A Blockchain-Based Crowdsourcing System with QoS Guarantee via a Proof-of-Strategy Consensus Protocol

Xusheng Cai[1], Yue Wang[2], Feilong Lin[1], Changbing Tang[2(✉)], and Zhongyu Chen[1]

[1] School of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China
{caixusheng,bruce_lin,czy}@zjnu.edu.cn

[2] College of Physics and Electronics Information Engineering, Zhejiang Normal University, Jinhua 321004, China
{wangyue,tangcb}@zjnu.edu.cn

Abstract. Crowdsourcing technology has been widely used in the online matching system, e.g., Uber and Airbnb, which provides an efficient matching service and enables a balance between service supply and demand. The current crowdsourcing platform generally leverages a centralized architecture through a third party for trust endorsement. However, this kind of architecture brings several challenges such as the quality of service (QoS) and trustability. In this paper, a novel blockchain-based crowdsourcing system (BCS) is proposed to guarantee QoS and fight with the malicious behaviors which employs a new consensus protocol *Proof-of-Strategy* to solve the well-known fork issue in the blockchain. Proof-of-Strategy also enables a fully distributed implementation of a crowdsourcing platform which prevents the damage of the information during the matching service. By this tamper-proof design, the task of matching supply and demand can prevent malicious behaviors, e.g., plagiarism and fraud. Moreover, a *Quality Rating Protocol* (QRP) is introduced to jointly work with *Proof-of-Strategy* for the guaranteed service quality. The existence of Nash equilibrium regards to BCS is given by the game theoretical analysis. The performance evaluation is presented to illustrate the effectiveness of our proposed method.

Keywords: Blockchain · Crowdsourcing · Quality of service · Consensus protocol · Incentive

1 Introduction

With an increasing number of mobile devices, crowdsourcing platform enjoys a great popularity from the online matching systems. Its inherent parallel computational power plays a significant role in cutting down the time in the task of the matching matter between publishment and acceptance [3]. Unlike the

Z. Zheng et al. (Eds.): BlockSys 2020, CCIS 1267, pp. 72–86, 2020.
https://doi.org/10.1007/978-981-15-9213-3_6

traditional offline service market, crowdsourcing platform can offer an efficient matching service in large scale with a guaranteed service quality [1,5,14]. A typical crowdsourcing system consists of three different entities: the requester, the worker and the central coordinator [13]. A requester issues a task to the central coordinator. Once the central coordinator of crowdsourcing platform receives the task, it will publish this task and recruit workers who response to the task request for completing the task and returning the result to the crowdsourcing platform. In general, the crowdsourcing platform is responsible for the worker recruitment, the task allocation, the result collection, the data processing, and the evaluation on the trust of all workers. It returns the final report of the task execution to the requester.

However, the crowdsourcing confronts a series of challenges relevant to Quality of Service (QoS) and trustability [2,6]. The traditional crowdsourcing system generally leverages a centralized architecture which relies on a trustworthy service provider in the coordination centre. This architecture leads to two major disadvantages. First, it is easy for the crowdsourcing system to suffer from a single point of failure and the privacy leakage due to the matching service provided by the central platform. Second, workers may return a poor-quality result with little effort as the response to the task request. But these workers still claim that they have completed the task with a noticeable effort to ask for higher benefits. This kind of behaviour is called "free riding". The requester may claim a lower benefit to those workers who complete the task normally, regardless how well the workers have done. This kind of behaviour is called "false reporting". Because of these two typical cheating behaviours, it is difficult to guarantee QoS of crowdsourcing system.

The existing researches can partially address these challenges in crowdsourcing system. Various incentive mechanisms are developed to ensure QoS in the crowdsourcing system [10,15]. These incentive mechanisms can prevent from the cheating behaviours for both requester and the workers. Firstly, the requester will not cheat to falsely report the quality of the task result. Thus, no matter the workers complete the task honestly or not, the requester needs to pay the workers amounting at its true value. Then, the workers are motivated to make efforts spontaneously and complete the task with a high quality. This is because in the incentive mechanism, the workers without any noticeable effort will not obtain a profitable income. A batch allocation for tasks with overlapping skill requirements in crowdsourcing system is proposed in [7] where a better performance on the total payment of requesters, the average income of workers and the maintenance of the close successful probability of the task completion can be achieved. Moreover, there are existing researches that solves the security issues caused by the centralization. Encryption and differential privacy are used to protect data privacy [11]. New blockchain based crowdsourcing systems are introduced in [8]. In these systems, it is no longer a centralized system to provide services which can prevent the single point of failure and privacy disclosure risks.

Nevertheless, these incentive mechanisms require a credible central platform while these crowdsourcing systems cannot guarantee QoS effectively. To guarantee

QoS and solve the security issues simultaneously, it is essential to develop a new framework of incentive mechanism for solving these challenges in crowdsourcing system. In this paper, a blockchain based framework is introduced for a distributed crowdsourcing system where a Proof-of-Strategy consensus protocol is developed to address the fork issue of the blockchain. The *Quality Rating Protocol* (QRP) is proposed to jointly work with the Proof-of-Strategy consensus protocol for guaranteeing QoS. The main contributions of this paper are threefold:

- A novel blockchain crowdsourcing system (BCS) is proposed which can guarantee QoS and trustability in a distributed manner. Moreover, a new consensus protocol, namely Proof-of-Strategy, is designed for new block confirmation which ensures that only one unique block will be determined in each block period, and the security is further improved.
- Under the assumption of no centralized trusted third party existing in the system, a QRP is introduced to jointly coordinate with the Proof-of-Strategy protocol for motivating both the requester and the workers honestly making efforts to the task. This finally results in a high quality of matching service. Furthermore, the existence of Nash equilibrium regards to this BCS is verified by two game theoretical analyses.
- QoS of BCS is estimated through extensive performance evaluation. The numerical results show that the task results given by the workers in different scales are all of high quality. Moreover, the effect of important parameters, on the quality of the matching service system is given while the guideline on the design of system parameters are presented.

The rest of the paper is organized as follows. Section 2 introduces the BCS and analyzes the process of the completion of a crowdsourcing task briefly, as well as, details discussion of the modeling process. Section 3 analyses the worker's utility and verifier's utility by game theory to verify the existence of Nash equilibrium in BCS. In Sect. 4, the performance of the model is analyzed by simulations and experiments. Finally, Sect. 5 summarizes this paper.

2 Blockchain-Based Crowdsourcing Model

This section mainly explains the specific procedure of how BCS works in each stage with the key elements. The main symbols used in this paper have been listed, as shown in Table 1.

2.1 System Model

In BCS, two currencies, S-coin for service reward and R-coin for reputation evaluation, are designed. As a currency, S-coin is only issued as a reward when new blocks are generated by the system. Of course, real currency can be exchanged between agents by S-coin. As a value currency generated by reputation, R-coin is rewarded or deducted by the system according to the quality after the agent

Table 1. The notations of explanation

Notation	Explanation
r_i	The amount of R-coin owned by agent i
s_i	The amount of S-coin owned by agent i
r_i'	The amount of R-coin declared by the agent i when competing with the verifier
s_i'	The amount of S-coin obtained by the agent i when competing for verifier
a	The verifier's S-coin reward
b	The sum of S-coin rewards for participating in the verifier's competition
c	Cost per unit of completion effort
t_i	The time when the agent i completes the task
p_i	The percentage of tasks completed by the agent i
D	Task difficulty
T	Task working time interval, $T = T_{deadline} - T_{publish}$
h	The social threshold
$F(D)$	Task cost paid by S-coin
$k, \eta, \mu, \alpha, \beta, \lambda$	The constants given by the system designer

completes the task. In BCS, there are three roles: 1) the requester, 2) the worker, 3) the verifier. Each agent can become a requester or a worker according to the actual situation. In addition, the Proof-of-Strategy consensus protocol is proposed to select the verifier. Every agent can participate the verifier competition. In the process of competition, the agent will consume a number of R-coin and get a number of S-coin. If the agent becomes a requester, it needs to pay the workers who accept the task with a number of S-coins to publish the task. If the agent becomes worker, it can get a number of S-coins for completing the task, and the system will reward or deduct a number of R-coins according to the task completion quality.

As shown in Fig. 1, the agent that wants to publish the task becomes the requester and the task information is broadcast to all agents. After receiving the task, the agent that wants to make a profit from completing the task becomes

the worker. The workers need to encrypt the results and broadcast them to the verifier when they complete the task. When the deadline comes, the verifier first checks whether the worker's R-coin balances, who has been sent the results, is higher than the social threshold. If so, it decrypts and reads the information of the time that the worker completed the task. After every worker has been checked, the verifier calculates the S-coin reward that the requester needs to pay through the function $F(x)$ and sends the payment requests to the requester. After the remunerations are paid by the requester and verified by the verifier, he will send results to the requester. The requester will receive the task result and generates the worker's R-coin reward and punishment results according to the Quality Rating Protocol (QRP). Then, the requester broadcasts it. Finally, the verifier will generate the new block, including the requester's payment to the worker, the R-coin information of the system rewards and punishments, and the task results, broadcasting to every agent.

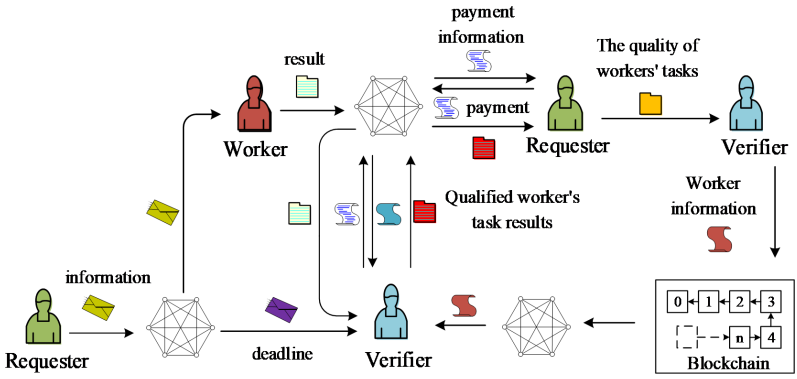


Fig. 1. Blockchain-based crowdsourcing system

2.2 Proof-of-Strategy Consensus Protocol

There are many problems in the traditional consensus protocol. For instance, a big waste of social resources is ubiquitousness in the consensus process of the *proof-of-work* (PoW) in regard to bitcoin. The fork problem of *proof-of-stake* (PoS) is not unusual. In order to solve this problem, a Proof-of-Strategy consensus protocol will be proposed in BCS. Each agent can choose a different strategy according to its own situation, which corresponds to a different success rate of becoming the verifier. The Proof-of-Strategy consensus is low-cost and can logically solve the fork problem, where a bribe attacks will be extremely expensive. At the same time, the "verifier" generates the next block can only be created by a selection policy that is competitive. Hence, it is hard to be predicted. The specific procedures of the proof-of-strategy consensus protocol are as follows.

First, the agent who wants to be a "verifier" is called the "competitor". It declares a certain amount of R-coin as its own strategy according to its own situation, and its declaration amount should not exceed its R-coin balance. After the declaration time is over, the system allocates S-coin according to the proportion of the declared R-coin in all declared R-coin, and deducts the declared R-coin, following the formula below:

$$s'_i = b \cdot \frac{r'_i}{\Sigma_{i=1}^{n} r'_i}, \tag{1}$$

where s'_i means the amount of S-coin obtained by the agent i when competing for the verifier. r'_i means the amount of R-coin declared by the agent i when competing with the verifier. b is a constant in the system, represents the total number of S-coins issued by the system during each competition to become the verifier. n is the total number of competitors.

Then, each newly allocated S-coin, with a minimum unit of "*satoshi*", has a continuous binary number of l_s in length. Every *satoshi* S-coin that a competitor gets can provide the system with a "bit" information, so the system will get mapping relationships between the new S-coin number and the value $\{0, 1\}$. Then it arranges the mapped $\{0, 1\}$ from small to large according to its corresponding s-coin number, and will obtain a string of binary numbers with the length of l_t. l_t is the newly allocated total amount of S-coins with the smallest unit "*satoshi*".

Finally, this binary number of length l_t is hashed to a binary number of the length l_s, and the competitor who has the same number S-coin becomes the verifier of this phase. The system awards an additional S-coin. It is worth noting that the proof-of-strategy consensus protocol splits the s-coin reward into two parts, i.e., $a + b$, so that agents that do not become the verifier will also have benefits. It can promote agents to actively participate in the verifier competition. In the competition, the "verifier" selection is based on the strategy of each competitor and is therefore unpredictable.

2.3 The Cost of Task

Generally, the type of tasks is not of single class, which may consist of a large number of classes. Different kinds of tasks have not only different complexities, but also the same kind of tasks with a different complexity. Considering that the complexity of a task is related to the time and the quantity of completion, we define the task complexity as follows:

Definition 1 (Task Complexity, D). *The complexity of the task published by the requester is determined by the linear combination of the average time of the worker completing the task and the reciprocal of the number of workers completing the task within the specified time, satisfying the condition:*

$$D = \begin{cases} \eta \cdot \frac{1}{N} \Sigma_{i=1}^{N} (t_i - t_{publish}) + (1 - \eta) \cdot \frac{1}{N} & N \geq 1, \\ 0 & N = 0, \end{cases} \tag{2}$$

where D is a numerical mapping of the task complexity. As a system parameter, η is used to control the effect on the average completion time and the number of workers completed on the complexity of the task. Of course, it may be different for different types of tasks. N represents the number of workers completing the task within the specified time. T_i represents the time taken by the ith worker to complete the task. $t_{publish}$ denotes the publish time of the task. When the average time to complete the task is long or the number of people completing the task within the specified time is small, it is said that the task is more difficult. otherwise it is simple.

After the task complexity is determined, according to the rule that the higher complexity, the higher cost. The task S-coin cost function $F(D)$ is given, which follows the formula:

$$F(D) = \begin{cases} F_{base} + k \cdot D & D \neq 0, \\ 0 & D = 0, \end{cases} \tag{3}$$

where, $F(D)$ represents the cost of S-coins required for the task, F_{base} represents the basic cost, and k is the system parameter.

2.4 Quality Rating Protocol

After accepting the task, a worker may has the incentive to take the payment and provides no effort to solve the task, a behavior commonly known as "free riding". In order to avoid this phenomenon and encourage workers to work honestly, two indexes of "task complexity: D" and "task completion degree: p" are introduced, and a rating agreement is constructed. The quality of work is quantified through $D \cdot p$, so as to allocate corresponding rewards and penalties.

Definition 2 (Task Completion Degree, p). *It is assumed that the degree of task completion is quantifiable and the cost per unit is c. This degree is denoted by p, $p \in [0, 1]$.*

p is also used to quantify the quality of the worker's completed tasks. The larger the p, the better the worker's working attitude and the higher the task quality.

Definition 3 (Quality Rating Protocol, QRP). *A quality rating protocol QRP is represented as a quadruple $\{r, \sigma, \tau, \varphi\}$, which consists of four components: a set of rating labels r, a social strategy σ, a rating scheme τ, and a pricing scheme φ.*

- $r \in R$ denotes the set of rating labels, where r represents the R-coin balance of each individual. If the r is too low, you will not respond to the task.
- $\sigma : r \rightarrow p$ is the social strategy adopted by each worker, where p represents the task completion degree of worker.

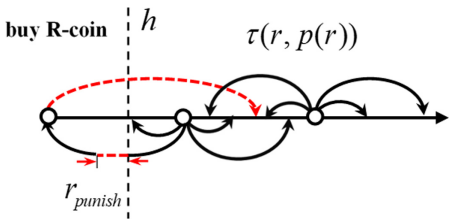


Fig. 2. Schematic representation of a rating protocol.

– $\tau : r \times p \rightarrow r$ specifies how a worker's R-coin balance should be updated based on its strategies. In this rating scheme, the higher the degree of completion of tasks, the more rewards to the worker, as follows:

$$\tau(r,p) = \begin{cases} r + \mu D(p-0.6) & r \geq h \text{ and} \\ & r + \mu D(p-0.6) \geq h, \\ r + \mu D(p-0.6) - r_{punish} & r \geq h \text{ and} \\ & r + \mu D(p-0.6) < h, \\ r & r < h \end{cases} \tag{4}$$

where $h > 0$ is the selected social threshold. When the r is less than h, the worker cannot accept the task. It is worth noting that when the R-coins balance of the worker changes from not less than h before the task to less than h after the task, the system will deduct its r_{punish} R-coins as the penalty. Therefore, the higher the R-coins balance, the more secure it is, and the higher the status it will be. μ is the system parameter, the designer can adjust μ to control the growth rate of R-coin in the whole system. A schematic representation of a proposed rating protocol is provided in Fig. 2.

– $\varphi : t \rightarrow s$ defines the rules that the designer uses to reward/punish workers in order to incentivize their service provision. In this pricing scheme, workers will get more rewards for completing tasks in a shorter time. Therefore, workers are encouraged to complete tasks quickly. Specific pricing scheme as follows:

$$\varphi(t_i) = S_{base} + [F(D) - N \cdot S_{base}] \cdot \frac{T_{deadline} - t_i + T_{base}}{T_{all}}, \tag{5}$$

where $S_{base} = \alpha \cdot F(D) \cdot \frac{1}{N}$, $T_{all} = \sum_{i=1}^{N}(T_{deadline} - t_i + T_{base})$. S_{base} represents basic rewards, $T_{deadline}$ represents task deadline. Parameter α is used to control the degree of incentive for the task completion time. The smaller α is, the stronger the desire of workers to complete the task in a short time will be.

3 Utility Analysis

This section mainly analyses the workers' utility by game theory to verify the existence of Nash equilibrium in BCS.

Considering that every agent can't get task when its R-coin is less than h, so it needs to purchase tasks from the system with S-coin. Supposed that all agents are rational and selfishness, then every agent will keep its R-coin from being less than h as much as possible. According to the principle of Economics, the value of R-coin to every agent should meet the law of Diminishing marginal utility, which means that the value of the unit of R-coin to the agent which has less R-coin is higher, and the rate of value decline is greater. Supposed that the value of a R-coin to every agent satisfies the utility function $Q(\cdot), Q(r_i) > 0, Q'(r_i) < 0, Q''(r_i) > 0, r_i \in (h, \infty)$.

3.1 Verifier's R-Coin Declaration Game

Considering a competition with a group of competitors to become the verifier. The agents denoted by $J = \{1, 2, ...\}$. The ith competitor declares a certain amount of R-coin, denoted by $r'_i, r'_i \leq r_i$. In the competition, the utility function of competitor i consists of 1) the expected revenue obtained from being the verifier, 2) S-coin revenue obtained from declaring R-coin, 3) the utility of deducting R-coin, which is formulated as follows:

$$w_i = a \cdot \frac{r'_i}{\Sigma_{i \in J} r'_i} + b \cdot \frac{r'_i}{\Sigma_{i \in J} r'_i} - Q(r_i) \cdot r'_i, \tag{6}$$

We employ the backward induction method to analyze the Stackelberg equilibrium of the utility function [9]. Given the utility function $Q(\cdot)$, the competitors compete to maximize their individual utilities by choosing their a certain amount of R-coin, which forms a noncooperative Verifier R-coin declaration Game (VRG) $G^v = \{J, R', \{w_i\}_{i \in J}\}$, where J is the set of competitors, R' is the strategy set of competitors, and w_i is the utility function of competitor i.

Definition 4. *A set of strategy profiles $R'^{ne} = \{r'^{ne}_1, ..., r'^{ne}_{|J|}\}$ is the Nash equilibrium of the VRG $G^v = \{J, R', \{w_i\}_{i \in J}\}$, if, for $\forall i \in J, w_i(r'^{ne}_i, R'^{ne}_{-i}, Q(\cdot)) \geq w_i(r'_i, R'^{ne}_{-i}, Q(\cdot))$, for $r'_i \geq 0$. Where R'^{ne}_{-i} represents the Nash equilibrium set excluding r'_i .*

Theorem 1. *A Nash Equilibrium exists in VRG $G^v = \{J, R', \{w_i\}_{i \in J}\}$.*

Proof. By differentiating w_i defined in Eq. (6) with respect to r'_i, we have $\frac{\partial w_i}{\partial r'_i} = \frac{(a+b)\sum_{j \in J-i} r'_j}{(\sum_{j \in J} r'_j)^2} - Q(r_j)$, and $\frac{\partial^2 w_i}{\partial r'^2_i} < 0$. Where J_{-i} represents a group of miners excluding i. Noted that w_i is a strictly concave function with respect to r'_i. Therefore, given any $Q(r_i) > 0$ and any strategy profile R'^{ne}_{-i} of the other competitors, the best response strategy of competitor i is unique when $r'_i \geq 0$. Accordingly, the Nash equilibrium exists in the noncooperative VRG G^v.

3.2 Worker's Attitude Game

Suppose that the working ability of the ith worker is v_i, which represents an inherent attribute of worker i. Considering a task with a group of workers

(denoted by $J' = \{1, 2, ...\}$), the ith worker chooses an attitude (denoted by $p_i, p_i \in [0, 1]$) to complete the task. Therefore, the cost for the worker i to complete the task is $p_i \cdot c$, and the time interval for the worker i to complete the task is p_i/v_i. In this task, the utility function of the worker i consists of 1) basic remuneration of S-coin, 2) S-coin revenue obtained from the requester, 3) the utility of rewarding or deducting R-coin from system, 4) the cost of complete the task, which is formulated as follows:

$$U_i = S_{base} + A \cdot \frac{\triangle t - \frac{p_i}{v_i}}{\Sigma_{j \in J'} \triangle t - \frac{p_j}{v_j}} + Q(r_i) \cdot \triangle \tau(r_i, p_i) - p_i \cdot c, \tag{7}$$

where $A = [F(D) - |J'| \cdot S_{base}]$, $\triangle t = t_{deadline} - t_{publish}$. We employ the backward induction method to analyze the Stackelberg equilibrium of the utility function. Given the utility function $Q(\cdot)$, the workers compete to maximize their individual utilities by choosing their attitude of completing the task, which forms a noncooperative Workers' attitude Game (WAG) $G^w = \{J', P, \{U_i\}_{i \in J'}\}$, where J' is the set of workers, P is the strategy set of workers, and U_i is the utility function of worker i.

Definition 5. *A set of strategy profiles $P^{ne} = \{p_1^{ne}, ..., p_{|J'|}^{ne}\}$ is the Nash equilibrium of the WAG $G^w = \{J', P, \{U_i\}_{i \in J'}\}$, if, for $\forall i \in J', U_i(p_i^{ne}, P_{-i}^{ne}, Q(\cdot)) \geq U_i(p_i, P_{-i}^{ne}, Q(\cdot))$, for $p_i \geq 0$. Where P_{-i}^{ne} represents the Nash equilibrium set excluding p_i*

Theorem 2. *A Nash Equilibrium exists in WAG $G^w = \{J', P, \{U_i\}_{i \in J'}\}$.*

Proof. By differentiating U_i defined in Eq. (7) with respect to p_i, we have $\frac{\partial U_i}{\partial p_i} = -\frac{A}{v_i} \cdot \frac{\sum_{j \in J' - i} \triangle t - \frac{p_j}{v_j}}{(\sum_{j \in J'} \triangle t - \frac{p_j}{v_j})^2} - \mu D Q(r_i) - c$, and $\frac{\partial^2 U_i}{\partial p_i^2} < 0$. Where J'_{-i} represents a group of miners excluding i. Noted that U_i is a strictly concave function with respect to p_i. Therefore, given any $Q(r_i) > 0$ and any strategy profile P_{-i}^{ne} of the other competitors, the best response strategy of competitor i is unique when $p_i \geq 0$. Accordingly, the Nash equilibrium exists in the noncooperative WAG G^w.

Through the above analysis, the Nash Equilibrium exists in both games. Therefore, the system designer can adjust the parameters so that only when all workers choose a high work attitude can they reach the Nash equilibrium. It is worth noting that in Eq. (7), if the worker's ability v_i for completing the task is at a low level, he will find out through calculation that his optimal strategy is not to participate in the task. In other words, even if he is completely conscientious about the task in this case, he is so limited by his ability cannot to provide high-quality results that he is unable to reap the benefits.

4 Performance Evaluation

In this part, to analyze the QoS given by the blockchain-based crowdsourcing model, we provide numerical results to illustrate its characteristics. We use DiDi

Data set for experimental settings where a ride-hailing business of an enterprise edition is extracted [4]. Specifically, in a modern company, there are a large number of customers that require a ride-hailing service when working after hours every day. Thus, completing the ride-hailing service for a specific company is a task. DiDi platform is the requester while the car driver who accepts the request is the worker. The working ability v_i of each worker is considered an inherent trait that will not change in the short term, and measured by the overall driver-partner rating. The social threshold is set to $h = 30$, basic pay is set to $F_{base} = 10$, and the function $Q(\cdot)$ is set to $Q(r_i) = 100/(r_i - h)$. Under these conditions, by dynamic update system parameters a, b and α in different scales, which guarantee that the optimal policy $r_i'^{ne}$ in VRG and p_i^{ne} in WAG of all agents satisfy $r_i'^{ne} \epsilon (0, h), p_i^{ne} \epsilon (0, 1)$. In the experiment, we chose a set of ride-hailing service data for a similar distance as the same kind of crowdsourcing task. Meanwhile, set $\Delta t = 30$ min. If the driver arrives at the destination in Δt, his task completion degree p is 100%. If not, $p = \Delta t \times v_i$. Under these Settings, we analyze the task quality of the system in the following aspects.

4.1 QoS Comparison Between BCS and General Incentive Model

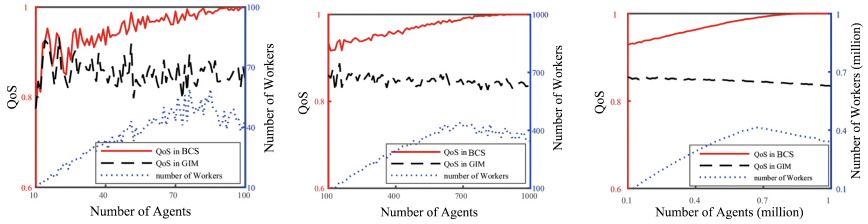
In order to facilitate the comparison, only used QRP to simulate the general incentive model (GIM) [12] without combining proof-of-strategy. In this case all workers will accept the task because they are profitable. Under the condition of keeping other parameters unchanged, agents with different working efficiencies are randomly selected for the system with total agents of 10–100, 100–1000 and 0.1–1 million respectively. The average task quality under the corresponding scale in BCS and general model is simulated, as shown in Fig. 3. The abscissa represents the total number of agents N, and the ordinate represents the QoS, which is the average task quality of the BCS and general model, and the number of workers participating in the task.

Obviously, compared with general incentive model, the QoS in BCS stay higher level, because workers who are not competitive in BCS will voluntarily give up accepting tasks. In the same scale, the more agents, the higher the QoS in BCS. In different scales, the larger the scale, the smaller the fluctuation of the QoS in BCS and the much stabler the system. Interestingly, under the same scale, the number of workers participating in tasks increases with the number of agents, but it tends to be flat after reaching at a certain large value. This shows that when the number of agents reaches a certain value, there will be a competition amongst agents. Some workers who do not have a competitiveness (the quality of completing tasks cannot reach a high level) will not accept tasks spontaneously.

4.2 The Effect of Parameters on QoS

1) **The Effect of T on QoS**

As a linear mapping of the task difficulty, the larger Δt, the more drivers



(a) 10-100 (b) 100-1000 (c) 0.1-1(Million)

Fig. 3. Average task quality of total agents under (a) 10–100 (b) 100–1000 (c) 100,000–1,000,000

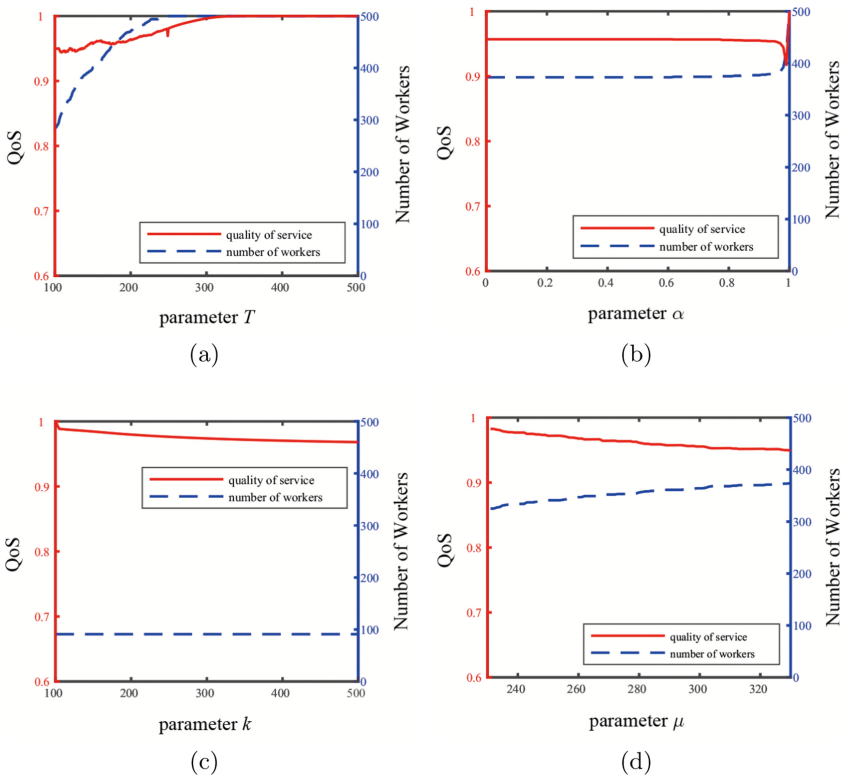


Fig. 4. The effect on (a) T, (b) α, (c) k and (d) μ on average task quality at $N = 500$

can complete the task with $p = 1$; otherwise, the smaller Δt, the less driver can complete task in Δt. In the total number of agents $N = 500$, for a task working time interval Δt, the effect on QoS is given in Fig. 4(a). The abscissa represents the task working time interval Δt, and the ordinate represents the QoS in the BCS and the number of workers participating in the task. It can be seen from the figure that the smaller Δt, the fewer agents can complete the task with a high quality, and the fewer drivers willing to accept the task; Conversely, the larger Δt, the more drivers can complete the task, and the more drivers will accept the task. With the growth of Δt, the quality of tasks is also improved.

2) **The Effect of α on QoS**
The parameter α affects the driver's base salary for completing the task. The driver will get more base salary if α increasing. In the total number of agents $N = 500$, for the importance parameter α of the task efficiency, the effect on the task quality is given in Fig. 4(b). The abscissa represents the importance parameter α of the task efficiency. It can be seen that when α is greater than 0.9, the average task quality will fluctuate slightly, and when α is less than 0.9, the average task quality will have little effect.

3) **The Effect of k on QoS**
In the total number of agents $N = 500$, for the payments range parameter k, the effect on task quality is given in Fig. 4(c). The abscissa represents the payments range parameter k. The larger the k, the more S-coins the driver gets under the same conditions, but the more S-coins the requester pays. And we can seen that with the increase of k, the task quality decreases slightly, but the decrease is not significant in the Fig. 4(c).

4) **The Effect of μ on QoS**
In the total number of agents $N = 500$, for the rewards range parameter μ, the effect on task quality is given in Fig. 4(d). The abscissa represents the rewards range parameter μ. It can be seen from the Fig. 4(d) that the larger μ, the more drivers will accept the task, and drivers can get the benefits with less effort; Conversely, the smaller μ, the fewer drivers willing to accept the task, but the more drivers can complete the task with a high quality. With the decreasing of the μ, the quality of tasks is also improved.

4.3 Guidance Suggestions for Design

Designer can adjust T, α, k and μ to balance the enthusiasm of the worker and the requester, and make the average task quality of the system maintain a high level. The smaller μ is, the more demanding it is to get the benefits by complete the tasks. The larger T, the more agents can complete the task, and the more agents can accept the task. The smaller α, the worker well use less time to complete the tasks, but the μ will be bigger. The larger k, the more S-coins the worker will get and this system will be able to attract more agents

so that the QoS of this system will be easier to maintain a high level. However, the requester need to pay more S-coin.

5 Conclusion

To address the problem that the QoS and the trustablility of the central platform in the traditional crowdsourcing cannot be guaranteed, we have proposed a blockchain-based crowdsourcing system (BCS). By introducing the blockchain technique into a crowdsourcing scenario, the model can achieves the properties of centerless, irrevocable and nonrepudiation during the process of crowdsourcing, thus avoiding the risk of attacks and frauds.

In addition, we have proposed the Proof of Strategy consensus protocol to solve the fork problem and the high cost problem of blockchain operation. In this consensus protocol, each agent chooses its own strategy to compete for its best interests, and reaches a consensus, determining the only verifier, according to the strategy chosen by all agents.

In BCS, we combine the Proof of Strategy consensus protocol and the quality rating protocol to guarantee the QoS, and verify the existence of BCS. The simulation results show that the task results given by workers in different scales are of high quality. At the same time, we also give some guidance suggestions for the parameter design of BCS.

Acknowledgments. This work was partly supported by the National Natural Science Foundation of China (No. 61877055), the Zhejiang Provincial Natural Science Foundation of China (Nos. LY18F030013), and the National innovation and entrepreneurship program for college students (No. 201910345015).

References

1. Baba, Y.: Statistical quality control for human computation and crowdsourcing. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 5667–5671. International Joint Conferences on Artificial Intelligence Organization (2018)
2. Checco, A., Bates, J., Demartini, G.: Quality control attack schemes in crowdsourcing. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 6136–6140. AAAI Press (2019)
3. Chittilappilly, A.I., Chen, L., Amer-Yahia, S.: A survey of general-purpose crowdsourcing techniques. IEEE Trans. Knowl. Data Eng. **28**(9), 2246–2266 (2016)
4. Dataset (2019). https://outreach.didichuxing.com/research/opendata/en/
5. Fang, Y., Sun, H., Chen, P., Huai, J.: On the cost complexity of crowdsourcing. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 1531–1537. International Joint Conferences on Artificial Intelligence Organization (2018)
6. Feng, W., Yan, Z., Zhang, H., Zeng, K., Xiao, Y., Hou, Y.T.: A survey on security, privacy, and trust in mobile crowdsourcing. IEEE Internet of Things J. **5**(4), 2971–2992 (2017)

7. Jiang, J., An, B., Jiang, Y., Shi, P., Bu, Z., Cao, J.: Batch allocation for tasks with overlapping skill requirements in crowdsourcing. IEEE Trans. Parallel Distrib. Syst. **30**(8), 1722–1737 (2019)
8. Li, M., et al.: CrowdBC: a blockchain-based decentralized framework for crowdsourcing. IEEE Trans. Parallel Distrib. Syst. **30**(6), 1251–1266 (2018)
9. Li, Z., Kang, J., Yu, R., Ye, D., Deng, Q., Zhang, Y.: Consortium blockchain for secure energy trading in industrial internet of things. IEEE Trans. Industr. Inf. **14**(8), 3690–3700 (2017)
10. Peng, D., Wu, F., Chen, G.: Pay as how well you do: a quality based incentive mechanism for crowdsensing. In: Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 177–186. ACM (2015)
11. To, H., Ghinita, G., Fan, L., Shahabi, C.: Differentially private location protection for worker datasets in spatial crowdsourcing. IEEE Trans. Mob. Comput. **16**(4), 934–949 (2016)
12. Wang, H., Guo, S., Cao, J., Guo, M.: Melody: A long-term dynamic quality-aware incentive mechanism for crowdsourcing. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 933–943 (2017)
13. Xia, J., Zhao, Y., Liu, G., Xu, J., Zhang, M., Zheng, K.: Profit-driven task assignment in spatial crowdsourcing. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 1914–1920. International Joint Conferences on Artificial Intelligence Organization (2019)
14. Xia, S., Lin, F., Chen, Z., Tang, C., Ma, Y., Yu, X.: A Bayesian game based vehicle-to-vehicle electricity trading scheme for blockchain-enabled Internet of vehicles. IEEE Trans. Veh. Technol. **69**, 6856–6868 (2020)
15. Zhang, Y., Van der Schaar, M.: Reputation-based incentive protocols in crowdsourcing applications. In: 2012 Proceedings IEEE INFOCOM, pp. 2140–2148. IEEE (2012)