Chen PP, Sun HL, Fang YL *et al.* Collusion-Proof result inference in crowdsourcing. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(2): 351–365 Mar. 2018. DOI 10.1007/s11390-018-1823-6

Collusion-Proof Result Inference in Crowdsourcing

Peng-Peng Chen, *Student Member*, *CCF*, *ACM*, Hai-Long Sun*, *Member*, *CCF*, *ACM*, *IEEE*
Yi-Li Fang*, *Member*, *CCF*, *ACM*, and Jin-Peng Huai, *Fellow*, *CCF*, *Member*, *ACM*, *IEEE*

State Key Laboratory of Software Development Environment, School of Computer Science and Engineering Beihang University, Beijing 100191, China

Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing 100191, China

E-mail: chenpp@act.buaa.edu.cn; sunhl@buaa.edu.cn; fangyili@act.buaa.edu.cn; huaijp@buaa.edu.cn

Received April 17, 2017; revised January 29, 2018.

Abstract In traditional crowdsourcing, workers are expected to provide independent answers to tasks so as to ensure the diversity of answers. However, recent studies show that the crowd is not a collection of independent workers, but instead that workers communicate and collaborate with each other. To pursue more rewards with little effort, some workers may collude to provide repeated answers, which will damage the quality of the aggregated results. Nonetheless, there are few efforts considering the negative impact of collusion on result inference in crowdsourcing. In this paper, we are specially concerned with the Collusion-Proof result inference problem for general crowdsourcing tasks in public platforms. To that end, we design a metric, the worker performance change rate, to identify the colluded answers by computing the difference of the mean worker performance before and after removing the repeated answers. Then we incorporate the collusion detection result into existing result inference methods to guarantee the quality of the aggregated results even with the occurrence of collusion behaviors. With real-world and synthetic datasets, we conducted an extensive set of evaluations of our approach. The experimental results demonstrate the superiority of our approach in comparison with the state-of-the-art methods.

Keywords crowdsourcing, quality control, collusion, collaborative crowdsourcing, result inference

1 Introduction

Crowdsourcing aims at eliciting human intelligence from the crowd to accomplish tasks that are computationally expensive or even hardly solvable for computers[1-2]. Its success has been witnessed in diverse applications, ranging from simple tasks, e.g., image labeling[3], sentiment analysis[4], to complex tasks, e.g., handwriting recognition[5-6], video description[7], translation[8], and text editing[9].

To complete some crowdsourcing tasks, especially complex tasks[10-11], workers are increasingly seeking to collaborate with each other. Recent studies show that there usually exist hidden collaborative networks (CN)[12-13] among workers on a crowdsourcing platform during task processing. Although most existing crowdsourcing platforms do not explicitly support collaboration, workers can communicate with each other via online social networks or even in person[14]. The motivations behind the collaboration among workers can be two-fold. On one hand, some tasks are rather difficult; thus workers need the help from each other. On the other hand, workers seek to obtain more monetary rewards with little effort. Particularly, inspired by the second motivation, some workers may collude with each other to provide low-quality answers that are harmful for the quality of crowdsourcing results. In the following, we summarize three types of collusion behaviors in collaborative crowdsourcing[15].

1) *Duplicated Submission.* On public crowdsourcing platforms like Amazon Mechanical Turk[16], the reward that a worker can obtain depends on the quality of his/her work. Thus a group of workers may col-

Regular Paper

This work was supported partly by the National Basic Research 973 Program of China under Grant Nos. 2015CB358700 and 2014CB340304, the National Natural Science Foundation of China under Grant No. 61421003, and the Open Fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2017ZX-14.

*Corresponding Author

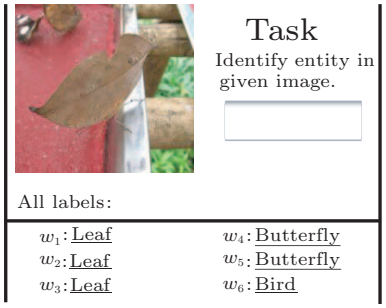
laboratively give the same answer to a task based on the collective intelligence of the group[17]. To guarantee that each member can obtain the reward, the group members will submit duplicated answers to the tasks that are processed by the group, which can damage the result quality due to the reduced diversity of answers.

2) *Group Plagiarism*. There can be some workers whose only objective is to earn as many rewards as possible; thus they are more prone to forming a group of colluders[18]. When one worker finishes all the tasks, others simply plagiarize his/her answers. In the worst case, no workers in the group process the tasks carefully, and the members just randomly forge an answer corresponding to a task.

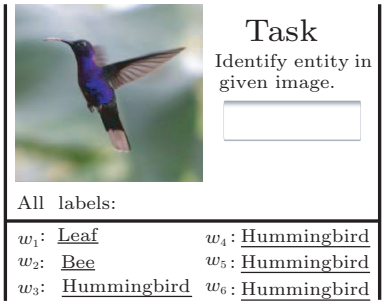
3) *Spam Accounts*. Some workers can register multiple accounts within one crowdsourcing platform, and then they can submit the same answer to a task for multiple times with different accounts[19]. This is often referred to as Sybil attacks, which was first introduced in distributed systems[20].

All the three kinds of collusion will result in some repeated answers being submitted, which will lower the quality of result inference.

For instance, we consider an image labeling task, in which each worker assigns one label to a picture. Fig.1(a) illustrates the labels given by six workers for a picture of a butterfly. For three of the six workers, two of them plagiarized the answer of the third one,



(a)



(b)

Fig.1. Two image labeling tasks. (a) With colluded answers. (b) Without colluded answers.

and then these three workers submitted the incorrect label *Leaf*. For the other three workers, two of them reported the correct label *Butterfly*, and the other one gave the label *Bird*. The voting algorithm consequently infers the final answer as *Leaf*, and then there is no chance to acquire the correct label *Butterfly*. Furthermore, it is easy to see that removing the two duplicated answers, *Leaf*, will yield the correct result. However, not all the repeated answers are generated by collusion behaviors. When tasks are easy to be handled or workers are with sufficient expertise, some of the labels submitted by independent workers are also likely to be correct and naturally the same as each other. For instance, Fig.1(b) illustrates the labels given by six independent workers for a picture of a hummingbird. Four workers reported the label *Hummingbird*. The labels *Bee* and *Leaf* are submitted by the other two workers respectively. If the three labels of *Hummingbird* are removed, the correct label *Hummingbird* cannot be obtained with the voting algorithm. Thus, determining whether the repeated answers given by workers are generated from collaboration or collusion is critical to the result inference in crowdsourcing.

There have been some efforts[21-23] concerning the collusion phenomenon in various domains. For instance, [21] studies the collaboration of bidders in all-pay auctions, but the authors only considered the collusion behaviors in the perspective of socio-economic. Literature [22] provides a detection method by computing the pairwise-similarity in the face of the collusion in the e-commerce platforms; however, the method is only limited to rating tasks. The work [23] is specially targeted at spacial crowdsourcing for data collection. The authors developed a statistical framework for collusion detection according to the spacial feature of datasets. However, the datasets from universal platforms (e.g., AMT or Crowdflower) do not contain this special feature. Thus, this method cannot be applied to tasks in universal platforms. In universal platforms, the hot spot is objectively grounded tasks, that is, the general tasks, such as single choice or sentiment analysis[1,24-25]. Nonetheless, no work has been seen to provide a solution to address the result inference issue for general tasks in crowdsourcing with collusion risks.

In this paper, we are concerned with obtaining high-quality results for general tasks even with the existence of collusion behaviors in potentially collaborative crowdsourcing. There are two major challenges to tackle the problem. First, it is challenging to detect the collusion behaviors. The collaboration of workers

that take place on today's crowdsourcing platforms cannot be observed directly, and the only possible means is to analyze the repeated answers submitted by workers. However, according to our preliminary analysis above, repeated answers do not necessarily imply collusion behaviors. Second, the mixing of collaborative answers and colluded answers will aggravate the difficulty of improving the quality of result inference in crowdsourcing. To address these two challenges, we study two issues in this paper, i.e., collusion detection and Collusion-Proof result inference. The former is mainly responsible for detecting the collusion behaviors by observing the result repetition and filtering out the repeated results introduced by collusive behaviors. The latter aims at achieving high-quality aggregated results using the output of collusion detection.

Specifically, in this paper, we introduce the worker performance change rate to measure the impact of the repeated answers on the quality of result inference. Then we develop a collusion detection mechanism named Collusion-Proof to identify answers generated from collusion behaviors. Generally speaking, the result inference methods fall into three categories: voting based methods, e.g., majority voting (MV)[26-27], worker ability based methods, e.g., Dawid and Skene model (DS)[28] and homogeneous Dawid and Skene model (HDS)[29-30], and task difficulty and worker ability based methods, e.g., generative model of labels, abilities, and difficulties (GLAD)[31]. By incorporating the results of collusion detection into majority voting (MV)[26-27], Dawid and Skene model (DS)[28], homogeneous Dawid and Skene model (HDS)[29-30], and generative model of labels, abilities, and difficulties (GLAD)[31], we propose C-MV, C-DS, C-HDS, and C-GLAD respectively for result inference in potentially collaborative crowdsourcing. To summarize, we make the following contributions.

1) We introduce a novel method to detect the collusion behaviors by computing the worker performance change rate caused by the repeated answers.

2) We design four result inference approaches for guaranteeing the quality of crowdsourcing results even in the case of collusion. To the best of our knowledge, this is the first attempt to provide a solution to the result inference in the consideration of the collusion among workers for general tasks.

3) We have conducted a set of experiments on both synthetic and real datasets to demonstrate the effectiveness and advantages of our approaches in comparison with the-state-of-the-art result inference approaches.

The remainder of this paper is organized as follows. Section 2 discusses related work. We formalize the studied problem in Section 3. Section 4 describes the overall workflow of our approach. We introduce the collusion detection mechanism and the four result inference methods in Section 5. Section 6 presents the experimental setup and results. We then discuss the open issues of this work in Section 7 and make conclusions in Section 8.

2 Related Work

Quality control is one of the core issues in crowdsourcing. Many studies on quality control mainly concern task design and result inference[1-2]. There are numerous result inference methods proposed to obtain as high-quality aggregated results as possible given the possibly noisy answers from crowd workers, for instance, majority voting[26], Dawid and Skene model[28], homogeneous Dawid and Skene model[29-30] and generative model of labels, abilities, and difficulties (GLAD)[31]. These methods are designed for aggregating answers from independent workers without the consideration of collaboration among workers and thus are not effective for the crowdsourcing tasks with colluders.

With the development of crowdsourcing, increasingly more studies begin to leverage the collaboration of workers to improve the quality of crowdsourcing, particularly related to complex tasks, such as text editing[9-11], software development[32]. In task design, collaboration can boost the performance of the workers, thereby improving the quality of task processing[14,33]. Here, workers mainly engage in two types of collaboration. First, workers collaborate to deal with tasks through multiple stages and/or rounds asynchronously. Chang *et al.*[17] designed a system named Revolt which enables groups of workers to collaboratively label data through three stages: vote, explain and categorize for creating high-quality training labels for machine learning. As early as 2009, Bernstein *et al.*[9] suggested a framework for crowdwork that serves to decrease mistakes of handwriting recognition by structuring a multi-round collaboration among workers. Similarly, Ambati *et al.*[34] suggested a collaborative workflow model that would better support crowdwork for translation tasks. Their approach splits workflow between different sets of workers. Teevan *et al.*[35] built the MicroWriter, a system in which the microtasking and crowd work is used to support collaborative writing within preexisting groups. Second, workers interact with each other synchronously via group-based cooperation. Rahman *et*

al.[36] studied task assignment optimization in collaborative crowdsourcing for translation. In their work, the workers were asked to work collaboratively on Google Docs and edited one another's translation in order to create an agreed-upon version. Salehi *et al.*[14] presented Huddler, a system that enables the assembly of familiar crowd groups for creating short advertisements. These studies improve the quality of crowdsourcing by boosting the worker performance of task processing.

However, the collaboration among workers can bring some collusive behaviors. Some literature[21-23,37-38] illustrates the threat of collusive behaviors to crowdsourcing in terms of investigation tasks. Literature [22] provides a method of collusion detection in the face of non-adversarial collusion in crowdsourcing; however, the method is only effective for opinion-based rating tasks. The authors of [23] developed a statistical framework for collusion detection based on the spatial location. However, this method cannot be applied to the tasks of universal platforms since it is closely related to spatial features such as latitude and longitude that cannot be acquired in universal crowdsourcing platforms. As we have mentioned before, both of them aim at solving the collusion detection problem for specific types of crowdsourcing tasks, while general tasks are the hot spot in universal crowdsourcing platforms[1]. Additionally, [21] studies the collaboration of bidders in all-pay auctions, and the authors considered the collusion behaviors in crowdsourcing contests. With the consideration of the characters of collusive behaviors, [37] proposes a collusion resistant worker selection method which aims at preventing the selection of colluders in crowd sensing. [38] studies the problem of distributing such tasks to workers with the goal of maximizing task privacy even when workers collude.

In summary, different from existing studies, this paper not only concerns detecting collusion behaviors, but also aims at providing a solution to address the result inference issue for general tasks in crowdsourcing with collusion risks.

3 Problem Formulation

In this section, we first introduce the notations used in the paper as shown in Table 1, and then define the collusion detection and the Collusion-Proof result inference problems.

Table 1. Symbols and Descriptions

Symbol	Description
T	Task set
W	Worker set
$\mathcal{L}$	Answer set
$\mathcal{L}_k$	Repeated set
$\mathcal{L}_k^-$	Answer set without the repeated answers in $\mathcal{L}_k$
R	Inference result
R_k	Inference result of $\mathcal{L}_k^-$
P_i	Precision of worker i in $\mathcal{L}$
P_i^k	Precision of worker i in $\mathcal{L}_k^-$
l_{ij}	Answer of task j from worker i
$E(P)$	Expectation of P
C	Number of alternative answers
$Var(P)$	Variance of P
$WPCR_k$	Worker performance change rate

As for collusion detection, we aim at detecting and filtering out the collusive answers which have a negative impact on result inference. In collaborative crowdsourcing, the repeated answers can be categorized into two groups: collusive and independent. We use b which is a boolean variable to denote the categories of answers: $b = 1$ means collusive answers, while $b = 0$ means the independent answers which are not involved with collusion behaviors. The collusive answers are the outcome of three kinds of behaviors: duplicated submission, group plagiarism, and spam accounts. Whereas, not all the repeated answers are generated by collusion behaviors. When tasks are easy to be handled and workers are of high-expertise, some answers submitted by independent workers tend to be correct and naturally the same as one another. First, we introduce Definition 1.

Definition 1 (Collusion Detection Problem, CDP). *Let $T = \{t_j | j \in I_T\}$[①] be the task set, and $W = \{w_i | i \in I_W\}$ be the worker set. We denote the answer set as $\mathcal{L} = \{L_1, L_2, ..., L_n\}$ where $L_i \in \mathcal{L}$ consists of the answers that worker i reports for $T' \subseteq T$, namely, for $\forall w_i \in W$, we get the answer set $L_i = \{l_{ij} | j \in I_T\}$ from worker i. $\mathcal{L}' \subseteq \mathcal{L}$ denotes a collection of repeated answers. The problem of CDP is to find a function $f : \mathcal{L}' \to \{0, 1\}$, which detects collusive answers from a collection of repeated answers.*

When collusive answers are detected and filtered out, an important problem is to infer the final result from the multiple unreliable answers as high-quality as possible. Formally, we define the Collusion-Proof result inference problem as follows.

[①]Note I_X is an index set of set X.

Definition 2 (Collusion-Proof Result Inference Problem, CRIP). *Let* $T = \{t_j | j \in I_T\}$[②] *be the task set,* $W = \{w_i | i \in I_W\}$ *be the worker set, and* $\Omega = \{x_c | c \in I_\Omega\}$ *be the answer domain set. We denote the answer set* $\mathcal{L} = \{L^1, L^2, ..., L^m\}$ *where* $L^j \in \mathcal{L}$ *consists of the answers that workers report for* t_j*, namely, for* $\forall t_j \in T$*, we get the answer set* $L^j = \{l_{ij} | i \in I_W\}$ *from workers. Let* $G = \{g | g : \Omega^{|T|\times|W|} \to \Omega^{|T|}\}$ *be the universal set of aggregation algorithms. Then with the well-defined value function* $v : G \to R$ *which measures the quality of the algorithms, we can formulate the problem of CRIP so as to find a function* $g^* \in G$ *which can obtain the maximum* $v(g^*)$*.*

4 Overview of the Workflow

In this section, we present the overview of our workflow as shown in Fig.2, where task scheduling, collaborative crowdsourcing, and Collusion-Proof result inference are the three main steps.

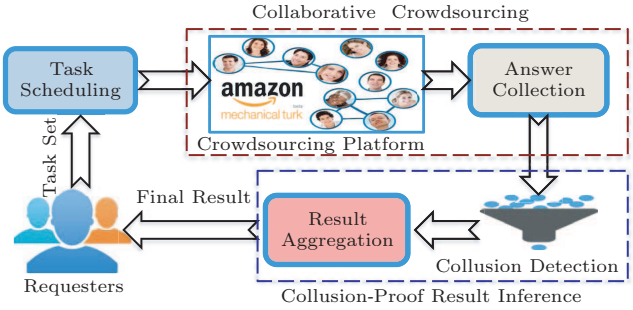


Fig.2. Collusion-Proof crowdsourcing workflow.

Task Scheduling. A requester publishes tasks to a crowdsourcing platform, e.g., Amazon Mechanical Turk in which requesters give the corresponding reward to a worker according to the quality of his/her answers. Tasks are assigned to workers according to the scheduling policies and user-specified constraints of the platform.

Collaborative Crowdsourcing. In practice, some crowdsourcing tasks can be processed collaboratively by a group of workers. Workers may collude with each other behind the scenes. For example, via online forums, a worker plagiarizes others who also do the same crowdsourcing work. After task processing, we collect answers and eliminate some noisy answers, e.g., some answers are apparently irrelevant to the picture in an image labeling task.

Collusion-Proof Result Inference. This step involves collusion detection and result aggregation. After collecting all the labels from workers, we use a collusion detection mechanism to detect the collusion behaviors, and then filter out the repeated answers generated by colluders. After the result filtering, we employ our result inference methods to infer the final result of each task and submit them to the requester.

Different from the general crowdsourcing workflow, the crowd in our workflow is not viewed as a collection of independent workers, instead, workers may collude with each other. In addition, we develop a collusion detection mechanism to detect the collusive behaviors of workers, and then employ our result inference methods to infer the high-quality result even in the case of collusion.

5 Collusion-Proof Result Inference

This section studies the CDP problem given by Definition 1 and the CRIP problem given by Definition 2. We mainly present Collusion-Proof to detect collusive behaviors, and then we develop C-MV, C-DS, C-HDS, and C-GLAD, which can filter out the repeated answers generated by the collusion behaviors and aggregate multiple unreliable answers to a credible one. First, in Subsection 5.1, we study the relationship between the occurrence probability of repeated answers and the ability of workers in collaborative crowdsourcing. Second, we present a method named Collusion-Proof to detect the collusion behaviors among workers by introducing the worker performance change rate in Subsection 5.2. Third, by incorporating detection results into MV, DS, HDS, and GLAD, we design C-MV, C-DS, C-HDS, and C-GLAD for guaranteeing the quality of crowdsourcing results in Subsection 5.3.

5.1 Analysis of Repeated Answers in Collaborative Crowdsourcing

The performance of workers on tasks is closely related to the answer set submitted by workers. To identify the repeated answers to be either collusive or independent, we start with studying the ability of independent workers on a repeated answer set.

We assume the tasks are with C classes. Given the answer set $\mathcal{L} = \{L_1, L_2, ..., L_n\}$, where $L_i \in \mathcal{L}$ consists of the answers that worker i reports for $T' \subseteq$ T, for $\forall w_i \in W$, we get the answer set $L_i = \{l_{ij} | j \in I_T\}$ from worker i. Let a_i be the ability of worker i.

[②]Note I_X is an index set of set X.

Let z_j be the ground truth of task j, and then if worker i reports a correct label for task j, according to the homogeneous Dawid and Skene model[29], the probability that label l_{ij} submitted by worker i for task j can be given as follows.

$$P(l_{ij}|z_j, a_i) = \begin{cases} a_i, & \text{if } l_{ij} = z_j, \\ \dfrac{1-a_i}{C-1}, & \text{otherwise.} \end{cases} \quad (1)$$

Here, the stronger the ability of a worker, the greater the probability of correctly completing task. $l_{ij} = z_j$ means that the worker submits a correct answer, while $l_{ij} \neq z_j$ means that the worker submits an incorrect answer. In this case, each worker's ability is not changing along with different tasks.

We assume the results submitted by workers contain no collusive repeated answers, i.e., all workers in $W = \{w_i | i \in I_W\}$ independently perform tasks. We assume that there are N workers in the worker set and M tasks in the task set. The worker ability is assumed as at least 0.5. When a worker's ability is 0.5, this means that the probability that his/her answer is true is 0.5. This is a common assumption. To the best of our knowledge, most result inference methods can be viewed as the variants of majority voting. Literature [26] suggests that the result quality after result inference is lower than that of the answer from the individual worker when the worker ability is below 0.5. In this case, the result inference methods based on majority voting will make no sense. With (1), we present the following equation to formulate the relationship between the occurrence probability of repeated answers and the ability of workers. We can give the probability that r workers of ability a_i report repeated answers for M tasks.

$$P_r(M, C, a_i) = (a_i^r + \frac{(1-a_i)^r}{(C-1)^{r-1}})^M.$$

Fig.3 plots P_r with varying worker ability when there are 10 tasks and the number of alternative answers C is 3. In the theoretical analysis, we can observe that the occurrence probability of the repeated answer set submitted by independent workers increases with the worker ability. We consider two extreme situations. When the worker ability is 0.5 and $r = 2$, the occurrence probability of repeated answers is 1.3×10^{-12} which is a rather small value. This implies that the low-quality independent workers can hardly generate repeated answers. In this case, repeated answers are very likely to be from colluders. Whereas when the worker ability is 1, the occurrence probability of repeated answers is 1. In this case, repeated answers are all from independent answers.

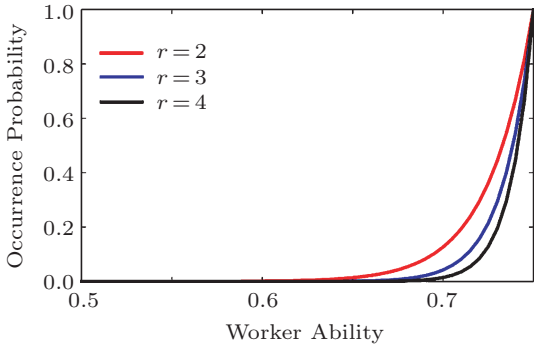


Fig.3. Occurrence probability of repeated answers with varying worker ability.

Thus we can identify collusion behaviors via analyzing repeated answers according to the worker ability, and then remove collusive answers. For instance, Fig.4 presents the answers submitted by five workers for 10 tasks, and each worker's ability is 0.6. With MV, the aggregated result accuracy we can achieve is 0.6. We can also observe that removing some repeated answers from worker w_1 to worker w_3 will improve the accuracy by 0.2.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w_1	A	C	A	B	B	B	C	C	A	A
w_2	A	C	A	B	B	B	C	C	A	A
w_3	A	C	A	B	B	B	C	C	A	A
w_4	B	B	B	A	B	B	A	A	B	C
w_5	A	C	B	A	A	A	A	A	B	C
MV	A	C	A	B	B	B	C	C	A	A
C-MV	A	C	B	A	B	B	A	A	B	C
Z	A	C	A	B	B	B	A	A	B	C

A Label A B Label B C Label C

Fig.4. Answers submitted by low-ability workers.

After the analysis of repeated answers, we learn that the stronger the ability of workers, the greater the occurrence probability of repeated answers. When workers are of low ability, the repeated answers are very likely to be collusive. When workers have high quality, most answers are correct and repeated. Even if some of them are collusive, related studies[26,39] have suggested that the impact of repeated answers on result inference is little. Thus, the worker ability inferred by the ground truth is an important factor to detect collusion. Actually, the ground truth is generally unknown. Therefore,

we will propose a method which can detect collusion behaviors without the ground truth.

5.2 Collusion Detection

In this subsection, we study the CDP problem given by Definition 1, and then introduce a metric, the worker performance change rate to develop a method named Collusion-Proof with the purpose of collusion detection.

Many studies[26,39] suggested that the ability of most workers in the crowd can be roughly considered equally. As discussed above, although there is no collusion, workers are much likely to provide repeated answers close to the correct answer when each worker possesses a good performance on the work, i.e., workers are of high ability. Sheng *et al.*[26] suggested that the performance of a collection of independent workers on a task set does not fluctuate too much when the worker ability is great. Since an individual worker performs pretty well and his/her answer set is very close to the aggregated result. In this case, it is clear that task processing with multiple workers is dispensable and removing some answers almost does not affect the closeness of the answers to aggregated result. That is to say, when some answers are removed in this case and the closeness of the answers to aggregated result varies little, the answers are very likely to be normal. Even if some answers are provided by high-quality collusive workers, it does not affect the quality of aggregated answers. Thus there is no need to filter out the consensus of the crowd even if there exist collusive workers. Whereas, when the closeness of answers to aggregated result changes obviously after removing some repeated answers, this means that workers are of low ability. In this case, these workers are much likely to be collusive. Consequently, we introduce the performance change rate related to the closeness of the answers to aggregated result.

Given answers $\mathcal{L} = \{\mathcal{L}_1, ..., \mathcal{L}_k, ..., \mathcal{L}_{n'}\}$, where $\mathcal{L}_k = (L_{k_1}, ..., L_{k_i}, ..., L_{k_{n''}})$ denotes the repeated answers from worker k_1 to worker $k_{n''}$ for task set T and L_{k_i} denotes the answers submitted by worker k_i. Note that $k_{n''} = 1$ means $\mathcal{L}_k$ is generated by a single worker. We can get aggregated answer set R corresponding to $\mathcal{L}$ with the inference methods as follows.

$$R = aggr\_method(\mathcal{L}, W, T), \tag{2}$$

where $W = w_i | i \in I_W$ is the worker set, $T = t_j | j \in I_T$ is the task set, and $aggre\_method(\cdot)$ is a universal formula for aggregation methods, such as majority voting[26], Dawid and Skene model[28], homogeneous Dawid and Skene model[29-30], and GLAD[31]. After removing the repeated answers of $\mathcal{L}_k$, we can obtain an answer set $\mathcal{L}_k^-$. As for $\mathcal{L}_k^-$, we can use (2) to infer the final result R_k.

Then the variance of the closeness of L_i to R can be used to formulate the performance change rate. If the ground truth is known beforehand, we can obtain the ability of a worker with respect to the closeness of his/her answers to the ground truth. However, the ground truth of tasks is generally unknown. Thus, we roughly apply the precision of a worker to estimate his/her worker ability and compute the precision of worker i corresponding to $\mathcal{L}$ as follows.

$$P_i = \frac{|L_i \cap R|}{|R|}.$$

Then we get the variance $Var(P)$.

$$Var(P) = \frac{1}{|\mathcal{L}|} \sum_{i=1}^{|\mathcal{L}|} \{P_i - E(P)\}^2,$$

where $E(P)$ is the expectation of P.

Similarly, after obtaining the precision of worker i corresponding to $\mathcal{L}_k^-$, namely, $P_i^k = \frac{|L_i \cap R_k|}{|R_k|}$, we can get the variance $Var(P^k)$ as follows.

$$Var(P^k) = \frac{1}{|\mathcal{L}_k^-|} \sum_{i=1}^{|\mathcal{L}_k^-|} \{P_i^k - E(P^k)\}^2,$$

where $E(P^k)$ is the expectation of P^k.

Finally, we define the performance change rate corresponding to $\mathcal{L}$ and $\mathcal{L}_k^-$.

$$WPCR_k = |Var(P^k) - Var(P)|.$$

When $WPCR_k$ is great, this means that repeated answers are much likely to be collusive. While $WPCR_k$ is small, this means that repeated answers are very likely to be from independent workers. Thus, we can use the worker performance rate to detect whether repeated answers are collusive or not.

An instance is presented in Fig.4. With answers L_i, we get that P_i of workers (from w_1 to w_5) is 1, 1, 1, 0.2, and 0.2 respectively. Then we obtain that $Var(P^k)$ is 0.56. When the repeated answers of workers (from w_1 to w_3) are removed, with answers $\mathcal{L}_k^-$ we get that P_i^k of each worker is 0.4, 0.4, 0.4, 0.8, and 0.8 respectively. Then we obtain that $Var(P^k)$ is 0.384. Finally, we can obtain that $WPCR_k$ is 0.176. This is a large value, and then repeated answers are filtered out. After filtering the result accuracy is improved by 0.2, from

0.6 to 0.8. Now we have introduced the worker performance change rate which serves to measure the impact of repeated answers on inferred results and used it as a criterion to detect the collusive behaviors. Then result quality can be improved by filtering the answers from colluders based on collusion detection results.

5.3 Result Aggregation

In this subsection, by incorporating the detection results into MV, DS, HDS, and GLAD, we obtain C-MV, C-DS, C-HDS, and C-GLAD. In practice, the answers submitted by some collusive workers may not always be the same for each task, which is also explained in [22]. However, most of the collusive answers should be the same or very similar. Otherwise the collusion will be meaningless. Therefore, we can still detect the collusion behaviors by analyzing repeated answers.

Then, we can derive C-MV, C-DS, C-HDS, and C-GLAD with Algorithm 1. Given an answer set $\mathcal{L}$, the decision can be made between two options: 1) filtering out repeated answers generated by colluders in the corresponding answer set; 2) preserving repeated answers in the corresponding answer set, and then implementing the result aggregation. Line 11 is the core of this algorithm which computes the worker performance rate of the answer set. *threshold* in line 12 determines whether it is necessary to filter out repeated answers in the corresponding answer set. In this paper, we conduct multiple experiments to set a proper threshold of Algorithm 1. We plan to employ machine learning to capture an optimal threshold in future work.

Algorithm 1. Result Inference Framework

Input: answer set $\mathcal{L}$
Output: inferred result R from $\mathcal{L}$
1 Initialization: $R = \emptyset$, $R' = \emptyset$;
2 **for** each repeated answer set $\mathcal{L}_k \in \mathcal{L}$ **do**
3 $R \leftarrow aggr\_method(\mathcal{L}, W, T)$;
4 $R_k \leftarrow aggr\_method(\mathcal{L}_k^-, W, T)$;
5 **for** each answer set $L_i \in \mathcal{L}$ **do**
6 $P_i = \frac{|L_i \cap R|}{|R|}$;
7 **for** each answer set $L_i \in \mathcal{L}_k^-$ **do**
8 $P_i^k = \frac{|L_i \cap R_k|}{|R_k|}$;
9 $Var(P) = \frac{1}{|\mathcal{L}|} \sum_{i=1}^{|\mathcal{L}|} (P_i - E(P))^2$;
10 $Var(P^k) = \frac{1}{|\mathcal{L}_k^-|} \sum_{i=1}^{|\mathcal{L}_k^-|} (P_i^k - E(P^k))^2$;
11 $WPCR_k = |Var(P^k) - Var(P)|$;
12 **if** $WPCR_k \geqslant threshold$ **then**
13 $\mathcal{L}_k \leftarrow \mathcal{L}_k^-$; % Filter out repeated answers generated by colluders
14 $R \leftarrow aggr\_method(\mathcal{L}, W, T)$;
15 **return** inferred result R;

6 Experiments

In this section, we present the evaluation results of our proposed methods in comparison with the-state-of-the-art methods with synthetic data and real datasets. In experiments with the synthetic data, we evaluated C-MV, C-DS, C-HDS, and C-GLAD with varying the worker ability and the number of independent workers. Based on the real data, we analyzed the existence of repeated answers generated by suspected colluders, evaluated Collusion-Proof, and evaluated C-MV, C-DS, C-HDS, and C-GLAD③.

6.1 Experimental Setting

Due to the distributed and anonymous nature of the workers from the crowdsourcing platforms (e.g., AMT or Crowdflower), it is difficult to capture a set of workers of the same ability in the real world. In simulation experiments, we aimed at exploring the effect of the threshold on the effectiveness of Algorithm 1 and exploring whether C-MV, C-DS, C-HDS, and C-GLAD outperform the-state-of-the-art methods when the worker ability changes. Moreover, we conducted the experiment to evaluate C-MV, C-DS, C-HDS, and C-GLAD with synthetic datasets of different numbers of independent workers.

In experiments with real datasets, we aim to perform the evaluation: 1) whether there exist repeated answers yielded by suspected colluders in real-world datasets; 2) whether colluders can be effectively detected by Collusion-Proof; 3) how C-MV, C-DS, C-HDS, and C-GLAD can improve the result quality with varying number of collusive groups and the collusion proportion.

We used precision, accuracy, and recall for evaluating the performance of Collusion-Proof and utilized the accuracy of aggregated answers after result inference to evaluate the performance of C-MV, C-DS, C-HDS, and C-GLAD. A baseline collusion detection method named Findcolluders and four baseline result inference methods (i.e., MV, DS, HDS, and GLAD) were also implemented.

Findcolluders. For the e-commerce platform, [22] detects collusive behaviors by computing the pairwise-similarity.

MV. It is the simplest and the most popular strategy. MV counts the votes for each alternative answer, and then assigns the majority answer as the final result for each task[26-27].

③We have released our code on https://github.com/cplzyangbuaa/collusion-proof.git, Oct. 2017.

DS. This inference method is an implementation of the approaches designed for single choice tasks and it assumes that the alternative answers are the same for different tasks[28].

HDS. This is a variation of DS[29-30].

GLAD. It estimates workers' performance by considering their expertise levels and the difficulty levels of tasks[31].

Since we are concerned with the general tasks like multi-class label task, for which DS is a typical method, the comparisons of aggregated accuracy obtained with DS and C-DS are presented with figures in the subsections concerning the evaluation of the effectiveness of result reference methods, i.e., Subsections 6.2.3, 6.2.4 6.3.4, and 6.3.5. For the sake of clear presentation, the comparisons of aggregated accuracy obtained with the other methods (i.e., MV, C-MV, HDS, C-HDS, GLAD, and C-GLAD) are presented with tables (i.e., Table 2∼Table 7).

6.2 Experiments with Synthetic Data

6.2.1 Generation of Synthetic Data

We generated synthetic data roughly modeled to have the same properties as the real dataset. We first simulated a set of tasks: multi-class label tasks which embrace four alternative answers and only one of them is true (the alternative answers do not vary in different tasks). The number of tasks is set to 2 000. DS is a typical and effective model designed for multi-class label tasks[28,30]. To measure the quality of each worker, it endows each worker i with a latent "confusion matrix", which gives the probability that worker i will classify the object into category k when presented with an object of true class j. We used the performance functions with respect to "confusion matrix" in the DS model to simulate the task processing of workers. We simulated a set of workers and used the proportion c of diagonal entries of the confusion matrix to characterize their average ability. When a set of workers are of great c, their average ability is strong. We assumed that the distribution of the ground truth z follows the uniform distribution among candidate answers, and then we could obtain the simulated datasets. The performance of a collusion group follows a uniform distribution, which means our simulations can cover the three types of collusion behaviors described in Section 1. Moreover, some additional repeated answer sets were added to simulate the impact of colluders on answers. In this case, the number of answer sequences in an added repeated answer set corresponds to the number of colluders in a group and the number of added repeated answer sets corresponds to the number of collusive groups. The number of groups and the number of colluders in each group could be set according to the specific experimental scenario.

6.2.2 Threshold of Algorithm 1

We conducted 50 rounds of experiments to explore the effect of the threshold on the effectiveness of Algorithm 1. On the basis of the experimental results, we can determine an empirical value for the threshold. The total number of workers was 9 and three of them were colluders in a group. With Algorithm 1, the collusion detection results were incorporated into MV, and then C-MV were derived. Fig.5 plots the result accuracy with varying the threshold of Algorithm 1. We can observe that the accuracy grows with the increase of the threshold when the threshold is close to 0. The reason is that almost all the repeated answers will be viewed as collusive answers, even if some answers are from normal workers. As the threshold continues to grow, the accuracy decreases no matter how the average ability of workers varies. This is because a larger threshold means that more likely repeated answers will be regarded as normal answers even when some of them are collusive. We can also observe that when the worker ability is 0.85, the fluctuation of curves is relatively small. Because, when workers are of high expertise, removing or preserving repeated answers does not have a significant influence on the inference result. With multiple rounds of experiments on varying threshold values, the threshold is ultimately set to 0.028. As a result, most independent answers are preserved and almost all the collusive answers are removed. Since we adopted the

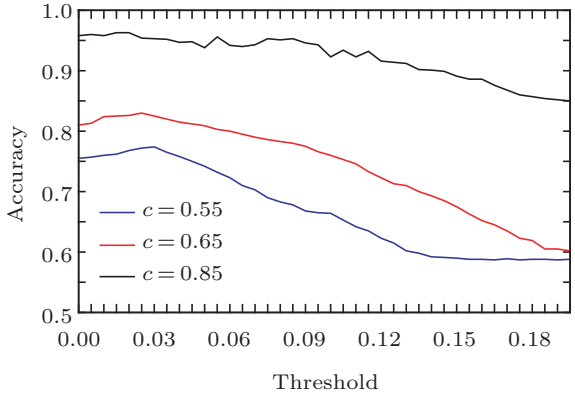


Fig.5. Accuracy with varying threshold.

similar settings for the rest of experiments, the threshold was set to 0.028 as well. As we have mentioned, in future, we will employ machine learning approaches to find the optimal threshold of Algorithm 1.

6.2.3 *Effectiveness of Result Inference with Varying Worker Ability*

In this experiment, there was one collusion group containing three colluders and each task was assigned to five workers. We evaluate the result accuracy of the eight inference methods.

Fig.6 and Table 2 illustrate the accuracy of the inferred results of these result inference methods with the increasing worker ability c. As the worker ability c grows, the accuracy of all the methods increases. C-MV, C-DS, C-HDS, and C-GLAD outperform the other four without collusion detection. In particular, the performance of C-DS is the best, which is 13.4%, 22.3%, 8.1%, and 23.2% better than that of MV, DS, HDS and GLAD on average respectively. The reason is that C-MV, C-DS, C-HDS, and C-GLAD can filter out repeated answers generated by the colluders, and C-DS model better targets the multi-class label setting in comparison with others.

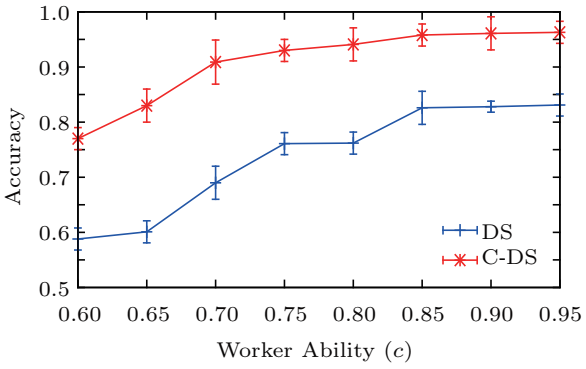


Fig.6. Accuracy with varying worker ability (DS, C-DS).

Table 2. Accuracy with Varying Worker Ability (the Others)

c	MV	HDS	GLAD	C-MV	C-HDS	C-GLAD
0.60	0.713	0.698	0.598	0.757	0.701	0.689
0.65	0.763	0.741	0.630	0.811	0.778	0.749
0.70	0.794	0.820	0.690	0.906	0.851	0.853
0.75	0.820	0.862	0.730	0.930	0.871	0.888
0.80	0.827	0.870	0.762	0.941	0.895	0.912
0.85	0.872	0.905	0.826	0.958	0.919	0.951
0.90	0.874	0.911	0.827	0.961	0.923	0.959
0.95	0.875	0.912	0.826	0.964	0.925	0.961

6.2.4 *Effectiveness of Result Inference with Varying Number of Independent Workers*

In this experiment, there were three colluders contained in a group and the number of independent workers varied from 2 to 5, which means the total number of workers ranged from 5 to 8.

The results are in Fig.7 and Table 3, and we can observe that more independent workers yield higher accuracy. C-MV, C-DS, C-HDS, and C-GLAD outperform their corresponding original methods, i.e., MV, DS, HDS, and GLAD. In particular, C-DS can obtain 11.4%, 25.1%, 24.9%, and 23.8% higher accuracy than MV, DS, HDS, and GLAD on average respectively.

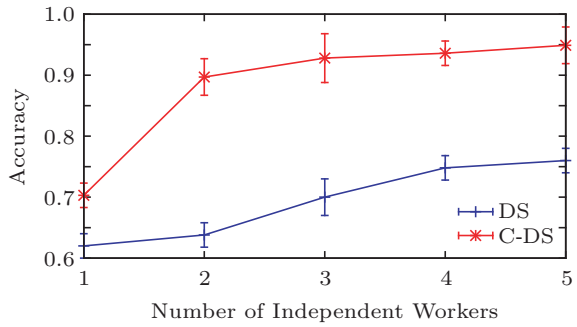


Fig.7. Accuracy with varying number of independent workers (DS, C-DS).

Table 3. Accuracy with Varying Number of Independent Workers (the Others)

	MV	HDS	GLAD	C-MV	C-HDS	C-GLAD
1	0.666	0.621	0.632	0.713	0.719	0.703
2	0.708	0.637	0.634	0.872	0.812	0.764
3	0.811	0.703	0.699	0.898	0.864	0.897
4	0.840	0.749	0.746	0.906	0.878	0.905
5	0.857	0.759	0.753	0.925	0.899	0.918

6.3 Experiments with Real Datasets

6.3.1 *Real Datasets*

We utilized two well-known real-world datasets, i.e., *ducks*④[40] and *adult2*⑤[41]. The task in dataset *ducks* is to identify whether an image contains a duck or not. It contains 39 tasks, each of which is answered by 108 workers, and each worker answers 39 tasks. The dataset *adult2* contains the categories (G, PG, R and X) of websites labeled by workers on MTurk. It contains 333 tasks. The average number of answers of each task is 10, and the number of tasks answered by each worker is 12.

④https://github.com/welinder/cubam/tree/public/demo/bluebirds, Jan. 2018.
⑤https://github.com/ipeirotis/Get-Another-Label/tree/master/data, Jan. 2018.

6.3.2 Analysis of Repeated Answers Yielded by Suspected Colluders

In crowdsourcing platforms, since the collusive behaviors are behind the scenes, it is difficult to detect the collusion during the task processing. Yielding repeated answers is an important feature of collusion. As we have mentioned, repeated answers can be broken down into two camps, i.e., collusive answers and naturally repeated answers. In order to explore the existence of these suspected collusive answers in the practical answer set, we analyzed the repeated answers in original real dataset *adult2*. Consequently, we found 21 groups of repeated answers on the statistical result. Based on our detection result, we identified 11 groups of repeated answers as suspected collusive answers since the accuracy of result inference is obviously improved after filtering as shown in Fig.8. The main reason is that the side-effects of collusion on result inference are relieved. In addition, since Findcolluders is only effective for opinion-based rating tasks, and lots of normal answers are deleted based on this method, we found that the result quality after filtering based on Findcolluders is the lowest.

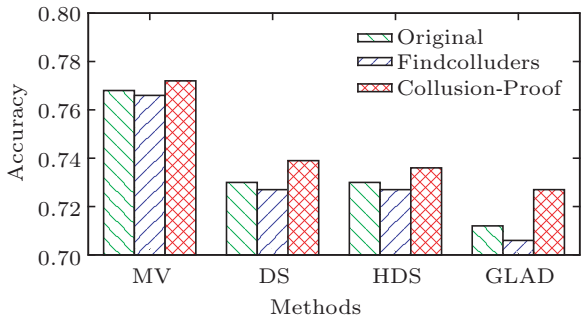


Fig.8. Accuracy with varying methods.

6.3.3 Effectiveness of Collusion Detection

In order to further evaluate the performance of Collusion-Proof on detecting collusion behaviors, we added some repeated answers generated by colluders into real-world datasets and implemented our detection method (i.e., Collusion-Proof) in comparison with the baseline method Findcolluders. We set the number of groups and the number of colluders in a group to 1 and 10 respectively.

As illustrated in Fig.9 and Fig.10, Collusion-Proof maintains a higher precision, accuracy, and recall than Findcolluders. As for Collusion-Proof, the precision of the dataset *ducks* is higher than that of dataset *adult2*. This is because the tasks completed by each worker of dataset *ducks* are more than those of dataset *adult2*, and the performance change rate can be estimated more accurately. As for Findcolluders, the precision of dataset *ducks* is lower than that of dataset *adult2*. The main reason is that the tasks of dataset *ducks* are easier to be performed, and the answers from dataset *ducks* are closer to the ground truth. In this case, workers exhibit high ability and normal repeated answers in dataset *ducks* are more than those in the dataset *adult*2. These normal answers are easily identified as collusive answers by Findcolluders based on similarity.

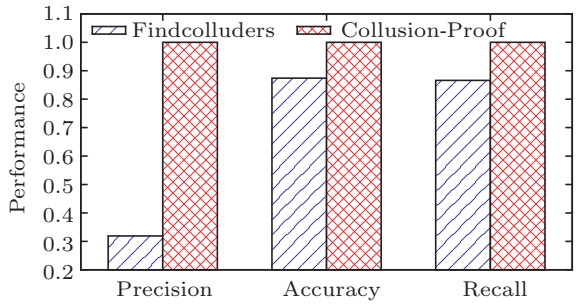


Fig.9. Performance with different detection methods in *ducks*.

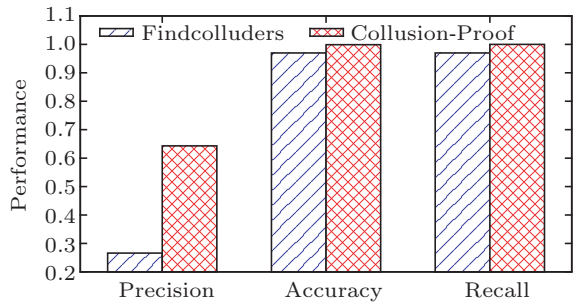


Fig.10. Performance with varying detection methods in *adult2*.

6.3.4 Effectiveness of Result Inference with Varying Collusion Proportion

On two real-world datasets, we utilized the accuracy of aggregated answers after result inference to evaluate the performance of C-MV, C-DS, C-HDS, and C-GLAD. There was one collusion group in the crowd and the collusion proportion changed from 0 to 1. We evaluate how collusion proportion affects the result quality.

As shown in Figs.11 and 12, Table 4, and Table 5, we can observe that the accuracy of all the methods decreases with the increasing collusion proportion. This is because the larger collusion proportion brings more repeated answers and makes the inferred result converge

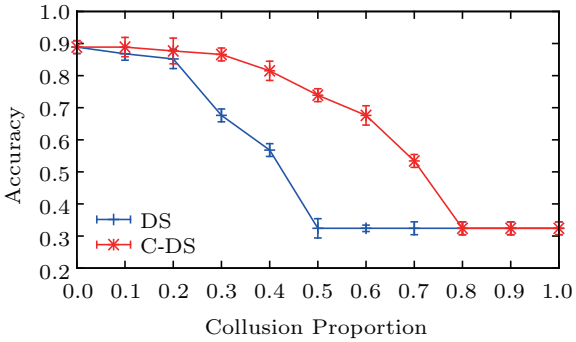


Fig.11. Accuracy with varying collusion proportion in *ducks* (DS, C-DS).

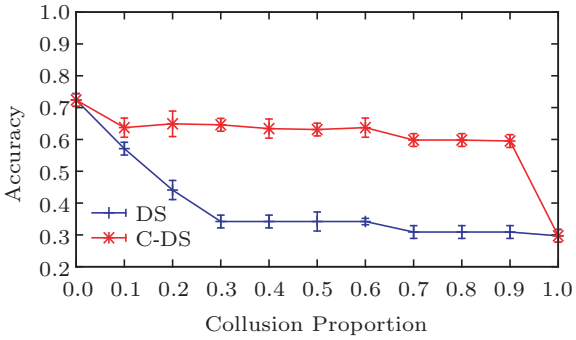


Fig.12. Accuracy with varying collusion proportion in *adult2* (DS, C-DS).

Table 4. Accuracy with Varying Collusion Proportion in *ducks* (the Others)

	MV	HDS	GLAD	C-MV	C-HDS	C-GLAD
0.0	0.759	0.583	0.685	0.757	0.583	0.685
0.1	0.759	0.522	0.583	0.755	0.562	0.583
0.2	0.722	0.422	0.556	0.750	0.553	0.565
0.3	0.639	0.359	0.513	0.742	0.543	0.554
0.4	0.556	0.324	0.324	0.732	0.556	0.539
0.5	0.324	0.324	0.324	0.759	0.434	0.437
0.6	0.324	0.324	0.324	0.722	0.432	0.431
0.7	0.324	0.324	0.324	0.324	0.324	0.422
0.8	0.324	0.324	0.324	0.324	0.324	0.410
0.9	0.324	0.324	0.324	0.324	0.324	0.324
1.0	0.324	0.324	0.324	0.324	0.324	0.324

Table 5. Accuracy with Varying Collusion Proportion in *adult2* (the Others)

	MV	HDS	GLAD	C-MV	C-HDS	C-GLAD
0.0	0.757	0.721	0.706	0.757	0.721	0.721
0.1	0.655	0.594	0.583	0.670	0.655	0.676
0.2	0.586	0.572	0.580	0.670	0.649	0.658
0.3	0.556	0.345	0.345	0.578	0.628	0.646
0.4	0.532	0.345	0.345	0.580	0.616	0.637
0.5	0.531	0.345	0.345	0.535	0.585	0.637
0.6	0.498	0.345	0.345	0.535	0.559	0.631
0.7	0.453	0.309	0.309	0.529	0.559	0.622
0.8	0.399	0.309	0.309	0.414	0.405	0.610
0.9	0.330	0.309	0.309	0.384	0.300	0.598
1.0	0.297	0.297	0.297	0.297	0.297	0.297

to answers generated by an individual worker, which lowers the quality of the inferred result. In particular, when the collusion proportion is 0, the performances of C-MV, C-DS, C-HDS, and C-GLAD are identical with those of their original versions without collusion detection mechanism. This is because the workers are independent and their answers contain no collusive repeated answers. When the collusion proportion is 1, we can observe that the curves of all the methods overlap with one another. This is because all workers are colluders and contained in a group. No matter whether or not the repeated answers are filtered out, the final result is generated by a single worker of average ability in multiple experiments. In most cases, we can also observe that no matter how the collusion proportion changes, C-MV, C-DS, C-HDS, and C-GLAD can achieve a higher accuracy than those without collusion detection mechanism.

6.3.5 *Effectiveness of Result Inference with Varying Number of Groups*

In this experiment, the number of collusive groups varied from 1 to 5, and each group had five members.

As shown in Figs.13 and 14, Table 6, and Table 7, the accuracy of all methods decreases with the increasing number of groups. The reason is that more groups will submit more repeated answers and these repeated answers greatly damage the quality of results. Still, we can also observe that C-MV, C-DS, C-HDS, and C-GLAD generate higher accuracy than their original versions without collusion detection. Among them, C-MV outperforms all the probabilistic methods (i.e., HDS, GLAD, C-HDS, and C-GLAD). This is because all the probabilistic inference methods entail learning parameters (e.g., the worker ability) according to the answers submitted by workers. However, in dataset *adult2*, the number of tasks completed by each worker is small, and the parameters cannot be accurately acquired.

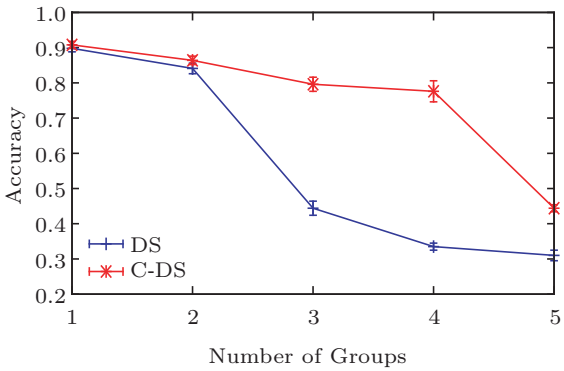


Fig.13. Accuracy with varying number of groups in *ducks* (DS, C-DS).

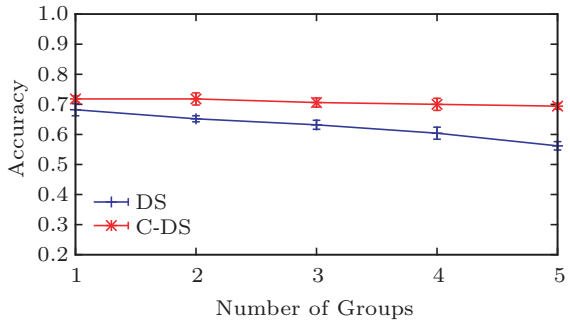


Fig.14. Accuracy with varying number of groups in *adult2* (DS, C-DS).

Table 6. Accuracy with Varying Number of Groups in *ducks* (the Others)

	MV	HDS	GLAD	C-MV	C-HDS	C-GLAD
1	0.712	0.583	0.556	0.719	0.583	0.583
2	0.611	0.583	0.556	0.704	0.582	0.583
3	0.481	0.556	0.556	0.667	0.576	0.556
4	0.426	0.556	0.556	0.593	0.572	0.565
5	0.370	0.519	0.463	0.500	0.565	0.556

Table 7. Accuracy with Varying Number of Groups in *adult2* (the Others)

	MV	HDS	GLAD	C-MV	C-HDS	C-GLAD
1	0.718	0.664	0.658	0.760	0.733	0.721
2	0.709	0.658	0.652	0.733	0.721	0.709
3	0.688	0.653	0.645	0.739	0.720	0.727
4	0.687	0.649	0.634	0.730	0.700	0.685
5	0.685	0.640	0.624	0.712	0.691	0.676

7 Discussion

In this section, regarding result inference in collaborative crowdsourcing, we discuss some research challenges and opportunities.

First, collaboration changes the performance of workers in groups during the task processing. For instance, a worker may turn to others for help via online forums, and then he/she can submit an answer of higher quality than the one completed independently. Related literature[29-30] suggested that the worker performance affects the result inference. In this case, a problem concerning how to estimate the worker performance in the case of collaboration to infer high-quality result rises.

Second, the accuracy of the inferred results can be improved by detecting and filtering out some repeated answers submitted by collusive workers. However, the optimization of detection algorithm for collaborative crowdsourcing is still an important problem, e.g., how to determine the threshold in Algorithm 1 is still an open problem. In this paper, a proper threshold is captured by using the empirical analysis which is a widely-adopted approach. Moreover, we plan to employ machine learning methods to capture the optimal threshold.

Third, it is difficult to capture the collaboration relationship among workers in hidden collaborative network on the real-world crowdsourcing platforms. Yin *et al.*[13] mapped the entire communication network of workers on a widely-used platform, Amazon Mechanical Turk by a self-reported method. While this work shows that the communication network exists, it is still an open problem to obtain the specific type of the collaboration among workers such as group plagiarism and sybils. It is of paramount importance to continue this study and to develop new techniques for a better understanding of the collaboration among crowdsourcing workers.

8 Conclusions

Although crowdsourcing is originally expected to solicit independent contributions from unknown workers, the collaboration among workers is becoming a known fact even on public crowdsourcing platforms. As an unexpected type of collaboration, the collusion among workers will damage the quality of crowdsourcing results. This work studied the Collusion-Proof result inference problem concerning crowdsourcing applications where there can be potentially collaboration among workers. We first designed a method to detect the collusion behaviors, and then incorporated the collusion detection results into existing result inference methods. The experimental results confirmed the effectiveness of our approaches.

References

[1] Li G L, Wang J N, Zheng Y D, Franklin M J. Crowdsourced data management: A survey. *IEEE Trans. Knowledge and Data Engineering*, 2016, 28(9): 2296-2319.

[2] Chen L, Lee D, Milo T. Data-driven crowdsourcing: Management, mining, and applications. In *Proc. the 31st Int. Conf. Data Engineering*, April 2015, pp.1527-1529.

[3] Deng J, Dong W, Socher R *et al.* ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2009, pp.248-255.

[4] Liu X, Lu M Y, Ooi B C, Shen Y Y, Wu S, Zhang M H. CDAS: A crowdsourcing data analytics system. *Proceedings of the VLDB Endowment*, 2012, 5(10): 1040-1051.

[5] Fang Y L, Sun H L, Li G L, Zhang R C, Huai J P. Effective result inference for context-sensitive tasks in crowdsourcing. In *Proc. the 21st Int. Conf. Database Systems for Advanced Applications*, April 2016, pp.33-48.

[6] von Ahn L, Maurer B, McMillen C, Abraham D, Blum M. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 2008, 321(5895): 1465-1468.

[7] Fang Y L, Sun H L, Zhang R C, Huai J P, Mao Y Y. A model for aggregating contributions of synergistic crowdsourcing workflows. In *Proc. the 28th AAAI Conf. Artificial Intelligence*, July 2014, pp.3102-3103.

[8] Zaidan O F, Callison-Burch C. Crowdsourcing translation: Professional quality from non-professionals. In *Proc. the 49th Annual Meeting of the Association for Computational Linguistics*, June 2011, pp.1220-1229.

[9] Bernstein M S, Little G, Miller R C, Hartmann B, Ackerman M S, Karger D R, Crowell D, Panovich K. Soylent: A word processor with a crowd inside. *Communications of the ACM*, 2015, 58(8): 85-94.

[10] Zhu Y S, Yue S C, Yu C, Shi Y C. CEPT: Collaborative editing tool for non-native authors. In *Proc. ACM Conf. Computer Supported Cooperative Work and Social Computing*, February 25-March 1, 2017, pp.273-285.

[11] Nebeling M, To A, Guo A H, De Freitas A A, Teevan J, Dow S P, Bigham J P. WearWrite: Crowd-assisted writing from smartwatches. In *Proc. CHI Conf. Human Factors in Computing Systems*, May 2016, pp.3834-3846.

[12] Gray M L, Suri S, Ali S S, Kulkarni D. The crowd is a collaborative network. In *Proc. the 19th ACM Conf. Computer-Supported Cooperative Work & Social Computing*, February 27-March 2, 2016, pp.134-147.

[13] Yin M, Gray M L, Suri S, Vaughan J W. The communication network within the crowd. In *Proc. the 25th Int. Conf. World Wide Web*, April 2016, pp.1293-1303.

[14] Salehi N, McCabe A, Valentine M, Bernstein M. Huddler: Convening stable and familiar crowd teams despite unpredictable availability. In *Proc. ACM Conf. Computer Supported Cooperative Work and Social Computing*, February 25-March 1, 2017, pp.1700-1713.

[15] Gadiraju U, Kawase R, Dietze S, Demartini G. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proc. the 33rd Annual ACM Conf. Human Factors in Computing Systems*, April 2015, pp.1631-1640.

[16] Sodré I, Brasileiro F. An analysis of the use of qualifications on the Amazon mechanical Turk online labor market. *Computer Supported Cooperative Work*, 2017, 26(4/5/6): 837-872.

[17] Chang J C, Amershi S, Kamar E. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proc. CHI Conf. Human Factors in Computing Systems*, May 2017, pp.2334-2346.

[18] Wang G, Wilson C, Zhao X H, Zhu Y B, Mohanlal M, Zheng H T, Zhao B Y. Serf and turf: Crowdturfing for fun and profit. In *Proc. the 21st Int. Conf. World Wide Web*, April 2012, pp.679-688.

[19] Adams S A. Maintaining the collision of accounts: Crowdsourcing sites in health care as brokers in the co-production of pharmaceutical knowledge. *Information Communication & Society*, 2014, 17(6): 657-669.

[20] Douceur J R. The Sybil attack. In *Proc. the 1st Int. Workshop on Peer-to-Peer Systems*, March 2002, pp.251-260.

[21] Lev O, Polukarov M, Bachrach Y, Rosenschein J S. Mergers and collusion in all-pay auctions and crowdsourcing contests. In *Proc. Int. Conf. Autonomous Agents and Multi-Agent Systems*, May 2013, pp.675-682.

[22] KhudaBukhsh A R, Carbonell J G, Jansen P J. Detecting non-adversarial collusion in crowdsourcing. In *Proc. the 2nd AAAI Conf. Human Computation and Crowdsourcing*, November 2014, pp.104-111.

[23] Xiang Q K, Nevat I, Zhang P F, Zhang J. Collusion-resistant spatial phenomena crowdsourcing via mixture of Gaussian processes regression. In *Proc. the 18th Int. Conf. Trust in Agent Societies*, May 2016, pp.30-41.

[24] Fang Y L, Chen P P, Sun K, Sun H L. A decision tree based quality control framework for multi-phase tasks in crowdsourcing. In *Proc. the 12th Chinese Conf. Computer Supported Cooperative Work and Social Computing*, September 2017, pp.10-17.

[25] Fang Y L, Sun H L, Chen P P, Deng T. Improving the quality of crowdsourced image labeling via label similarity. *Journal of Computer Science and Technology*, 2017, 32(5): 877-889.

[26] Sheng V S, Provost F, Ipeirotis P G. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proc. the 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, August 2008, pp.614-622.

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

[28] Dawid A P, Skene A M. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society*, 1979, 28(1): 20-28.

[29] Raykar V C, Yu S P, Zhao L H, Valadez G H, Florin C, Bogoni L, Moy L. Learning from crowds. *Journal of Machine Learning Research*, 2010, 11: 1297-1322.

[30] Gao C, Lu Y, Zhou D Y. Exact exponent in optimal rates for crowdsourcing. In *Proc. the 33rd Int. Conf. Machine Learning*, June 2016, pp.603-611.

[31] Whitehill J, Ruvolo P, Wu T, Bergsma J, Movellan J. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. the 22nd Int. Conf. Neural Information Processing Systems*, December 2009, pp.2035-2043.

[32] Garcia-Molina H, Joglekar M, Marcus A, Parameswaran A, Verroios V. Challenges in data crowdsourcing. *IEEE Trans Knowledge and Data Engineering*, 2016, 28(4): 901-911.

[33] Shin H, Park T, Kang S, Lee B, Song J, Chon Y, Cha H. CoSMiC: Designing a mobile crowd-sourced collaborative application to find a missing child in situ. In *Proc. the 16th Int. Conf. Human-Computer Interaction with Mobile Devices & Services*, September 2014, pp.389-398.

[34] Ambati V, Vogel S, Carbonell J. Collaborative workflow for crowdsourcing translation. In *Proc. ACM Conf. Computer Supported Cooperative Work*, February 2012, pp.1191-1194.

[35] Teevan J, Iqbal S T, Von Veh C. Supporting collaborative writing with microtasks. In *Proc. CHI Conf. Human Factors in Computing Systems*, May 2016, pp.2657-2668.

[36] Rahman H, Roy S B, Thirumuruganathan S, Amer-Yahia S, Das G. Task assignment optimization in collaborative crowdsourcing. In *Proc. IEEE Int. Conf. Data Mining*, November 2015, pp.949-954.

[37] Torshiz M N, Amintoosi H. Collusion-resistant worker selection in social crowdsensing systems. *Journal of Computer and Knowledge Engineering*, 2017, 1(1): 9-20.

[38] Celis L E, Reddy S P, Singh I P, Vaya S. Assignment techniques for crowdsourcing sensitive tasks. In *Proc. the 19th ACM Conf. Computer-Supported Cooperative Work & Social Computing*, February 27-March 2, 2016, pp.836-847.

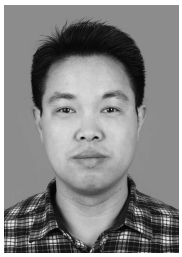
[39] Wang L, Zhou Z H. Cost-saving effect of crowdsourcing learning. In *Proc. the 25th Int. Joint Conf. Artificial Intelligence*, July 2016, pp.2111-2117.

[40] Welinder P, Branson S, Belongie S, Perona P. The multidimensional wisdom of crowds. In *Proc. the 23rd Int. Conf. Neural Information Processing Systems*, December 2010, pp.2424-2432.

[41] Ipeirotis P G, Provost F, Wang J. Quality management on Amazon Mechanical Turk. In *Proc. ACM SIGKDD Workshop on Human Computation*, July 2010, pp.64-67.

Peng-Peng Chen is a Ph.D. student in the School of Computer Science and Engineering, Beihang University, Beijing. His research interests mainly include crowd computing/crowdsourcing, and social computing. He is a student member of CCF and ACM.

Hai-Long Sun received his B.S. degree in computer science from Beijing Jiaotong University, Beijing, in 2001. He received his Ph.D. degree in computer software and theory from Beihang University, Beijing, in 2008. He is an associate professor in the School of Computer Science and Engineering, Beihang University, Beijing. His research interests include crowdsourcing, software analytics and distributed systems. He is a member of CCF, ACM, and IEEE.



Yi-Li Fang is a Ph.D. student in the School of Computer Science and Engineering, Beihang University, Beijing. His research interests mainly include crowd computing/crowdsourcing, social computing and decision science. He is a member of CCF and ACM.

Jin-Peng Huai received his Ph.D. degree in computer science from Beihang University, Beijing, in 1993. He is a professor in the School of Computer Science and Engineering, Beihang University, Beijing. He is an academician of the Chinese Academy of Sciences and the vice honorary chairman of China Computer Federation (CCF). His research interests include big data computing, distributed systems, virtual computing, service-oriented computing, trustworthiness, and security.