. Inserting checks is a solution proposed in [66] to identify cheaters from a pool of turkers. In order to stop spammers, the authors introduced a spam detection system called *Umati*. Umati includes a golden standard task (a task with knowing correct response). If a turker cannot answer the golden task correctly, he or she will be logged out and blocked for future tasks. It is clear that it is hard to find golden tasks for all the applications.

7.2.2 Worker Selection

Another way to control the quality of responses is to select proper turkers to assign tasks based on turkeys' ratings. *CDAS* proposed in [92] selects turkers based on their ratings before task assignment.

CDAS: CDAS stands for Crowdsourcing Data Analytics System which uses a quality sensitive answering model [92] as its core. The main idea of CDAS is to achieve a better quality result with less processing cost in a data analytics job. CDAS converts a job into two parts, one will be completed by humans and the other one will be computed by computers.

The architecture of CDAS mainly contains three components. First one is a job manager whose duty is to separate

the given jobs into human and computer jobs. It converts the given jobs into processing plan by describing the association of other two components, crowdsourcing engine and program executer. The second component, crowdsourcing engine, is responsible for separating the human job process into two sections. In the first section, CDAS generates a template to convert the main job into different tasks and estimates the total number of workers needed based on workers' performance. In the second section, all the answers are collected, combined and refined to remove the ambiguity. The last component, program executer, summarizes the results of crowdsourcing engine.

The core of crowdsourcing engine is split into prediction model and verification model. The performance of crowdsource engine is dependent on these two models. The prediction model is used to identify the number of workers needed for a particular task and the workers are selected based on their previous performances. Authors proposed a modified version of the voting system to predict the number of workers needed based on their accuracy. In CDAS, an answer is accepted if $\frac{n}{2}$ workers provide the same answer. However, this voting method will encounter a problem when there are three answers and none of the answers got more than 50 percent. So the probability that at least $\frac{n}{2}$ workers return the same answer is:

$$P_{\frac{n}{2}} = \sum_{\mathbb{U} \subseteq u, |\mathbb{U}| \geq \left\lceil \frac{n}{2|} \right\rceil} \left(\prod_{u_i \in \mathbb{U}} \prod_{u_i \notin \mathbb{U}} (1 - a_j)) \right), \tag{5}$$

where $A = \{a_1, a_2, a_3 \cdots a_n\}$ represent the accuracy of all n workers and $\mathbb{U}$ represents a subset of worker set ($\mathcal{U}$) with total size not less than $\lceil \frac{n}{2} \rceil$. The expectation probability can be calculated as

$$E[P_{\frac{n}{2}}] = \sum_{k=\lceil \frac{n}{2} \rceil} \binom{n}{k} \mu^k (1-\mu)^{n-k}, \tag{6}$$

where the mean value of workers' accuracy is represented as μ. After finding these two values, a binary search with modified parameters is used for optimization.

The second model is a verification model, which is used to evaluate the quality of work. Instead of voting based verification, CDAS employs a probability-based verification method by using Bayesian analysis. The probability of a specific answer $\overline{r} \in R$ being the correct answer is:

$$P(\overline{r} \,|\, \Omega) = \frac{P(\Omega \,|\, \overline{r}) P(\overline{r})}{P(\Omega)} \tag{7}$$

$$= \frac{P(\Omega \,|\, \overline{r}) P(\overline{r})}{\sum_{r_i \in R} P(\Omega \,|\, r_i) P(r_i)}. \tag{8}$$

In the above equations, R is the domain of answers and Ω stands for the observation of the answer's distribution. They also calculate the confidence of the worker and the answer, the results show that the workers with high accuracy often have more confidence. The confidence of an answer being correct is calculated and updated regularly on time. When we have enough data to support the current answer, the process can be terminated.

Robust personal classifier: Due to spammers, noisy labels are often generated which are not useful [117]. One solution to avoid noisy labels is to repeatedly label to get an estimate on correct labels [113]. Using attributes of different instances has attracted various researchers and many models have come up by exploring attribute features. Kajino and Kashima [76] proposed *Personal Classifier* model which worked well with the environment they had set-up. However, it is not realistic as the assumption that all the workers give equal contribution is inaccurate as the contribution of each worker depends on their individual abilities. *Robust Personal Classifier* [93] did not take any such assumption and gave a very good classification between genuine workers and cheaters.

In *Robust Personal Classifier* each worker will be assigned a score which represents his expertness. Unlike Personal Classifier model, this model uses "a surrogate optimization algorithm" to avoid parameters for learning to rate and also to guarantee convergence. The system has a learning phase where the expertness of each user is discovered. The system will give high weights to genuine users and low weights to cheaters. After the learning process, it is clear that workers with different expertise scores contribute unevenly. So, using these expertise scores, spammers can be identified and eliminated.

7.2.3 Turkers with Specific Objectives

When the whole work is given to a set of people, there is a higher chance to get faulty answers compared to giving a specific task to a set of people. Thus, by forming a different set of workers for decomposed tasks in a system, the quality can be increased dramatically. For example, for a content generating task, instead of making one turker write and proof read the content, assign the writing task to a set of turkers and proof reading to another set of turkers will improve the quality. Bernstein et al. [37] and Hirth et al. [67] use this method to increase the quality of answers.

Find – Fix – Verify: In [37], the authors mainly addressed open-ended tasks and split one task into three parts. The first part is *Find*, where the sections required changes are found in the work submitted and this task will be given to different workers. All such sections found are aggregated to find the issues reported the most and those specific sections will be forwarded to the next step.

The second part is *Fix* phase, where different workers will be recruited to fix the problematic patches. The same patch will be given to more than one worker to reduce possible bad work.

After the *Fix* phase, in the third part, *Verify* phase, another set of workers will be hired to vote on the fixed patches. The reason for selecting a new set of people for verifying is to ensure that the worker is not verifying his/her own fix. After this phase, most of the noisy labels will be refined. The drawback of this method is that if a worker is late to choose a task, all others will have to wait for him or her to go to *Fix* phase. So keeping a timeout parameter will be better to address this problem.

Control group approach: In [67], the authors proposed using a control group to improve answer quality. Specifically, they categorized all crowdsourcing tasks into three types. Routine tasks, which are relatively easy to cheat,

complex tasks and creative tasks, which are relatively hard to cheat. They assumed that there are N workers in total and each worker is only assigned one task. Furthermore, they assumed that no worker will intentionally cheat the system and all the wrong results are the outcome of carelessness since workers want to finish the tasks as fast as possible. The probability of returning wrong task result is $P_w = P_c.P_{w|c}$ in which P_c is the probability that a random worker is a cheater and $P_{w|c}$ is the probability that a worker is not a cheater.

To avoid the wrong task results, there is a group of workers to control the answers, called *control group*. Specifically, when a worker submits results, instead of sending the results directly to the requester, they will be sent to another set of workers (control group), whose duty is to evaluate the submitted results according to the given criteria. As it is possible that the control group itself can have cheaters, once the control group finishes the rating of submitted results, those ratings are sent to the crowd again to select the correct answers through majority decision. The worker will be paid if the answer he or she submitted is considered valid.

As Control Group approach uses majority decision method, group size is important. Out of N total workers, the control group method will choose m random workers and the probability that a worker is a cheater is p_c. There is another probability showing that the worker submits an invalid result denoted as $p_{w|c}$. The invalid results follow a binomial distribution and the number of invalid tasks must be greater than $\frac{m}{2}$ due to the majority decision model. The probability of a successful majority decision p_m is given by:

$$p_m = p\left(X < \frac{m}{2}\right) = P\left(X \leq \left\lfloor \frac{m-1}{2} \right\rfloor\right) \tag{9}$$

$$= \sum_{k=0} \binom{m}{k} p_W^k (1-p_w)^{m-k}. \tag{10}$$

Here X represents the total number of errors. The total cost of CG method is a combination of four values.

$$C_{CG} = c_{wd}.p_{wd} + c_{w\bar{d}}.p_{w\bar{d}} + c_{rd}.p_{rd} + c_{r\bar{d}}.p_{r\bar{d}}. \tag{11}$$

Here, p_{wd} denotes the probability and c_{wd} represents the expected cost of the situation when a worker submits a wrong answer and the control group marks that answer as invalid. $p_{w\bar{d}}$ represents the probability and $c_{w\bar{d}}$ denotes the expected cost of the situation when a wrong answer submitted by a worker is marked valid by the control group. The probability of the situation when the correct answer submitted by a worker is marked valid by the control group is represented as p_{rd} and the cost of the same situation is denoted as c_{rd}. $p_{r\bar{d}}$ represents the probability and $c_{r\bar{d}}$ represents the expected cost of the situation when a worker submit correct answer and control group mark it as invalid.

7.2.4 Iterative Aggregation

Normally in the iterative aggregation process, there will be two steps. First by regular updating, these approaches will try to combine the answers of each task, based on the expertise of the worker [106]. Second, the expertise of the worker will be adjusted based on the answers he or she provides. Raykar et al. [108], Whitehill et al. [131], Karger et al. [78], Yu and Nickerson [136] proposed different Iterative aggregation models. In these models, there is a set of computational rounds in which the probability of correct answers is incremented in every round. Unlike non-iterative methods, these models do not need any filtering questions.

Supervised Learning from Multiple Experts (SLME): The SLME method proposed in [108] works like Expectation Maximization model, but instead of using confusion matrix, SLME uses measures from statistics, sensitivity, i.e., the correctly assigned positive answer ratio and specificity, i.e., correctly assigned negative answer ratio. These measures are only defined for binary labeling. Hence, it won't be compatible with multiple labeling.

Generative model of Labels, Abilities and Difficulties (GLAD): GLAD proposed in [131] also works like Expectation Maximization model and takes into consideration workers' expertise and toughness of question like ELICE [83]. The system addresses two special cases, one is that the probability of getting the toughest question correct is little and the other one is that a worker with high expertise has a higher chance of answering the question right.

Iterative learning: Like other systems, iterative learning [78] also addresses the expertise of a worker along with the toughness of a question. Unlike others methods that there is only one overall reliability value which is the expertise value, the reliability of each answer in iterative learning is different. Also, the toughness of each question will be different for each worker. So the final expertise of a worker is the sum of the answer reliability weighted with the question difficulty. The model is based on standard belief propagation [78].

Free choice model: In single choice model [52], only the best output of the previous iteration is considered as the input for next round. Even though this model is easy to implement, it does not consider most of the previous answers which lead to inaccurate results or will require more iterations. But free choice model uses all previous outputs as inputs to the next iteration [136]. So unlike the single choice model, this model considers all the previous answers for each iteration which might increase the cost associated. A genetic algorithm is an example for this. First there will be generation one which evaluates and then later combines to produce offsprings. Then, these offsprings will be considered as generation two. Generation two will also need to go through the evaluation and combine steps like generation one. Sometimes the combination of high qualified parents can produce a low qualified offspring. So all the high qualified parents will be promoted to next generation directly. This process is time-consuming [60].

7.2.5 Non-Iterative Aggregation(Selective)

Majority Vote (MV): MV is a straight method where a specific value is chosen without preprocessing. We will calculate the probability of a specific label based on the majority vote over the total number of answers received for a question. However, there is a high chance that the system will fail. The system also does not consider the knowledge level of every worker [88].

Honeypot (HP): This method is same like MV but workers will be filtered during the preprocessing step to avoid

cheaters. The filtering is done by including a set of questions whose answers already known. These questions will be included along with the other questions and all those who fail to get a specified number of these questions right will be categorized as cheaters and removed. The possible labels are computed using MV method [91]. The problem faced in this model is that, if the toughness of the filtering questions used is high, some of the genuine workers might be identified as spammers.

Expert Label Injected Crowd Estimation (ELICE): This model is an extension of HP and filtering questions are used to determine the expertise level of each worker [83]. The expertise level is decided by taking the ratio of correct answers given by a worker to the total number of asked questions. It also determines the toughness of these questions by analyzing the number of workers got the specific question right. Then logistic regression method is used to calculate the probability of an answer being correct. This method addresses not only the expertise level of the worker but also the toughness of questions. As all the questions are weighted by expertise and toughness, the probability that the answer is correct will be well balanced. ELICE also faces the same problem faced in *Honeypot* model, where a genuine worker might be identified as a spammer depending on the toughness of filtering questions asked.

7.2.6 Others

This section summarizes the other methods used for quality control.

Major decision approach: This model [67] takes the same assumptions as those in the *Control Group Approach* and chooses the majority answer as the result to the requester. Specifically, when a task is created and distributed, the model assures that the same task will be duplicated and will be given to i number of workers. These workers will submit their answers back to the model and from all the answers, the majority answer will be chosen as the result to the requester. All the workers who submit the task will be paid. The total cost of this model is $C_{MD} = C_1.m + p_w.C_{fp}$. Here C_1 denotes the cost of successfully completed tasks, m denotes the number of workers, p_w denotes the probability that a task result is wrong and C_{fp} denotes the penalty given for accepting an invalid result.

Expectation maximization algorithm: This algorithm follows an iterative model and contains two steps. In the first step, it estimates the correct answer by analyzing all the submissions from workers [54]. In the second step, the estimated answer of each task is compared with the label submitted by the worker to figure out the quality of each worker. After this, the final output of the algorithm will be a *Confusion Matrix* containing all the possible error probabilities of each worker. From this matrix, the overall error rate can be obtained by adding up the non-diagonal elements.

Expectation Maximization Algorithm has some limitations. The system did not account for the users who had biased answers. Right now what system does is to reject all such answers which demotivate workers. Ipeirotis et al. [75] came up with a different idea which fixes the problems faced before. Authors set up an experiment where workers had to label websites into two categories. Once the EM algorithm is run, workers' error rate is known. With this known error rates, authors transformed each label given by worker, to another label which is the best possible estimate for the assignment. As the ultimate aim is to find the cost required to estimate the actual label, after normalizing, the cost is found using the equation:

$$Cost(p) = \sum_{i=1}^{L} \sum_{j=1}^{L} p_i.p_j.c_{ij}. \tag{12}$$

Here p_i is the probability that an object is classified to class i and p_j, the probability that an object is classified to class j. c_{ij} denotes the cost required to classify an object of class i to class j and L denotes the total possible classes. The cost will be zero if the classification is correct, and one otherwise. This Model, proposed in [75], considered the bias problem faced in Expectation Maximization Algorithm and is used well for quality controlling [85].

7.3 Pros and Cons of Quality Control Models

Table 4 on page 16 gives pros and cons of all quality control techniques mentioned above.

8 Real Case Scenario

To get a better understanding of the crowdsourcing process, in this section, we demonstrate a real crowdsourcing scenario with the help of *gMission* [48], [47]. We will explain different modules of the system with the algorithms currently used. We have also compared the algorithms used in gMission with other developed algorithms. Furthermore, we have highlighted some solutions which can be used to improve the functioning of gMission. As mentioned before, gMission is an open-sourced general spatial crowdsourcing platform developed by Dr. Lei Chen's group at the Hong Kong University of Science and Technology. gMission uses credits as incentive. Instead of giving free credits to motivate users, a user has to earn credits by first answering different posted tasks. Later, using those earned credits, one can post his/her own tasks to get answers. While posting a task, the requester has to mention how many credits he or she is willing to pay for the whole task and the number of people he or she wants to work on the task. Once the task is completed by the mentioned number of people, it will be removed from the job pool and the credit will be split equally among the workers. The system currently restricts each task with maximum 20 credits and 10 workers in total. gMission has a 'campaign' section for the requesters having more requirements. In the campaign section, the requester can post many questions and can let any number of people answer each question. The workers will get one credit for answering each question. This section can be used for tasks like image-labelling and video annotation. To motivate workers, in this case, every week, users are ranked based on the credits they earned and Starbucks coupons are given to the top three contributors.

While we believe that the current incentive type and model used in gMission are effective, the system can be improved considerably by making a few changes. First, instead of just adding up credits while completing the tasks, the system can include a combination of a level system like *Yahoo! Answers* and a privilege system like *Stack Overflow*

TABLE 4
Pros and Cons of Quality Control Models

Category	Technique	Pros	Cons
Pre-task execution	Gold Standards	• Easy implementation • Efficient feedback system • High productivity	• More resource needed • More time required • Difficult to find workers with matching requirement
	Worker's Skill Estimation	• Reduces the error rate • Easily scalable	• Complex implementation
Post-task execution	CDAS	• Higher success rate • Less processing cost	• Longer query response time
	Robust Personal Classifier	• Improves robustness • Clear differentiation from malicious and good worker	• Longer processing time
	Find - Fix - Verify	• Increases the productivity • Easy implementation	• Single worker's delay may cause the chain delay
	Control Group Approach	• Improves the result quality	• More human resource needed • More time required
	Major Decision Approach	• Higher success rate • Easy implementation	• Hard to deal that answers hold a weightage of 50 percent on both sides.
	Expectation Maximization Algorithm	• Individual error rates are considered	• The system fails when true responses are unknown • Longer processing time
	Supervised Learning from Multiple Experts (SLME)	• Uses sensitivity and specificity instead of confusion matrix	• Incompatible with multiple labels
	Generative model of Labels, Abilities and Difficulties (GLAD)	• Considers worker expertise and question difficulty	• The performance depends on the initial value of worker expertise and question difficulty
	Iterative Learning (ITER)	• Considers worker expertise and question difficulty individually	• Complicated implementation
	Free Choice Model	• Can solve design problems	• Very time consuming • Hard to scale
	Majority Vote (MV)	• Easy implementation	• Do not consider different expertise level of worker • System fails if most of them are spammers
	Honeypot (HP)	• Untrustworthy workers are filtered initially • Easy implementation	• Difficult trapping questions will lead to the misidentification of genuine users.
	Expert Label Injected Crowd Estimation (ELICE)	• Uses trapping questions to determine the expertise level • Easy implementation	• Difficult trapping questions will lead to the misidentification of genuine users

[30], [22]. Right now, all the users in the system, including the new users and the experienced users, are considered the same. If gMission keeps a system, in which a user with more points can ask more questions a day or a user with more points can access the statistics of the site (It is not easy to get access to real-time user interaction data.), there will more competition among users to achieve more points, which means more tasks can be completed. This can be implemented based on levels. Every new user will start from Level 1 and he/she can go up a level after achieving a specific number of credits. Once the user reaches a level, a set of privileges will be unlocked which he/she will get access to. The gaming industry and websites like Yahoo! Answers and Stack Overflow have clearly shown the impact of using a level—privilege based mechanism to make the system more interesting.

Even though gMission has not considered monetary incentives, it will be good to have such a system. With *Rating and Reward Dividing Model* which considers heterogeneous skill set of workers, their reputation and others, incentives can be given based on the combination of the quality of work submitted and the workers' expertise level [133]. The advantage of this system is that gMission can generate some money as service charge from the requesters which can be used to maintain the servers and pay the developers.

For task assignment, gMission uses *K-nearest neighbors* algorithm. This has been working fine so far as the tasks posted are based on locations. So the algorithm will pick workers near to the task location and push the tasks to them. Users can see the tasks on their cellphone, accept the task if they like, complete the task and earn the credits. Like mentioned before, this system is good but it can be improved if we consider workers' expertise before assigning them a task. Right now, when a user is registered in gMission, the system has little information about the user's expertise. In this scenario, assuming that there will be enough users registered, we can use the *Dual Task Assigner* method for task assignment [69]. This method will learn the user's behavior and determine the expertise on the fly. As this method also considers heterogeneous tasks, this will be apt for gMission.

If gMission incorporates *Dual Task Assigner* method to the currently used *K-nearest neighbors* algorithm, there is a high possibility to have a scenario where there is a worker close to the specified location with less expertise compared to another expert worker who is a little far. To tackle this scenario, gMission can include a technique in which the tasks will be pushed only to those workers inside a specified radius. This technique should be done real-time so that the worker walking inside the specified radius should also be able to get the task. The key idea behind this technique is that the answers received from an expert worker, who is a little far from the location, will be better than the answers received from a non-expert worker who is closer.

As of now, *gMission* uses majority voting for the result aggregation. If gMission uses *Dual Task Assigner* method mentioned before, there is no need to use any other pre-task quality control methods. Dual Task Assigner will identify the expertise level of each user and based on the questions, tasks can be assigned to users. If gMission wants to use some other task assignment method, then it can use the *Worker's Skill Estimation* method discussed in [107]. This method will identify the skill level of the users using different factors and based on the skill level, gMission can decide which task should be assigned to whom.

For the post-task execution quality control, the suitable one for gMission is CDAS [92] because of the high success rate. With the power of Bayesian analyses which is used to employ the probability-based verification, and also with the help of expertise level of users which will be derived from Bayesian analysis, correct answers can be scrapped out. As the confidence level of each answer is updated regularly in real-time time, this technique can be very well used in gMission if the stream of answers coming from a location is high. For the aggregation method, even though the implementation is complex, Iterative Learning method proposed in [78] is highly recommended. The main reason to choose this method is because it not only considers users' expertise, but also takes the toughness of each question into consideration which is extremely helpful when gMission implements tasks with different complexity. The implementation of quality control and an improved aggregation method has being started for gMission.

gMission is used among students and a campaign was run recently. The task given was to take the photos of different parts of the university. The campaign was very successful and a lot of images were submitted. Those images were later used for different experiments in this project [49]. Like mentioned before, *gMission* is an open source product. The repository is available at *GitHub* for anyone to modify for their use. With the help of the crowd, who are ready to work and the developers, who love coding, gMission can be scaled to different levels and has a huge potential in the crowdsourcing industry.

9 Discussion

As we discussed before, although a lot of research and work has been done on crowdsourcing, there are still a lot of open research issues, which will be highlighted in this section. Basically, these open problems are related to people involved in crowdsourcing, algorithms used for task assignment and answer aggregation, and implemented platforms. We will discuss them one by one as follows.

With respect to the people involved in crowdsourcing, one of the fundamental problems is how to recruit workers [57], that is, finding the right set of workers, attracting and retaining them. As the type of tasks published in the system requires various level of knowledge on various topics for answering tasks, the selection of workers is always challenging. In addition, as workers are recruited over the Internet, there is no guarantee on their working behavior These factors could potentially sabotage the whole system. Even if we find the right users, how to attract them to complete the task is another big issue. There are mainly four methods used to attract users. First one is volunteering, in which the system does not have to be concerned about attracting workers as volunteering is their main motivation. Wikipedia is the perfect example for this [29]. The second one is the authoritative way where one can be pushed to do a job if the requester has the authority to do so. The third one is the widely accepted payment method where each user is paid for his or her work. But there are still a lot of research need to be conducted to identify how much one should be paid. Like mentioned before, a requester does not want to give away more money and the workers do not want to work for less payment. The fourth one is making users "pay" for one system to use another system. This method is widely seen while downloading some materials on the Internet. The website will ask the user to finish some survey or to type something to unlock the downloadable link in order to proceed to the next step. By finishing the survey to get the link, the user indirectly completes an online task. reCaptcha is a typical example for this, where a user types whatever is shown on the screen to confirm that the user is not a robot [126]. To retrain workers, GWAPs uses an entertaining solution [123]. Xiaomi takes a crowdsourcing approach to brainstorm most of its features. People contributed and are motivated to contribute more to the project by seeing how impactful their contributions are [77].

Even if somehow we could pull workers to do tasks, there are some human-limitations associated with crowd because of which some tasks cannot be done. For example, crowd cannot be used to do accurate predictions. In the experiment conducted by Mr. Shane Killian, users were supposed to predict the exact position of roulette ball in the middle of spinning and the results showed how difficult it is to predict the position accurately considering that the ball is in spinning motion rather than static [84]. Crowd might be able to successfully predict something about static objects, but it is extremely tough to predict situations in which a lot of dynamic factors are involved [82].

With respect to algorithms for task assignment and answer aggregation, how to assign a task to authentic, knowledgeable workers, how to evaluate their work and how to combine them are the major research issues. Like mentioned before, the ultimate goal is to achieve the best possible answer for the task using least amount of resources. Even though we get some workers with a fair knowledge on the topic, there is no guarantee that best possible answers can be obtained. Based on different factors like the topic, the toughness of the topic, the toughness of the task, one may or may not answer. Also, the chance of getting the

correct answer is less if the worker does not understand the problem. As there is no messaging option available, inefficient way of structuring questions could also lead to no answers or wrong answers. So the ideal goal of crowdsourcing is yet to be achieved. It is also challenging to evaluate users' work. As the system has no idea about users' behavior, it is extremely tough to identify spammers and rule their work out. Spammers will always find a way to cheat the system and the system should be capable enough to identify those and remove them. Combining the answers is also another big issue. If the quality control and aggregation algorithms fail to do their task, the money that requesters spent on crowdsourcing projects will be wasted. So it is very important to make sure that these algorithms rule out spammers, assign right tasks to the right workers and find the correct answer without any mistake.

Finally, with respect to platforms, as the system has no idea about the workers joining the task force, there exist possibilities that the workers doing different tasks could leak the information to your rivals, which is a huge threat to users' privacy. Another concern is the Intellectual Property policies. Some workers provided answers which are highly relevant to the task posted by the requester. The requester decided to use the solution but later found that there was an IP violation. This is because the contributor later claimed that it is his or her work or he or she might have copied from somewhere else which the requester had no idea of. This is quite possible and the consequences could be disastrous. Moreover, submissions that are rejected by the requester will be under his/her possession. This could lead to potential copyright problems [127].

All of above discussed problems are the open research problems which need more research in the future.

10 Conclusion

Crowdsourcing sure is an amazing concept where a given problem is solved with the help of the crowd. With the help of this human-powered problem-solving paradigm, a lot of complex tasks can be solved using fewer resources. The main advantage of crowdsourcing over conventional problem-solving methods is that unlike traditional methods, in crowdsourcing, online workers are ready to work on different tasks for micro payments, which reduces the cost of a job drastically. Also, as the work is done on the Internet, the requester does not have to worry about setting up an infrastructure. But, crowdsourcing also has some disadvantages. As the entire process, such as recruitment, task assignment and result collection, is done on the Internet, the requester will not get a chance to meet any worker. Hence, the requester will not know whether a worker is genuine or a spammer as he or she does not have access to their personality data. Like that, there are other challenges as well in crowdsourcing.

In this paper, we have given a brief description of different technologies used to solve diverse problems faced in crowdsourcing. The main problems we have addressed in this work are proper designing of incentives to motivate workers, recruiting workers for tasks, overall quality control to increase the success rate and aggregating answers. We have given a brief study of different methods and technologies proposed by researchers to address above mentioned challenges. We have compared and discussed different techniques and have included pros and cons of each. However, to keep the work accurate to the point, we have not gone in depth of each method and explain every component like the interface design of softwares, in-depth flow of control, different formulae used and their proofs.

Crowdsourcing has a huge potential in various fields and we hope that more research will be done in this field to make the crowdsourcing system more stable and promising.

Acknowledgments

The work is partially supported by the Hong Kong RGC Project N_HKUST637/13, ANR-13-CORD-0020, National Grand Fundamental Research 973 Program of China under Grant 2014CB340303, the National Science Foundation of China (NSFC) under Grant No. 61502021 and 61328202, and NSFC Guang Dong Grant No. U1301253.

References

[1] Amazon Mechanical Turk. [Online]. Available: http://www.mturk.com/
[2] Amazon Mechanical Turk - Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Amazon_Mechanical_Turk
[3] Best crowdsourcing/crowdfunding websites. [Online]. Available: https://www.quora.com/What-are-the-best-crowdsourcing-crowdfunding-websites
[4] ClickWorker. [Online]. Available: http://www.clickworker.com/en
[5] Crowd Guru. [Online]. Available: http://www.crowdguru.org/
[6] Crowd4U Projects. [Online]. Available: http://crowd4u.org/en/projects
[7] CrowdFlower. [Online]. Available: http://www.crowdflower.com/
[8] CrowdFlower Lawsuit Could Change Crowd Labor Industry Forever. [Online]. Available: https://tinywork.wordpress.com/2013/07/29/crowdflower/
[9] From "TOYODA" to "TOYOTA". [Online]. Available: http://www.toyota-global.com/showroom/emblem/history/
[10] Gengo. [Online]. Available: http://gengo.com/
[11] Idea Bounty. [Online]. Available: http://www.ideabounty.com/
[12] InnoCentive. [Online]. Available: http://www.innocentive.com/
[13] Lingotek. [Online]. Available: http://www.lingotek.com/
[14] List of open innovation crowdsourcing examples. [Online]. Available: http://www.boardofinnovation.com/list-open-innovation-crowdsourcing-examples/
[15] MicroTask. [Online]. Available: http://microtask.com/
[16] OpenIDEO. [Online]. Available: http://openideo.com/
[17] Quora. [Online]. Available: http://www.quora.com/
[18] Quora Credits. [Online]. Available: https://www.quora.com/What-are-Quora-credits
[19] Samasource. [Online]. Available: http://www.samasource.org/
[20] Stack Overflow. [Online]. Available: http://stackoverflow.com/
[21] Stack Overflow Reputation & Moderation. [Online]. Available: http://stackoverflow.com/help/whats-reputation
[22] Stack Overflow User Privileges. [Online]. Available: http://stackoverflow.com/help/privileges
[23] The Long History of Crowdsourcing - and Why You're Just Now Hearing About It. [Online]. Available: http://www.crowdsource.com/blog/2013/08/the-long-history-of-crowdsourcing-and-why-youre-just-now-hearing-about-it/
[24] TopCoder. [Online]. Available: https://www.topcoder.com/
[25] UpWork. [Online]. Available: https://www.upwork.com/
[26] UpWork - Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Upwork
[27] What is a website similar to Amazon Mechanical Turk? [Online]. Available: https://www.quora.com/What-is-a-website-similar-to-Amazon-Mechanical-Turk
[28] wikiHow [Online]. Available: https://www.wikihow.com/
[29] Wikipedia. [Online]. Available: https://www.wikipedia.org/
[30] Yahoo Answers Points and Levels. https://https://answers.yahoo.com/info/scoring_system

[31] A. Amato, A. D. Sappa, A. Fornés, F. Lumbreras, and J. Lladós, "Divide and conquer: Atomizing and parallelizing a task in a mobile crowdsourcing platform," in *Proc. ACM Int. Workshop Crowdsourcing Multimedia*, 2013, pp. 21–22.

[32] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart, "CrowdMiner," *Proc. VLDB Endowment*, vol. 6, no. 12, pp. 1250–1253, 2013.

[33] D. Ariely, U. Gneezy, G. Loewenstein, and N. Mazar, "Large stakes and big mistakes," *Rev. Econ. Stud.*, vol. 76, pp. 451–469, 2009.

[34] S. Assadi, J. Hsu, and S. Jabbari, "Online assignment of heterogeneous tasks in crowdsourcing markets," in *Proc. AAAI Conf. Human Comput. Crowdsourcing*, 2015, pp. 12–21.

[35] S. Basu Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Task assignment optimization in knowledge-intensive crowdsourcing," *Proc. VLDB Endowment*, vol. 24, no. 4, pp. 467–491, 2015.

[36] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi, "Active sampling for entity matching," in *Proc. ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2012, pp. 1131–1139.

[37] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: A word processor with a Crowd Inside," in *Proc. ACM Symp. User Interface Softw. Technol.*, 2010, pp. 313–322.

[38] M. S. Bernstein, J. Teevan, S. Dumais, D. Liebling, and E. Horvitz, "Direct answers for search queries in the long tail," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 237–246.

[39] J. P. Bigham, S. White, T. Yeh, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, and B. White, "VizWiz: Nearly real-time answers to visual questions," in *Proc. ACM Symp. User Interface Softw. Technol.*, pp. 333–342, 2010.

[40] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan, "Asking the right questions in crowd data sourcing," in *Proc. IEEE Int. Conf. Data Eng.*, 2012, pp. 1261–1264.

[41] B. Bollobás, *Random Graphs*. Cambridge, U.K.: Cambridge Univ. Press, 1998, pp. 215–252.

[42] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence: Int. J. Res. New Media Technol.*, vol. 14, pp. 75–90, 2008.

[43] D. C. Brabham, "Moving the crowd at iStockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application," *First Monday*, vol. 13, pp. 1–16, 2008.

[44] K. T. Chan, I. King, and M.-C. Yuen, "Mathematical modeling of social games," in *Proc. 2009 Int. Conf. Comput. Sci. Eng.*, vol. 4, pp. 1205–1210, 2009.

[45] K.-T. Chen, C.-J. Chang, C.-C. Wu, Y.-C. Chang, and C.-L. Lei. (2010). Quadrant of euphoria. [Online]. Available: http://mmnet.iis.sinica.edu.tw/proj/qoe/

[46] K.-T. Chen, C.-J. Chang, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "Quadrant of euphoria: A crowdsourcing platform for QoE assessment," *IEEE Netw.*, vol. 24, no. 2, pp. 28–35, Mar./Apr. 2010.

[47] Z. Chen and P. Chen, gMission. [Online]. Available: http://www.gmissionhkust.com/

[48] Z. Chen, C. J. Zhang, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, and Y. Tong, "gMission," *Proc. VLDB Endowment*, vol. 7, pp. 1629–1632, 2014.

[49] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, "Reliable diversity-based spatial crowdsourcing by moving workers," *Proc. VLDB Endowment*, vol. 8, pp. 1022–1033, 2015.

[50] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, "KATARA: A data cleaning system powered by knowledge bases and crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2015, pp. 1247–1261.

[51] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and F. Players, "Predicting protein structures with a multiplayer online game," *Nature*, vol. 466, pp. 756–760, 2010.

[52] P. Dai, C. H. Lin, Mausam, and D. S. Weld, "POMDP-based control of workflows for crowdsourcing," *Artificial Intell.*, vol. 202, pp. 52–85, 2013.

[53] S. B. Davidson, S. Khanna, T. Milo, and S. Roy, "Using the crowd for top-k and group-by queries," in *Proc. Int. Conf. Database Theory*, 2013, pp. 225–236.

[54] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the EM algorithm," *Appl. Statist.*, vol. 28, pp. 20–28, 1979.

[55] G. Demartini, B. Trushkowsky, and T. Kraska, "CrowdQ: Crowdsourced query understanding," in *Proc. Biennial Conf. Innovative Data Syst. Res.*, 2013, pp. 137–140.

[56] N. R. Devenur and T. P. Hayes, "The adwords problem," in *Proc. ACM Conf. Electron. Commerce*, 2009, pp. 71–78.

[57] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the World-Wide Web," *Commun. ACM*, vol. 54, pp. 86–96, 2011.

[58] C. Eickhoff and A. P. D. Vries, "How crowdsourcable is your task?" in *Proc. Workshop Crowdsourcing Search Data Mining*, 2011, pp. 11–14.

[59] F. B. Fabian Kneissl, "MetropolItalia: A crowdsourcing platform for linguistic field research," in *Proc. IADIS Int. Conf. WWW/Internet*, 2012, p. 7.

[60] Y. Fang, H. Sun, R. Zhang, J. Huai, and Y. Mao, "A model for aggregating contributions of synergistic crowdsourcing workflows," in *Proc. AAAI Conf. Artificial Intell.*, 2014, pp. 3102–3103.

[61] S. Faridani, B. Hartmann, and P. Ipeirotis, "What's the right price? Pricing tasks for finishing on time," in *Proc. AAAI Conf. Artificial Intell.*, 2011, pp. 26–31.

[62] J. Freitas, A. Calado, D. Braga, P. Silva, and M. S. Dias, "Crowdsourcing platform for large-scale speech data collection," in *Proc. Jornadas en Tecnologa del Habla and II Iberian SLTech Workshop*, 2010, pp. 183–186.

[63] U. R. I. Gneezy and A. Rustichini, "Pay Enough or Don't Pay at All," *Quart. J. Econ.*, vol. 115, pp. 791–810, 2000.

[64] D. Haas, J. Ansel, L. Gu, and A. Marcus, "Argonaut: Macrotask crowdsourcing for complex data processing," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1642–1653, 2015.

[65] C. Harris, "You're hired! an examination of crowdsourcing incentive models in human resource tasks," in *Proc. Workshop on Crowdsourcing Search Data Mining*, 2011, pp. 15–18.

[66] K. Heimerl, B. Gawalt, K. Chen, T. Parikh, and B. Hartmann, "CommunitySourcing: Engaging local crowds to perform expert work via physical kiosks," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 1539–1548.

[67] M. Hirth, T. Hoßfeld, and P. Tran-Gia, "Analyzing costs and accuracy of validation mechanisms for crowdsourcing platforms," in *Proc. Math. Comput. Modelling*, 2013, vol. 57, pp. 2918–2932.

[68] C.-j. Ho, S. Jabbari, and J. W. Vaughan, "Adaptive task assignment for crowdsourced classification," in *Proc. Annu. Int. Conf. Mach. Learn.*, 2013, pp. 534–542.

[69] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proc. AAAI Conf. Artificial Intell.*, 2012, pp. 45–51.

[70] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *Proc. ACM Conf. Electron. Commerce*, 2010, pp. 209–218.

[71] J. Howe, "Crowdsourcing: A Definition," 2006. [Online]. Available: http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html

[72] J. Howe, "The rise of crowdsourcing," 2006. [Online]. Available: http://www.wired.com/2006/06/crowds/

[73] C. Hu, P. Resnik, Y. Kronrod, and B. Bederson, "Deploying monotrans widgets in the wild," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 2935–2938.

[74] P. G. Ipeirotis, "Be a top mechanical turk worker: You need $5 and 5 minutes - a computer scientist in a business school," 2010. [Online]. Available: http://www.behind-the-enemy-lines.com/2010/10/be-top-mechanical-turk-worker-you-need.html

[75] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proc. ACM SIGKDD Workshop Human Comput.*, 2010, pp. 64–67.

[76] H. Kajino and H. Kashima, "Convex formulations of learning from crowds," *Trans. Japanese Soc. Artificial Intell.*, vol. 27, pp. 133–142, 2012.

[77] M. Kan, "Xiaomi takes crowdsourced phone development model abroad," (2013). [Online]. Available: http://www.computerworld.com/article/2497141/mobile-wireless/xiaomi-takes-crowdsourced-phone-development-model-abroad.html

[78] D. R. Karger, S. Oh, and D. Shah, "Iterative learning for reliable crowdsourcing systems," in *Proc. Neural Inform. Process. Syst.*, 2011, pp. 1953–1961.

[79] G. Kazai, "An exploration of the influence that task parameters have on the performance of crowds," in *Proc. CrowdConf.*, 2010.

[80] G. Kazai, J. Kamps, and N. Milic-Frayling, "Worker types and personality traits in crowdsourcing relevance labels," in *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, 2011, pp. 1941–1944.

[81] L. Kazemi and C. Shahabi, "GeoCrowd: Enabling query answering with spatial crowdsourcing," in *Proc. Int. Conf. Advances Geographic Inf. Syst.*, 2012, pp. 189–198.

[82] R. Keith, "When crowdsourcing doesnt work," (2013). [Online]. Available: http://crowdsourcingweek.com/blog/when-crowdsourcing-doesnt-work/

[83] F. Khattak and A. Salleb-Aouissi, "Quality control of crowd labeling through expert evaluation," in *Proc. Workshop Comput. Social Sci. Wisdom Crowds*, 2011, pp. 1–5.

[84] S. Killian, "Testing the wisdom of crowds," (2010). [Online]. Available: https://www.youtube.com/watch?v=3Sgd1Ibnz2s

[85] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with Mechanical Turk," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2008, pp. 453–456.

[86] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[87] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 2012, pp. 1003–1012.

[88] L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin, "Limits on the majority vote accuracy in classifier fusion," *Pattern Anal. Appl.*, vol. 6, pp. 22–31, 2003.

[89] W. Lasecki, C. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. Kushalnagar, and J. Bigham, "Real-time captioning by groups of non-experts," in *Proc. ACM Symp. User Interface Softw. Technol.*, 2012, pp. 23–34.

[90] J. Le, A. Edmonds, V. Hester, and L. Biewald, "Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution," in *Proc. SIGIR Workshop Crowdsourcing Search Evaluation*, 2010, pp. 21–26.

[91] K. Lee, J. Caverlee, and S. Webb, "The social honeypot project: Protecting online communities from spammers," in *Proc. Int. Conf. World Wide Web*, 2010, pp. 1139–1140.

[92] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang, "CDAS: A crowdsourcing data analytics system," *Proc. VLDB Endowment*, vol. 5, pp. 1040–1051, 2012.

[93] Z. Liu, L. Luo, and W.-J. Li, "Robust crowdsourced learning," in *Proc. IEEE Int. Conf. Big Data*, 2013, pp. 338–343.

[94] A. Lynch, "Crowdsourcing is not new - The history of crowdsourcing (1714 to 2010)," 2010. [Online]. Available: http://blog.designcrowd.com/article/202/crowdsourcing-is-not-new–the-history-of-crowdsourcing-1714-to-2010

[95] A. Marcus and A. Parameswaran, "Crowdsourced data management: Industry and academic perspectives," *Found. Trends Databases*, vol. 6, no. 1-2, pp. 1–161, 2015.

[96] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller, "Human-powered sorts and joins," *Proc. VLDB Endowment*, vol. 5, pp. 13–24, 2011.

[97] W. Mason and D. J. Watts, "Financial incentives and the "performance of crowds"," in *Proc. ACM SIGKDD Workshop Human Comput.*, 2009, vol. 11, pp. 77–85.

[98] A. Moreno, J. L. De La Rosa, B. K. Szymanski, and J. M. Barcenas, "Reward system for completing FAQs," *Frontiers Artificial Intell. Appl.*, vol. 202, pp. 361–370, 2009.

[99] A. Morishima, N. Shinagawa, T. Mitsuishi, H. Aoki, and S. Fukusumi, "CyLog/Crowd4U," *Proc. VLDB Endowment*, vol. 5, pp. 1918–1921, 2012.

[100] K. K. Nam, M. S. Ackerman, and L. a. Adamic, "Questions in, knowledge in?: A study of naver's question answering community," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2009, pp. 779–788.

[101] M. Negri, L. Bentivogli, and A. Marchetti, "Divide and conquer : Crowdsourcing the creation of cross-lingual textual entailment corpora," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2011, pp. 670–679.

[102] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos, "Platemate: Crowdsourcing nutritional analysis from food photographs," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 1–12.

[103] A. Parameswaran and N. Polyzotis, "Answering queries using humans, algorithms and databases," in *Proc. Conf. Innovative Data Syst. Res.*, 2011, pp. 160–166.

[104] H. Park, R. Pang, A. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom, "An overview of the deco system," *ACM SIGMOD Record*, vol. 41, pp. 22–27, 2013.

[105] H. Park and J. Widom, "Query optimization over crowdsourced data," *Proc. VLDB Endowment*, vol. 6, pp. 781–792, 2013.

[106] N. Quoc Viet Hung, N. T. Tam, L. N. Tran, and K. Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *Proc. Web Inf. Syst. Eng.* 2013, pp. 1–15.

[107] H. Rahman, S. Thirumuruganathan, S. B. Roy, S. Amer-Yahia, and G. Das, "Worker skill estimation in team-based tasks," *Proc. VLDB Endowment*, vol. 8, pp. 1142–1153, 2015.

[108] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy, "Supervised learning from multiple experts: Whom to trust when everyone lies a bit," in *Proc. Annu. Int. Conf. Mach. Learn.*, 2009, pp. 1–8.

[109] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson, "Who are the crowdworkers?: Shifting demographics in mechanical turk," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2010, pp. 2863–2872.

[110] Y. Roth, "11 of the 12 Best Global Brands use creative crowdsourcing," 2012. [Online]. Available: http://yannigroth.com/2012/03/23/xx-of-the-100-best-global-brands-use-creative-crowdsourcing/

[111] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[112] R. N. Salas, "Humanity, teamwork, and art in post-earthquake Nepal," *New England J. Med.*, vol. 373, pp. 205–207, 2015.

[113] V. S. Sheng, F. Provost, and P. G. Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 614–622.

[114] M. S. Silberman, L. Irani, and J. Ross, "Ethics and tactics of professional crowdwork," *XRDS: Crossroads*, vol. 17, pp. 39–43, 2010.

[115] G. Smith and H. Rudge-Pickard, "Longitude problem," (2015). [Online]. Available: http://crazysquirrel.com/computing/software/vrml/essay/longitude_problem.jspx.

[116] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng, "Cheap and fast but is it good?: Evaluating non-expert annotations for natural language tasks," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2008, pp. 254–263.

[117] A. Sorokin and D. Forsyth, "Utility data annotation with amazon mechanical turk," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog.*, 2008, pp. 1–8.

[118] K. Starbird, "Digital volunteerism during disaster: Crowdsourcing information processing," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 1–4.

[119] C. Sun, N. Rampalli, F. Yang, and A. Doan, "Chimera: large-scale classification using machine learning, rules, and crowdsourcing," *Proc. VLDB Endowment*, vol. 7, no. 13, pp. 1529–1540, 2014.

[120] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen, "CrowdCleaner: Data cleaning for multi-version data on the web via crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, 2014, pp. 1182–1185.

[121] K. Tuite, N. Snavely, D.-y. Hsiao, N. Tabing, and Z. Popovic, "PhotoCity: Training experts at large-scale image acquisition through a competitive game," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 1383–1392.

[122] University of Tsukuba, "FusionCOMP," (2009). [Online]. Available: http://www.kc.tsukuba.ac.jp/~fusioncomp/index.html

[123] L. von Ahn, "Games With A Purpose (GWAP)," (2008). [Online]. Available: http://www.gwap.com/

[124] L. von Ahn, "Duolingo," in *Proc. Int. Conf. Intell. User Interfaces*, 2013, pp. 1–2.

[125] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2004, vol. 6, pp. 319–326.

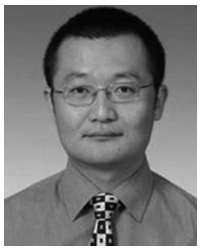
[126] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-based character recognition via web security measures," *Science*, vol. 321, pp. 1465–1468, 2008.

[127] P. Wagorn, "The Problems with Crowdsourcing and how to Fix them," (2014). [Online]. Available: https://www.ideaconnection.com/blog/2014/04/how-to-fix-crowdsourcing/

[128] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER," *Proc. VLDB Endowment*, vol. 5, pp. 1483–1494, 2012.

[129] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proc. Int. Conf. Manag. Data*, 2013, pp. 229–240.
[130] S. E. Whang, P. Lofgren, and H. Garcia-Molina, "Question selection for crowd entity resolution," *Proc. VLDB Endowment*, vol. 6, pp. 349–360, 2013.
[131] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *Proc. Neural Inf. Process. Syst.*, 2009, pp. 2035–2043.
[132] F. Wu, D. M. Wilkinson, and B. a. Huberman, "Feedback loops of attention in peer production," in *Proc. IEEE Int. Conf. Comput. Sci. Eng.*, 2009, vol. 4, pp. 409–415.
[133] H. Xie, J. C. Lui, J. W. Jiang, and W. Chen, "Incentive mechanism and protocol design for crowdsourcing systems," in *Proc. Annu. Allerton Conf. Commun., Control, Comput.*, 2014, pp. 140–147.
[134] G. Xintong, W. Hongzhi, Y. Song, and G. Hong, "Brief survey of crowdsourcing for data mining," *Expert Syst. Appl.*, vol. 41, pp. 7987–7994, 2014.
[135] X. Yin, W. Liu, Y. Wang, C. Yang, and L. Lu, "What? How? Where? A survey of crowdsourcing," in *Frontier and Future Development of Information Techology in Medicine and Education*, New York, NY, USA: Springer, 2014, vol. 269, pp. 221–232.
[136] L. Yu and J. V. Nickerson, "Cooks or cobblers?" in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 1393–1402.
[137] M.-C. Yuen, I. King, and K.-S. Leung, "A Survey of crowdsourcing systems," in *Proc. IEEE Int. Conf. Privacy, Security, Risk, Trust, IEEE Int. Conf. Social Comput.*, 2011, pp. 766–773.
[138] M.-C. Yuen, I. King, and K.-S. Leung, "TaskRec: A task recommendation framework in crowdsourcing systems," *Neural Process. Lett.*, vol. 41, pp. 223–238, 2015.
[139] C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao, "Reducing uncertainty of schema matching via crowdsourcing," *Proc. VLDB Endowment*, vol. 6, pp. 757–768, 2013.
[140] Y. Zhang and M. van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2012, pp. 2140–2148.
[141] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "QASCA: A quality-aware task assignment system for crowdsourcing applications," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2015, pp. 1031–1046.
[142] Y. Zhou, D. Chakrabarty, and R. Lukose, "Budget constrained bidding in keyword auctions and online knapsack problems," in *Proc. Internet Netw. Econ.*, 2008, pp. 566–576.
[143] S. Zhu, S. Kane, J. Feng, and A. Sears, "A crowdsourcing quality control model for tasks distributed in parallel," in *Proc. SIGCHI Conf. Human Factors Comput Syst.*, 2012, pp. 2501–2506.

Anand Inasu Chittilappilly received the integrated MS degree in software engineering from the Vellore Institute of Technology, Vellore, India, in 2014. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research is focused on crowdsourcing.



Lei Chen received the BS degree in computer science and engineering from Tianjin University, China, in 1994, the MA degree from the Asian Institute of Technology, Thailand, in 1997, and the PhD degree in computer science from the University of Waterloo, Canada, in 2005. He is now an associate professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His current research interests include crowdsourcing on social networks, uncertain and probabilistic databases, Web data management, and multimedia. He currently serves as an associate editor-in-chief for the *IEEE Transactions on Data and Knowledge Engineering* and a Trustee Board Member of VLDB Endowment. He is a member of the IEEE.

Sihem Amer-Yahia is currently the director of research (DR1) at the National Center for Scientific Research in the Grenoble Informatics Laboratory. Her research interests include the intersection of large-scale data management and analytics with an application to the social web. She is a member of the Very Large Data Base (VLDB) Endowment and serves on the editorial boards of the *ACM Transactions on Database Systems*, the *VLDB Journal*, and *the Information Systems Journal*. She is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.