Xu *et al. J Wireless Com Network* (2022) 2022:55
https://doi.org/10.1186/s13638-022-02135-1

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Blockchain-based multi-skill mobile crowdsourcing services

Weize Xu[1], Hongyue Duan[2], Xiao Chen[3*], Jie Huang[4*], Deyong Liu[2] and Yichao Chen[2]

*Correspondence:
chenxiao@zjinfo.gov.cn;
huangjie@zust.edu.cn

[1] Department of Cardiac Surgery, The Children's Hospital, Zhejiang University School of Medicine, National Clinical Research Center for Child Health, Hangzhou, China
[2] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
[3] Institute of Scientific and Technical Information of Zhejiang Province, Hangzhou, China
[4] School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou, China

Abstract

With the boom in 5G technology, mobile spatial crowdsourcing has shown great dynamism in industrial mobile communications and edge computing node management. But the traditional crowdsourcing system is not advanced enough to adapt to the new environment. Typically, traditional crowdsourcing workflow is hosted by a centralized crowdsourcing platform. However, the centralized crowdsourcing platform faces the following problems: (1) single point of failure, (2) user privacy leakage, (3) subjective arbitration, (4) additional service fee, and (5) non-transparent task assignment process. To improve those problems, we replaced the centralized crowdsourcing platform with a decentralized blockchain infrastructure. And we analyzed the challenge problems of multi-skilled spatial crowdsourcing tasks in the blockchain crowdsourcing system. In addition, a crowdsourcing task allocation algorithm has been proposed, which implements a transparent task distribution process and can adapt to the computing-constrained environment on the blockchain. Compared with the TSWCrowd blockchain-based crowdsourcing model, our system has a higher task allocation rate under the same conditions. And the experimental result shows our work has good economic feasibility, which decentralizes the crowdsourcing process and significantly reduces the additional consumption of the crowdsourcing process.

Keywords: Mobile crowdsourcing, Blockchain, Spatial communication, Smart contract

1 Introduction

5g, crowdsourcing, and blockchain are all hot words that have been many discussed in recent years. 5g represents the latest technology in wireless network communication, providing a high-speed, low latency technology for wireless data transmission. Crowdsourcing is a standard method of using the wisdom of crowds to solve problems, from fixing plumbing to T-shirt print design. Blockchain provides privacy, decentralization, and highly available infrastructure services. When crowdsourcing software meets blockchain, it will emerge privacy, decentralized, and high availability crowdsourcing platform. Unlike traditional crowdsourcing, the blockchain-based crowdsourcing platform eliminates the centralized intermediary platform of the traditional crowdsourcing model. What's more, it implements direct communication between the requester and the worker. The identities of all participants are reciprocal, and their behavior toward each other is regulated through the smart contract.

Chaer et al. [1] proposed a scheme: using crowdsourcing to solve the construction of 5g cellular towers, which allows individuals to build 5g cellular towers and share them through crowdsourcing, which will reduce the construction and promotion costs of 5g facilities and will promote the development of 5g. It also enables and settles the costs and benefits between the sharers and the consumers through a decentralized blockchain-based crowdsourcing platform and regulates the behavior between individual investors and operators through smart contracts.

At the same time, the development of 5g has brought new vitality to mobile crowdsourcing. Stable and high-speed wireless networks enable people to have higher accuracy and better stability in the information they can access through their mobile devices. With 5g technology, people can filter and locate crowdsourcing tasks more accurately and quickly, improving the efficiency of the crowdsourcing process.

A crowdsourcing system consists of three characters: requesters, workers, and a centralized crowdsourcing platform (Fig. 1). As the figure shows, the centralized crowdsourcing platform controls all the crowdsourcing processes. However, this kind of crowdsourcing system based on a centralized platform has many shortcomings. Disadvantages of the centralized crowdsourcing platform are listed in [2]: (1) single point of failure, which generally occurs in centralized systems, (2) user privacy leakage—user privacy information(e.g., name, email address, and phone number) is stored in the database of crowdsourcing platforms, which has the risk of privacy leakage and data loss, (3) subjective arbitration—when there is a disagreement between the requester and the worker, they will get help from the crowdsourcing platform to give a subjective arbitration, which may lead to "false-reporting" and "free-riding" [3], and (4) additional service fee—when a crowdsourcing task is completed, the crowdsourcing platform will require users to pay for services, which increases users' costs.

To solve the problems in traditional crowdsourcing, many researchers have introduced blockchain technology into the crowdsourcing process and replaced the centralized crowdsourcing platform with blockchain, a decentralized infrastructure.

The blockchain-based decentralized crowdsourcing system can completely solve the four problems mentioned above:

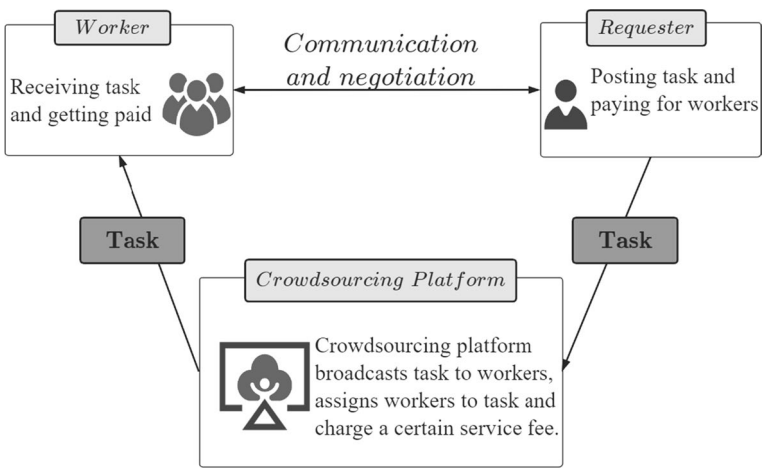

Fig. 1 Crowdsourcing system

1. *Single point of failure* Blockchain is jointly maintained by distributed nodes around the world, and it is only necessary to ensure that more than 50% of the nodes are working properly to guarantee the availability of the system. So the blockchain-based crowdsourcing system will be completely immune to the problem of single point of failure.
2. *User Privacy leakage* The blockchain system has native anonymity, which effectively protects users' privacy information.
3. *Subjective arbitration* Blockchain crowdsourcing system can regulate the behavior of crowdsourcing participants through smart contract technology and ensure the fairness of the crowdsourcing process through a predefined procedure. Rely on well-designed smart contracts, problems such as "false-reporting" and "free-riding" can be avoided.
4. *Additional service fee* The blockchain crowdsourcing system does not have a centralized intermediary platform to collect service fee, but still, needs to pay a certain blockchain transaction fee and gas fee. Li et al. [2] shows that the consumption generated by blockchain crowdsourcing is competitive with the service fee charged by traditional crowdsourcing platforms.

Yuan et al. [4] is the first to design and implement a private and anonymous blockchain-based crowdsourcing system, and solve the problem of data leakage and identity leakage in a decentralized crowdsourcing system. Li et al. [2] designed a decentralized crowdsourcing system with reliability, fairness, security, and low services fee. Liu et al. [5] proposed a system for blockchain-based software crowdsourcing. Yuan et al. [6] used a blockchain-based crowdsourcing system to outsource computationally intensive tasks in machine learning.

Cheng et al. [7] proposed a new complex crowdsourcing task model: Multi-Skill required spatial crowdsourcing task, which is a model closer to real life. For example, a requester wants to hold a party, and he needs to hire some workers to help him cook, play music, and take photographs. This task is a typical multi-skill crowdsourcing task, which requires the participating workers to have matching skills, such as playing instruments, cooking, and photography. Especially with the development of GPS-equipped smart devices and wireless mobile networks, spatial crowdsourcing tasks are gradually commercialized and becomes a very successful model in recent years. Spatial crowdsourcing tasks are gradually commercialized and becomes a very successful business model in recent years. Platforms such as MTurk [8], Uber [9], TaskRabbit [10] are developing greatly rapidly, and online taxi and take-out food have gradually become an important way of life for people.

However, existing blockchain-based crowdsourcing-related efforts cannot support mobile crowdsourcing tasks with multi-skill requirements. They usually involve simple online crowdsourcing tasks. Furthermore, the current blockchain-based crowdsourcing system usually typically set the design focus on implementing a general decentralized crowdsourcing framework. They usually use an intuitive method for task assignment, which is a first-come, first-serve mechanism. If a qualified worker first claims a task, they assign the task to the first-claim worker.

Thus, the goal of the research is to design a blockchain-based crowdsourcing system that (1) supports mobile crowdsourcing tasks with multiple skill requirements, (2) contains an efficient, fair, and transparent task assignment algorithm to improve the completion rate of crowdsourcing tasks while ensuring the fairness of the task assignment process, and (3) is equipped with a reliable fairness assurance mechanism to avoid false reporting and free-riding.

We can ensure that the execution process of the task assignment algorithm is credible and transparent by using smart contract technologies. However, since each node in the blockchain network stores a complete copy of the transaction, when the number of tasks and workers to be allocated in the system is too large, the gas fee generated by the task assignment algorithm executed on-chain may bring additional costs. The main challenge is how to design a fair and transparent task assignment algorithm for a crowdsourcing system to adapt to the blockchain environment with limited computing resources and to meet the demand of spatial crowdsourcing tasks with multi-skill requirements. The proposed task assignment algorithm should take into account many factors such as spatial coordinates, skill requirements, and budget constraints. To solve the above problems, we have implemented a task assignment algorithm and a decentralized crowdsourcing system. Highlights of our original contributions in this paper are as follows:

1. We discussed the decentralized workflow of the blockchain crowdsourcing system and analyzed the problems faced by multi-skilled spatial crowdsourcing tasks in the blockchain crowdsourcing system. And we have built a crowdsourcing system based on blockchain technology, using smart contracts to realize the interaction between requesters and workers, ensuring that the crowdsourcing system is decentralized and fair.
2. We deployed our task assignment algorithm with smart contracts and ensured the transparency and fairness of the algorithm execution process. Finally, we implemented a crowdsourcing-based task quality evaluation mechanism through the oracle contract to ensure the fairness of crowdsourcing participants.
3. We make comparable experiments to compare with TSWCrowd. Experimental results show that our system achieves reliable task matching with lower cost and higher task allocation rates.

The remainder of the paper is organized as follows: The second part shows the related work and the third part discussed the preliminary and problem statement. The Fourth part describes the proposed system architecture. The fifth part analyzed the feasibility of the system. The final part carries on the experiment and makes the result analysis, and the fifth part summarizes the whole article.

2 Related works

In this section, we will describe the proposed task allocation strategy. There are three related topics in our research: spatial crowdsourcing task matching, smart contract on Ethereum, and the blockchain-based crowdsourcing system.

2.1 Spatial crowdsourcing task matching

Spatial Crowdsourcing platform [11] requires workers to travel to a scheduled location and complete the crowdsourcing task, such as taking photographs, cleaning a room, and repairing a computer.

According to the task allocation mechanism, we can classify spatial crowdsourcing into two categories: (1) WST(worker selects task), with the help of a recommendation algorithm [12, 13], the platform recommends tasks to workers and workers choose their interested tasks, and (2) TSW(task selects worker), the platform allocates tasks to workers with optimization goals such as maximum profit. Our work will use the TSW mechanism to maximize the task allocation rate.

Tong et al. [14] proved that the greedy algorithm performs well for some online task matching problems. But they mainly research the model that is one worker finishes one task. In our system, the spatial task requires multiple skills. In other words, tasks in our system may need several workers to complete. Song et al. [15] study the multi-skilled crowdsourcing task assignment problem in real-time spatial crowdsourcing. They believe that the workers and tasks appear dynamically and task assignments should be performed in real-time. But the proposed online-exact algorithm's high complexity will limit its application in the blockchain-based crowdsourcing system. Our system also used the online task allocation mechanism. Once a new task has appeared, the system will assign the optimal team of workers.

Cheng et al. [7] researched the multi-skilled spatial crowdsourcing problem. They want to find an optimal worker-and-task assignment strategy. And they proposed three effective methods to allocate workers to tasks, including greedy, g-divide-and-conquer, and cost-model-based adaptive algorithms. In this paper, the problem of multi-skilled spatial crowdsourcing is defined, and they proved that the problem is NP-hard. But the algorithm complexity of this work is not suitable for the blockchain environment.

Recommendation systems are often used in crowdsourcing systems for task assignment. Yin et al. [16] proposed a holistic personalized recommendation framework , which are based on joint matrix factorization and cognitive knowledge mining. They study the hidden relationships among users, which are mined from the APIs. This helps to optimize the task recommendation model of the crowdsourcing system.

The current crowdsourcing task allocation mechanism relies on the operation and maintenance of a centralized crowdsourcing platform. Therefore, the opacity of the task allocation process is a shortcoming of the centralized crowdsourcing system. There is no guarantee that the platform will perform a dishonest assignment for its benefit. To build a transparent task allocation process, we will introduce the blockchain-based crowdsourcing system to guarantee users' benefit.

2.2 Smart contract on ethereum

IBM [17] gives a simple definition of smart contract on blockchain: Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's

involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met.

When smart contract meets blockchain, it will reveal some unique features. First, the program code of the smart contract will be recorded and verified by the blockchain nodes. Therefore, once a smart contract has been deployed, it can no longer be modified. Second, smart contracts are executed between anonymous and trustless independent nodes, without centralized control and coordination with third-party authorities, which can ensure transparent and fair execution. Third, the smart contract can have its digital encryption currency or other digital assets, and complete the transfer of assets when the predefined conditions are triggered [18].

Ethereum uses the Ethereum Virtual Machine (EVM) to implement a public blockchain platform that supports Turing-complete smart contracts. The entire Ethereum system can be regarded as a singleton distributed state machine. Each node in the Ethereum network runs an EVM instance and uses a consensus mechanism to ensure that the local EVM has the same state as the EVM of other nodes. Many high-level programming languages can be used to write Ethereum smart contracts, such as Solidity and Serpent, and the contract code is compiled into EVM bytecode and deployed on the blockchain for execution.

The automatic generation of smart contract code is an important issue that affects its promotion and development. Gao et al. [19] developed an automatic tool for practical use. This tool allows the user to import Web Services Description Language code as input. As result, the corresponding UPPAAL code is output from the input code. It is very close to a crowdsourcing smart contract scenario, where the requestor enters the task requirements. Then, the tool automatically generates the corresponding smart contract. It will reduce the threshold for using smart contracts in crowdsourcing.

Ethereum is currently the most popular smart contract development platform and can be used to design various decentralized blockchain applications (DApps), such as digital rights management, data sharing, and crowdfunding [20].

2.3 Blockchain-based crowdsourcing system

With the development of blockchain technology, blockchain 2.0 technology is proposed by the Ethereum Blockchain Platform. It supports the Turing complete smart contracts. To improve the crowdsourcing process by taking advantage of the various features of blockchain technology, many researchers use smart contract technology to deploy crowdsourcing systems to the blockchain platform.

Li et al. [2] proposed a general decentralized crowdsourcing system framework based on blockchain, which is named CrowdBC. CrowdBC uses smart contract technology to deploy a crowdsourcing system on the blockchain. According to the classification of task allocation mechanism mentioned above, CrowdBC is a WST system. But recommendation algorithm is difficult to execute on-chain, because of the limited computing resources. Therefore, the system may face the problem of a low task allocation rate.

Kadadha et al. [21] proposed to run the crowdsourcing platform entirely on Ethereum blockchain while incorporating auctions. They used a Repeated-Single-Minded Bidder (R-SMB) auction mechanism to assign workers to the task. This work introduced a task allocation mechanism for the blockchain-based crowdsourcing system. Among the

many workers bidding for a task, the best one is selected and assigned the task. The system is more suitable for scenarios with fewer tasks and more workers, but the problem of low task allocation rate has not been resolved.

Gao et al. [22] proposed a blockchain-based crowdsourcing framework - TSWCrowd (Task Select Worker Crowd). In this framework, tasks are sorted according to specific rules; thus, tasks with higher priority are assigned to workers earlier. The task allocation mechanism only considers the time and distance priority and is not suitable for spatial crowdsourcing tasks with multi-skill requirements. In the following section, we implemented a FIFO algorithm to simulate the allocation mechanism of the TSWCrowd system.

Since blockchain is a weak regulatory environment, trust-based evaluation mechanisms are unreliable. Gao et al. [23] proposed a method to calculate the integrated trust value. The method contains Bayesian inference considering penalty factors and third-party nodes and their reputation. It can be used to calculate the credibility of the participants with blockchain crowdsourcing. Gao et al. [24] proposed a memory-augmented autoencoder approach for detecting anomalies, dishonest nodes among blockchain crowdsourcing participants can be identified.

In summary, no system can not only guarantee the rationality of the task allocation scheme in the blockchain environment, but also consider the spatial crowdsourcing task with multi-skill requirements.

2.4 5g-crowdsourcing

In the 3g era, people could listen to online songs and read online texts anytime and anywhere. Coming to the 4g period, the transmission speed of wireless communication networks has been greatly improved. People can use 4g networks to make video calls with friends and watch videos outdoors. The development of 5g technology is a revolution in wireless communication systems. It is predictable: as 5g technology continues to develop and promote, its high speed, high bandwidth, and low latency will once again change people's lifestyles. Stable high-speed transmission, lower latency, higher positioning accuracy, and other advantages benefit mobile space crowdsourcing a lot. In this section, we review the related work of many researchers in 5g-Crowdsourcing.

Yu et al. [25] designed an edge computing-based photograph crowdsourcing framework, which meets the requirements of latency-intensive and resource-consuming applications in 5G. This work presents a new framework for photograph crowdsourcing in a 5G MEC network environment that can leverage a large number of mobile and IoT devices (e.g., smartphones, surveillance cameras) and real-time neighboring MEC resources for 3D reconstruction.

Supporting the increased bandwidth, the cellular industry is looking for high-frequency bands with high data rates in the spectrum above 24 GHz, commonly referred to as "millimeter wave". The introduction of millimeter waves significantly reduces the coverage radius due to high penetration loss and the blocking characteristics of this spectrum. The reduced coverage radius leads to increased infrastructure costs for network providers. Bandara et al. [26] have designed a crowdsourcing mechanism to collect network data through mobile applications and use these data to generate real-time network coverage maps. In addition, with the help of machine learning techniques, the collected

data will be used to predict future network quality requirements and the location of base stations. The results of this study will benefit network providers to reduce infrastructure costs by optimizing their infrastructure.

Location-based services are evolving rapidly in current wireless communication systems, which will play an important role in future 5G networks. Zhang et al. [27] investigate how to encourage users to participate in crowdsourcing by providing continuous incentives based on their performance. First, they propose performance and rewards aligned contracts to maximize principal utility and provide continuous incentives for users to participate in crowdsourcing activities. Second, extend the incentives from one-dimensional to multi-dimensional to characterize real-world generalities and provide users with a comprehensive reward package. The results show that by using the proposed incentive mechanism, principals successfully maximize utility and users receive a continuous incentive to participate in crowdsourcing activities.

With the development of 5g, it within reaches that combined crowdsourcing mechanisms to solve edge computing problems. Gao et al. [28] proposed a deep reinforcement learning method, which focuses on large-scale edge computing task offloading and obtaining offloading policies with reduced privacy loss, energy consumption, and time delays. Yin et al. [29] propose a new neural network model for quality-of-service prediction in the mobile edge computing environment, which improved the performance and reduces the overfitting problem.

These works reveal the connection between 5g and crowdsourcing, and there is no doubt that mobile spatial crowdsourcing will be applicable to more scenarios as 5g technology continues to evolve. At the same time, the crowdsourcing approach, as a pervasive problem-solving method, can also drive the continuous promotion and development of 5g technology.

3 Methods

3.1 Preliminary and problem statement

Blockchain-based crowdsourcing system is no longer a new vocabulary. Many works have proved the feasibility and security of blockchain-based crowdsourcing. However, most of the previous work is aimed at simple crowdsourcing scenarios, and this article is dedicated to extending the blockchain crowdsourcing system to a more complex crowdsourcing scenario: multi-skilled spatial crowdsourcing and its task allocation process.

In this section, we analyze the decentralized workflow of the Ethereum blockchain to verify the feasibility of our work and explore the challenging issues that may be faced. Ethereum blockchain is a decentralized platform, and its security is guaranteed by a total of 2722 miner nodes [30] in the global Ethereum network.

3.1.1 Decentralized workflow of blockchain

As Fig. 2 shows, we can think of Ethereum as a distributed singleton database, and each miner node is a complete backup of the database. Talk to a database with more than 2700 backups, the data writing operations need to be very cautious. In programming, the usual practice is to lock the database when data needs to be updated to ensure that only one person(Primary) can write to the database at the same time to avoid data consistency-related problems. Ethereum blockchain uses the proof-of-work

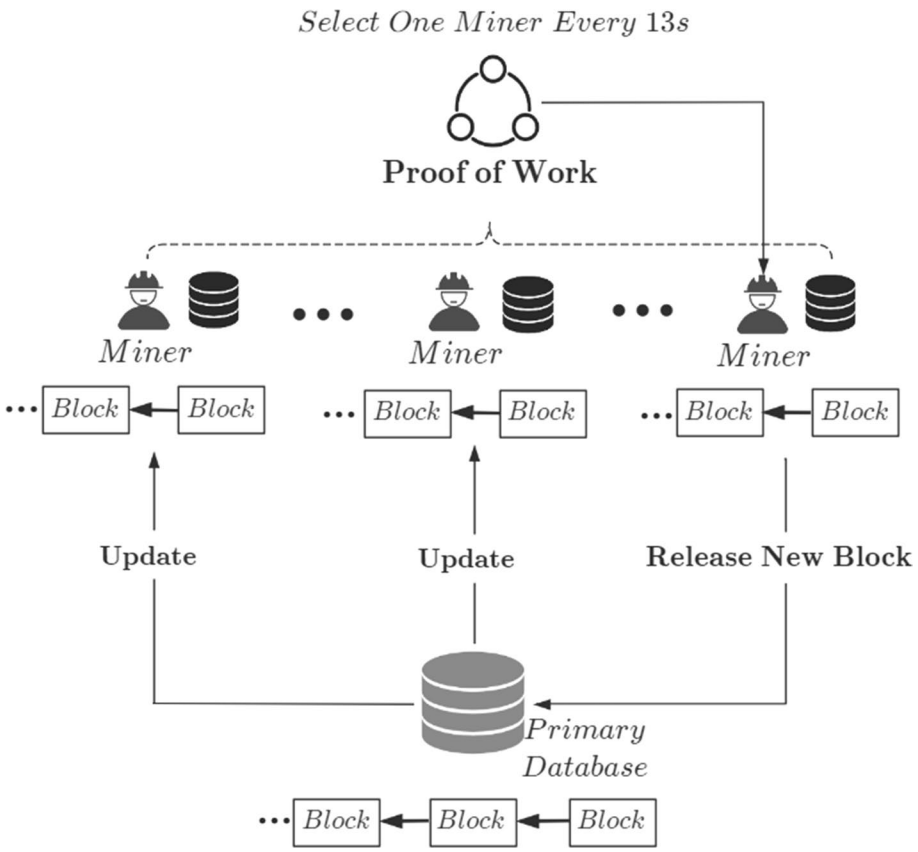

Fig. 2 Ethereum status update process

mechanism to guarantee that only one person can write to the Ethereum state database at a time. Proof-of-work is a competitive consensus algorithm. Nodes with stronger HashRate have a greater probability of winning the competition. This article does not elaborate on the consensus mechanism of proof-of-work. In short, the mechanism will select a node to update the Ethereum database every 13 s.

The release of a new block represents an update of the Ethereum state database, and each block contains many transactions. These transactions include transfer transactions, smart contract deployment transactions, and smart contract invocation transactions. After the new block is released, other miner nodes serve as backups of the database and need to update their local database to be consistent with the primary database. Therefore, the other miner nodes will verify the newly released block and execute the transfer, contract deployment, and contract invocation operations with the time order in the new block. Ethereum guarantees the consistency of the results of operations, especially for contract calls, the same input will produce the same output. In this way, the Ethereum system has completed a round of data updates, which means that part of the transactions submitted by users is executed by the Ethereum system. Every transaction has been verified and executed by all the honest nodes in the global Ethereum network. As long as the HashRate of honest nodes accounts for more than 51% of the network, then Ethereum can be considered sufficiently secure.

There have been many works that have analyzed and discussed the security of the Ethereum system. We can think that once a transaction is uploaded into the blockchain, the consequences of the transaction are irreversible.

3.1.2 Problem statement

We described the decentralized workflow of the Ethereum blockchain, and the decentralized crowdsourcing system based on the blockchain also operates in the same process. CrowdBC represents the current blockchain crowdsourcing system. In simple terms, CrowdBC is more like a public whiteboard in the community. Requesters can post tasks on the whiteboard, and workers can accept tasks on the whiteboard. Although CrowdBC also involves the historical reputation evaluation of workers and the minimum requirements of tasks and other features, it does not involve any task assignment process. Workers and tasks are matched according to personal wishes, which may lead to a low task assignment rate. For this reason, we have included a multi-skilled spatial crowdsourcing task allocation algorithm in our system, and we are committed to implementing this allocation algorithm through an open and transparent Ethereum smart contract. The code of the algorithm is publicly deployed on Ethereum, and the process of algorithm execution is also verified by more than 2700 miner nodes on Ethereum. But this also brings the problem about the time and space complexity of the algorithm. Every calculation and storage in the algorithm are executed in all miner nodes. To avoid wasting resources and reward miners, Ethereum stipulates that miners can charge for these calculations and storage processes. Therefore, we propose the first challenge: (1) how to make the final task allocation rate of the task allocation algorithm as high as possible while limiting the time and space complexity of the algorithm.

In the Ethereum system, all operations are processed in the form of transactions. Ethereum transactions have a certain delay problem, which is mainly due to the 13 s block time set by the Ethereum system. For spatial crowdsourcing, time is a very important constraint. If a task released by the requester needs to be completed within 3 h, but the task release process takes 1 h, it must be an unacceptable time delay. Therefore, the second challenge is: (2) how to make sure that the average transaction latency of Ethereum could support multi-skill spatial crowdsourcing tasks. In fact, the transaction delay of Ethereum can be controlled, and this issue will be analyzed in detail later.

In the blockchain crowdsourcing system, a decentralized blockchain platform replaces the traditional centralized crowdsourcing platform. Without the supervision of a trusted third party, the trust relationship established by the requester and the worker through the blockchain should be reliable. The general steps of the blockchain crowdsourcing system are as follows: (1) the requester publishes tasks through smart contracts, (2) workers receive tasks through smart contracts, and (3) workers complete the task and get the reward. However, both workers and requesters have incentives to profit by not following the process. For example, workers may falsely report that they have completed tasks to obtain task rewards. The requester may deliberately evaluate the workers' work as low quality, thereby reducing their expenses. In CrowdBC, a *solutionEvaluate*$(\cdot)$ function is used to evaluate the quality of the task. This function is set up when the task is released, and the quality of crowdsourcing tasks is evaluated through code logic. This method may be effective for some simple online crowdsourcing tasks (e.g. data labeling,

image recognition, etc.), but it may not be feasible for multi-skilled spatial crowdsourcing. For example, the requester has issued a task that needs to repair the water pipe. It is difficult to say that the pre-set code logic can be used to determine whether the water pipe is repaired or not. So far we can put forward the third challenge: (3) how to implement an effective task quality evaluation mechanism for multi-skilled spatial crowdsourcing tasks in the decentralized blockchain platform. It is interesting to use computer vision methods to do quality evaluation of crowdsourcing tasks. Gao et al. [31] proposed a mutually supervised few-shot segmentation network, which could full utilization a small number of pixel-level annotated samples and performed well in many computer vision tasks. This low training requirements network model can be applied to the quality assurance process of blockchain crowdsourcing.

3.2 System architecture

3.2.1 Overview

After the previous analysis, our proposed blockchain crowdsourcing system needs to implement the following modules.

1. *Decentralized crowdsourcing process* The basic processes of crowdsourcing is user registration, post-task, and reward distribution. To decentralize the crowdsourcing processes, we designed two smart contracts to manage the crowdsourcing process, namely the Worker Manager Contract and Task Manager Contract.
2. *Efficient task allocation algorithm* The multi-skilled spatial crowdsourcing task allocation is a complicated problem. Many works have discussed this problem. Because of the limited computing and storage of the blockchain, these methods in previous works are difficult to apply to the blockchain crowdsourcing system. To reduce the gas consumption of the algorithm and improve the task allocation rate, we designed a task allocation algorithm with a greedy strategy. The algorithm is deployed in the Task Manager Contract to ensure the openness and transparency of the task assignment process.
3. *Reasonable and effective task acceptance mechanism to ensure the fairness of the crowdsourcing process* The multi-skilled spatial crowdsourcing task allocation is a complicated problem. Many works have discussed this problem. Because of the limited computing and storage of the blockchain, these methods in previous works are difficult to apply to the blockchain crowdsourcing system. To reduce the gas consumption of the algorithm and improve the task allocation rate, we designed a task allocation algorithm with a greedy strategy. The algorithm is deployed in the Task Manager Contract to ensure the openness and transparency of the task assignment process.

3.2.2 Implementation of the decentralized crowdsourcing process

The proposed blockchain-based crowdsourcing system is built by smart contracts. We implement two types of smart contracts: Worker Manager Contract (WMC) and

Worker Manager Contract

Multi-Skilled Worker :		
Longitud(uint)	Latitude(uint)	Skills(uint[])
T(uint)	D(int)	Address
A Set of Multi-Skilled Worker :		WorkerP[ ]

Function	***Parameters***	**Return**
WorkerRegister( )	*Worker*	bool
SetLocation( )	*Longitud*、*Latitude*	bool
SetInfo( )	*Skill*、*D*	bool

Fig. 3 Worker manager contract

Task Manager Contract

Multi-Skilled Task :			
Longitud(uint)		Latitude(uint)	Skills(uint[])
T(uint)	B(uint)	Dur(uint)	RequesterAddress
A Set of Multi-Skilled Task :			TaskP[ ]
A Map of Assignment Instance:			Assignment[task]team[ ]

Function	***Parameters***	**Return**
PostTask( )	*Task*	bool
Withdraw( )	*Message.sender*	bool
TaskAssign( )	*Task*, *WorkerP[ ]*	Assignment Instance
Payment( )	*Team[ ]*, *Task*, *Score*	bool

Fig. 4 Task manager contract

Task Manager Contract (TMC). Through these contracts and methods, we deploy the crowdsourcing process on the decentralized Ethereum blockchain.

Worker manager contract

In multi-skill spatial crowdsourcing scenario, the worker is a complex entity, which is called a multi-skilled worker. Multi-skilled workers' data structure is shown in the figure.

WorkerP is a set of multi-skilled workers. For a worker, *Latitude* and *Longitude* represent the specific location in timestamp T, *Skills* represents the skills that the worker has mastered, and D represents the maximum distance the worker is willing to move. *Address* represents the worker's Ethereum account address.

The main functions of this contract are shown in the Fig. 3 *WorkerRegister*$(\cdot)$ is used to initialize a new worker and store the worker's information in the *WorkerP* set. Workers can change their real-time location through the *SetLocation*$(\cdot)$ function, and other information(*Skill* and D) through *SetInfo*$(\cdot)$ function.

Task manager contract

The multi-skilled task's data structure is described in Fig. 4, *TaskP* is a set of multi-skilled tasks. *Latitude* and *Longitude* represent the specific location. T is the task's start time and *dur* is the duration time. So we can calculate that the deadline of a task is $T + Dur$, and workers are required to finish the tasks before the deadline. *Skills* represents the skill requirements for a task. Furthermore, each task is associated with a budget, B, which is the salaries of workers. *RequesterAddress* address represents the

requester's Ethereum account address. $Assignment[task]team$ represents the mapping between a task and a team, indicating the assignment relationship.

The main functions of this contract are shown in the figure. The requester can create a new crowdsourcing task by $PostTask(\cdot)$ function. When the PostTask function is called, the requester has to deposit some eth in the contract as mortgage payments to prevent the "free-riding" attack. The mortgage payments amount is required to be no less than the budget. When the deposited mortgage payments are confirmed, the contract will store the newly created task in *TaskP*. After the task is successfully released, the requester can call the $TaskAssign(\cdot)$ function. The $TaskAssign(\cdot)$ function will select a team from all the assignable workers for this task. The assignment instance returned by the function will be recorded in map $Assignment[task]team$. The task allocation algorithm will be explained in detail in the next section.

After the task is completed by works, the $Payment(\cdot)$ function will pay the workers according to the evaluation score of the task. The task quality evaluation mechanism will be introduced in the later section. The requester can retrieve the remaining budget through the $withdraw(\cdot)$ function. Or if the task is not completed after the deadline, the requester can retrieve all the budget using the $withdraw(\cdot)$ function.

3.2.3 Implementation of the task allocation algorithm

Ethereum will charge a certain gas fee for any operation that consumes storage and calculation resources. And Ethereum sets a gas limit to prevent endless loops in smart contracts, which may cause Ethereum to crash. To achieve a transparent and efficient task allocation mechanism, we deploy the proposed algorithm on Ethereum in the form of a smart contract.

Task allocation algorithm

The task allocation algorithm will consume a certain amount of gas each time it is executed. To reduce this consumption, we consider an optimization scheme for skill coverage rate. As Fig. 5 shows, there is a task requires a set of skill$\{S_1, S_2, S_3, S_4, S_5\}$, and a set of workers$\{W_1, W_2, W_3, W_4, W_5\}$. As the figure shows, every worker has mastered certain skills. In this case, two teams$\{W_1, W_4\}$ and $\{W_1 W_3, W_5\}$ can satisfy the task's require. Under the optimization scheme for skill coverage rate, the team $\{W_1, W_4\}$ will be assigned to the task. While reducing the number of dispatched workers, it can also increase the task allocation rate. With the same set of workers and task sets, our algorithm allows more tasks to be allocated.

The definition of skill coverage rate the size of the intersection of the worker's skills and the task's skills. As in the example above, W_1's skill coverage rate is 3, and W_2's skill coverage rate is 2.

Algorithm 1 shows the greedy framework, the output of algorithm1 is an assignment instance. An assignment instance contains a *task* and a set of *workers*, which means that these workers are assigned to the task. The function *getValidWorker* in the line 1 shows in the algorithm2. In this function, we find all the valid workers for a certain task from the *WorkerP*. And then, we use the function *formTeam* to group a team from the valid workers (line 2). The function *formTeam* shows in the algorithm3 that it will form a team for the input *task* with a greedy strategy(lines3). Then, we

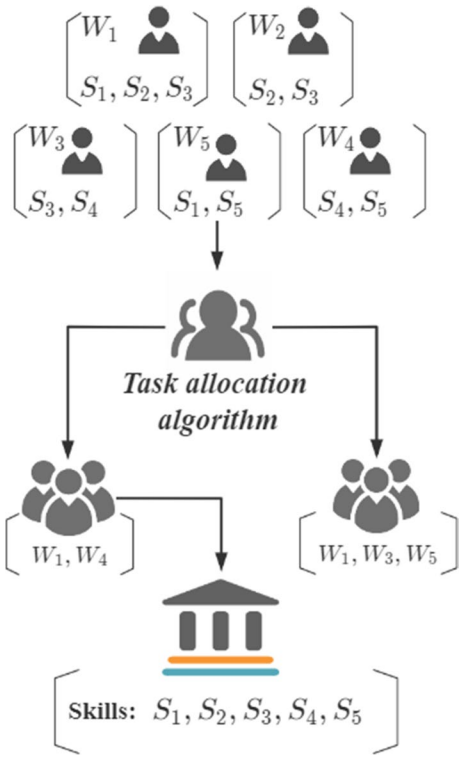

Fig. 5 Example of task allocation

check whether the workers in the *team* can meet the skill requirements of the task and whether the budget of the *task* is sufficient.

Algorithm 1: Task Allocation Algorithm

input : A Task *task*, A Set of Workers *WorkerP[]*
output: *assignmentInstance*

```
validWorkers[ ] = getValidWorker(task, workerP)
team[ ] = formTeam( task,validWorkers[ ] )
if Satisfy ( team[], task ) then
    if calculatePayment(team[]) < task.budget then
        WorkerP[ ] = WorkerP[ ] − team[ ]
        return AssignmentInstance(task,team[ ])
else return false;
```

In Algorithm 2, we select a set of workers for a task. First, we check whether there is an overlap between the skills mastered by the worker and the skills required for the task. Then, we confirm that the distance between the worker and the task is less than

the *Workeri.d* (the maximum distance of the worker willing to move). In this way, we can ensure that the selected workers are effective for this task.

Algorithm 2: getValidWorker

```
input : A Task task, A set of Workers workerP
output: validWorkers[]
foreach Workeri in the WorkerP do
    if SkillSatisfy(task, Workeri) then
        if Workeri.d < distance(task, workeri) then
            validWorkers[] ← workeri
return validWorkers[]
```

Algorithm 3 shows the *formTeam* function. In each loop, we select a worker from *validWorkers*[ ] with a maximum skill coverage rate. Due to Ethereum's gas limit, we need to set a maximum number of loop executions to prevent endless loops.

Algorithm 3: formTeam

```
input : A Task task, validWorkers[ ]
output: team[ ]
for maximum of execution times do
    if SkillSatisfy ( task, team[ ] ) then
        break
    maxworker = null
    maxworker = argmax(team, validWorkers[])
    if maxworker! = null then
        team[] ← maxworker
return team[]
```

The Algorithm 4 *argmax* can find the *maxWorker* with the highest skill coverage rate from all the valid workers. First, we calculate the current *team*[]′*s* skill coverage rate(lines2). And then we traverse all workers in *validWorkers*[] to find the *maxWorker*.

Algorithm 4: argmax

input : $team$, $validWorkers[]$
output: $maxWorker$

```
1 maxSCR = 0
2 maxItem1 = calculateSCR(team, task)
3 maxWorker = null foreach Workeriinthe validWorkers do
4     maxItem2 = calculateSCR(team, task, Workeri)
5     SCR = maxItem1 − maxItem2
6     if maxSCR < SCR then
7         maxSCR = SCR
8         maxWorker = Workeri
9 return maxWorker
```

3.2.4 FIFO task allocation algorithm

In addition to the greedy algorithm, we also implemented an intuitive FIFO task allocation algorithm, which allocates tasks to workers in the order in which they join the system. This algorithm is used as a baseline algorithm to compare our greedy algorithm. The comparison results are shown in the experimental section.

Algorithm 5 shows the FIFO task allocation algorithm. The algorithm adds workers who meet the requirements to the team in chronological order until the team satisfies the task's requirements or traverses all valid workers(lines 3–8). The end of the algorithm is the same as algorithm1(lines 9–13). The algorithm does not consider the worker's task coverage rate and assigns workers to tasks with an intuitive idea, which is first-come, first-serve.

Algorithm 5: FIFO Task Allocation Algorithm

input : $A\ Task\ task, A\ Set\ of\ Workers\ WorkerP[]$
output: $assignmentInstance$

```
validWorkers[] = getValidWorker(task, workerP)
team[] = null
foreach Workeriinthevalidworkers do
    if task.skillNumber == 0 then
        break;
    if skillSatisfy(task, Workeri) then
        team[] ← Workeri
        task.skill = task.skill − Workeri.skill
if Satisfy(team[], task) then
    if calculatePayment(team[]) < task.budget then
        WorkerP[] = WorkerP[] − team[]
        return AssignmentInstance(task, team[])
else return false;
```

3.2.5 The task quality evaluation scheme based on crowdsourcing

In traditional crowdsourcing, the quality evaluation of tasks is supervised by the crowdsourcing platform, through a trusted third party to solve the "false-reporting" and "free-riding" problems that may arise between the requesters and the workers. And it also brings the problem of "subjective arbitration", and fairness is difficult to guarantee. In decentralized crowdsourcing, a trusted third party is removed. But the "false-reporting" and "free-riding" issues have not been resolved. How to ensure fairness among crowdsourcing participants is the main topic of this section.

We propose a task quality evaluation scheme for spatial crowdsourcing tasks with multi-skill requirements. Because of the skill requirements, these crowdsourcing tasks are usually complicated, and it is hard for the requester to predict how the tasks will be completed. In CrowdBC, the requester announces the evaluation strategy of the task at the same time as the task is published. As stated in section 3.2, this kind of scheme is difficult to work in our scenario. Therefore, we propose a task quality evaluation scheme based on a blockchain oracle in a crowdsourcing way.

A blockchain oracle is a service that connects smart contracts with the outside world, primarily to feed information in from the world [32]. In general, we will select 10 juries to score the completion of a crowdsourcing task; then, the scoring result of

Oracle Contract

Score Is A Enum :	Score { A、B、C、D}
The Choosen Workers :	ChossenWorkers[ ]
A Map of Scoring results :	ScoringResults[worker]score
A Map of Task And Score :	Scores[task]score

Function	***Parameters***	**Return**
SubmitScore( )	*score*	bool
Evaluate( )	*SocringResluts*	socre
ChooseWorkers( )	*WorkerP[ ], Task*	ChossenWorkers[ ]

Fig. 6 Oracle contract

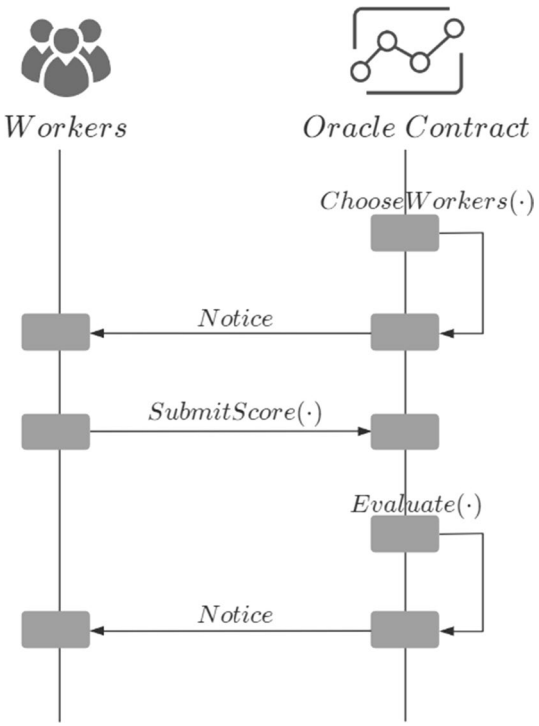

Fig. 7 Task quality evaluation process

the jury is used as the score parameter of the *Payment*(·) function in TMC through the blockchain oracle. We implemented the oracle with a smart contract, and it shows in the Fig. 6

Figure 7 shows the task quality evaluation process. In the first step, the contract uses the *ChooseWorkers*(·) function to select 10 workers as the jury. These selected workers are required to have skills that match the task. If a sufficient number of workers cannot be selected, the function will randomly select workers until the number is sufficient. Secondly, the selected workers will receive a notification and be asked to score the task based on the materials(e.g. photographs of the task scenes) provided by the original participants of the task. By using the *SubmitScore*(·) function, those selected workers can upload their scores to the blockchain. And the *SubmitScore*(·)

function will check whether the person who submitted the score is included in the *ChossenWorkers*[] set. Finally, when the contract has collected enough submissions, it will call the *Evaluate*(·) function to calculate the final result. To encourage the jury to give correct evaluations, workers who submitted the same score as the final score will be rewarded. The final score will be store in the *Scores*[*task*] map for the *payment*(·) function to query. Algorithm 6 described the quality evaluation process.

Algorithm 6: Quality evaluation process

input : $ATasktask, ASetofWorkersWorkerP[]$
output: $Score$

```
jury[] = null;
foreach WorkeriintheWorkerP do
    if jury.size >= 10 then
        break;
    if skillSatisfy(task, Workeri) then
        jury[] ← Workeri
Wait jury update scores,and get an array of Scores[]
task.Score = sum(Scores[])/Scores.size
return task.Score
```

3.2.6 The payment scheme

The final bounty received by the workers of a team participating in the crowdsourcing process is strongly correlated with the rating received for the task. In addition, the bounty received by each worker in a crowdsourcing process is not exactly equal, but a weighted average based on the skill coverage. According to our proposed task assignment algorithm, workers with higher skill coverage will be selected first to complete the task, which means that workers with higher skill coverage may be responsible for more work and therefore get more bounties. Define the skill coverage rate of *workeri* as *SCRi* (skill coverage rate), and the calculation method about *SCR* has been given above.

As shown in Algorithm 7, the payment function is used for the final bounty settlement. First, the total SRC of the team, *totalSCR*, is calculated. SCRi/totalSCR represents the workload of a worker on a crowdsourced task, and a higher skill coverage usually represents a higher workload. The task is scored on a ten-point scale, and the ratio of the task acceptance score to the full score (Score/10) indicates the task completion rate. Finally, based on the workload and task completion, the algorithm allocates the budget pledged at the beginning of the task to the workers involved in the crowdsourcing process. The final remainder is returned to Requester.

Algorithm 7: Payment function

input : Task $task$, a team of the Task, $team[]$
output:

$totalSCR = 0$
$CompletionRate = Score/10$
foreach $Workeriintheteam[]$ **do**
    $totalSCR+ = SCRi$
foreach $Workeriintheteam[]$ **do**
    $bounty = (SCRi/totalSCR) * CompletionRate * task.budget$
    $transfer(bounty, workeri)$
$transfer(task.budget, task.RequesterAddress)$

3.3 System process

3.3.1 User registration

Requesters and workers need to register in the system before they participate in the crowdsourcing process(e.g., post task, accept tasks, get payment, etc.). The Ethereum address can uniquely identify a user; workers in our system have much other information(e.g., location information, effective active time, and skills mastered) besides the identity. Therefore, workers should register other information into the system through the Worker Manager Contract.

3.3.2 Creating task

After the requester finished the registration, they can publish tasks through Task Manager Contract. To publish a task, requesters call the *PostTask*$(\cdot)$ function and pass in the longitude, latitude, skill requirements, and other information of the task. Furthermore, the requester should transfer the budget to the Task Manager Contract. When the task is finished, the pre-transferred budget will be used to pay the workers, and the rest will be returned to the requester.

3.3.3 Task assignment

After a task is published, the requester can call the *TaskAssign*$(\cdot)$ function to assign workers to the published task. This function uses the task allocation algorithm mentioned above to form a team for the task from the current active workers. Sometimes the allocation will fail because the number of active workers is insufficient. At this time, the requester can wait for new active workers to join the system and then call the allocation function again, or call the withdraw function to retrieve his prepaid budget. If the assignment is successful, the assigned worker should go to the task location to complete the task.

3.3.4 Task quality evaluation and payment

The process of task quality assessment has been described in the previous section. When the task is finished, the original participants of the task should provide some evidence materials for scoring. And the Oracle Contract will use the method mentioned above to score the task. After the scoring is completed, either the requester or the workers can call the $Payment(\cdot)$ function to pay the workers. It is worth noting that the $Payment(\cdot)$ function only accepts one parameter, which is the task. $Payment(\cdot)$ function will query Oracle Contract to get the score of the task and reward the honest workers who participated in the scoring. In this way, we can ensure that the scoring process is honest and the crowdsourcing process is fair.

3.4 System feasibility analysis

In this section, we mainly discuss the feasibility of the system. It is expanded from two aspects, one is the transaction delay problem of the Ethereum blockchain mentioned in the third section, and the other is the time complexity of the task allocation algorithm.

3.4.1 Analysis of ethereum transaction delay problem

Ethereum's transaction time delay is mainly caused by the block generation time set by the Ethereum system. At present, the block generation time of Ethereum is about 13 s. In other words, every 13 s, Ethereum will process a batch of pending transactions. In the most ideal case, a transaction will be processed after one block interval (13 s). Counting the transmission delay of the transaction in the Ethereum network, we can assume that the transaction has a basic delay of two block intervals. For the current design of Ethereum, its TPS (transaction per Second) is between 15 and 25 [33]. Compared with traditional centralized systems, this is not a very high value. (e.g. Visa, 6000TPS). Once the number of transactions exceeds the capacity of Ethereum, it is easy to generate transaction congestion. For the period of mid-2020, Ethereum's average transaction delay reached 44 min. Excluding some extreme cases, most of the time the average transaction delay of Ethereum is between 2 and 5 min [34].

Moreover, the congestion waiting time caused by too many transactions can be controlled. If the requester needs to publish a very urgent task, the requester can appropriately increase the gas price and priority fee. For miners, packaging this transaction can get more revenue, which can naturally attract miners from the Ethereum network to prioritize packaging your emergency transaction. Of course, this will increase the transaction fee that needs to be paid for the transaction.

In terms of current Ethereum transaction congestion and throughput, the transaction delay will not exceed 5 min. If the task issued by the requester has a time limit of 3 h, and the delay of the Ethereum transaction takes up about 2% of the time, we believe that this time delay is acceptable.

For multi-skilled spatial crowdsourcing tasks such as repairing water pipes and holding parties, due to the commuting time of workers, most spatial crowdsourcing tasks will have a time limit of more than 3 h. However, for Uber and Didi, this kind of time-critical crowdsourcing task, the 5 min delay may require users to carefully consider whether it

is acceptable. According to the official Ethereum, ETH2.0 may be launched by the end of 2022. At that time, the system throughput of Ethereum will be improved by several orders of magnitude, and the transaction delay problem will be completely solved.

3.4.2 Analysis of algorithm time complexity

We use a greedy task allocation algorithm based on skill coverage rate. And form a team of workers from all workers based on the skill requirements and space-time constraints of the multi-skill spatial crowdsourcing task. In the first step, the algorithm uses the *getValidWorker* method to find the set of workers that meet the conditions from all workers. Secondly, the algorithm uses the *formTeam* method to form a team from all the valid workers according to the greedy strategy.

In the above process, *getValidWorker* method needs to traverse the worker set to find workers who meet the conditions. The *formTeam* method also needs to traverse the set of workers found in the last step to form a team. In each traversal, some calculations need to be done, such as judging whether the skills meet the requirements, calculating the skill coverage, and the logic of team formation. We set the time complexity of each traversal to be t, and the size of the worker to be n. The algorithm needs to traverse the worker set twice, so we can conclude that the time complexity of the algorithm is $O(2n * t)$. In general, the proposed algorithm has linear time complexity, and its gas consumption is compared in detail in the experiment section.

Our algorithm mainly considers the skill coverage rate of workers, which means that the number of workers we select for a team will be as small as possible. This not only reduces the requester's budget consumption but also increases the overall task allocation rate. A higher task allocation rate also means fewer algorithm execution times. From this point of view, our algorithm can also reduce consumption on the Ethereum blockchain. The detailed experimental results are shown in the next section.

4 Results and discussion

The System we have implemented is deployed on the Ethereum blockchain. The optimization of the task allocation algorithm in the system can achieve a higher success rate of task assignment.

To evaluate these goals, we deploy the system on a local private chain and compare the task allocation success rate with an intuitive method.

4.1 Program setup

We use both real and synthetic data to test our proposed task allocation algorithm. The location and time data come from the real data set. Then, we use a random algorithm to

Table 1 Dataset parameters

Number of worker	**[60, 120]**
Location (latitude, longitude)	([102,105],[102,105])
Time	[1615910403, 1615996772]
Skills	[1, 10]

construct the skills of the workers and the skill requirements for the tasks. The specific data parameters are shown in Table 1.

The smart contracts are deployed on the local private Ethereum blockchain. The programming language we used is solidity [35]. The truffle framework and ganache-cli are used to deploy the smart contracts.

The data source uses the real data source published by DIDI, but since DIDI's data source only has latitude, longitude, and time constraints, it does not contain the attributes of skills. So in the experiments of this research, some processing was done on this data source. A skill constraint is added for each crowdsourcing task. The integers defined in the range of [1,10] represent the skill constraints. Randomly generated skill constraints were constructed for each task based on real latitude, longitude, and time data.

4.2 Task allocation process

The blockchain crowdsourcing system designed in the CrowdBC [2] does not include the task assignment algorithm, and workers claim the tasks of interest on their own. The first worker who claims a task often gets that task, so the FIFO algorithm implemented above can simulate the task assignment process of CrowdBC. The task assignment algorithm of [22] is based on time priority and has some similarities with the task assignment algorithm of FIFO. Since the work that can be compared does not open-source code, a FIFO algorithm is implemented in this paper as a baseline for the comparison experiment. It will confirm whether the skills of the workers meet the task requirements, and then assigns tasks to the workers in the order of the time they joined the system.

As shown in Fig. 8, the ordinate represents the success rate of task assignment, and the abscissa represents the number of workers in the system. We synthesized 3 sets of 25 task data to represent tasks of different difficulties. In Fig. 8a, each task requires 3 skills, representing the simple task. Figure 8b, c, respectively, represents a medium and difficult task. We compared the success rate of task assignment with 60, 90, and 120 workers, and adjust the difficulty of the task for comparative experiments.

Obviously, as the number of workers increases, the success rate of task assignments is rising. The greedy algorithm we proposed can easily achieve a success rate of more than 90% when the number of workers is sufficient. At the same time, when the task is difficult and the number of workers is small, the Greedy algorithm is also significantly better than the FIFO algorithm.

When the task is simple (with 3 skill requirements), the FIFO algorithm can also exhibit a success rate of 80% with a sufficient number of workers guaranteed. However, as the difficulty of the task increases, the success rate of the FIFO algorithm decreases a lot when the requirements for workers are higher. A success rate around 50% will mean that the FIFO algorithm may need two assignment processes to complete the task assignment, compared to our proposed greedy algorithm which only needs one time to complete the task assignment. This indicates that the greedy algorithm can save 50% of the gas consumption than the FIFO algorithm.

We can conclude that the Greedy algorithm has a higher task allocation rate. If we can assume that there are about 90 active workers in the system, then a 90% task allocation rate can be easily achieved through the greedy algorithm proposed.

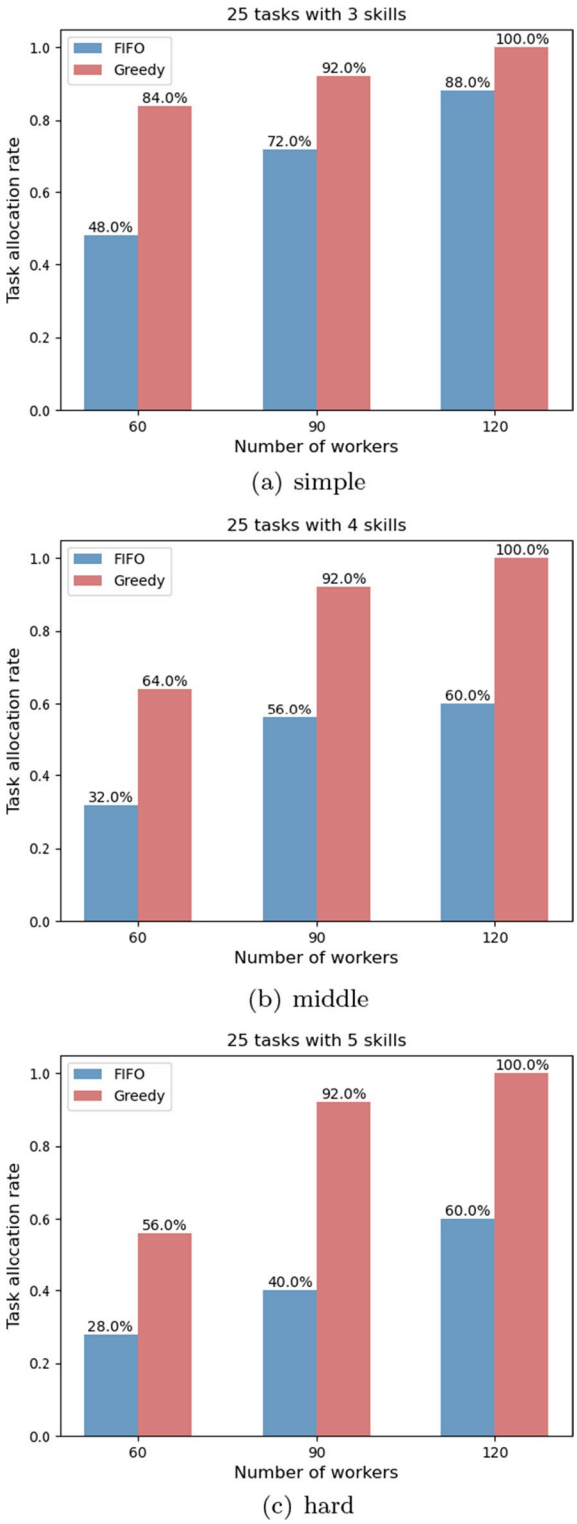

Fig. 8 Comparison of the success rate of task assignment

4.3 Smart contract cost

In this experiment, we verified the economic feasibility of the proposed system. To achieve decentralization, the blockchain-based crowdsourcing system uses the Ethereum blockchain platform to replace the centralized platform in traditional crowdsourcing. There are currently 2722 nodes [30] that maintain Ethereum globally. To incentive these miner nodes, the Ethereum platform charges a certain gas fee for each operation that consumes computer resources. To compare with TSWCrowd [22], we use the gas price and ether price of Ethereum in July 2020. 1gas is set to 5×10^{-9} Ether, at the same time 1Ether = \$233.02(2020.7.6). Table 2 shows the gas consumption in Ethereum.

Compared with TSWCrowd, the cost of our system is slightly higher. In the TSWCrowd, one round of task allocation costs \$3.29. In our system, the task allocation costs \$5.9, but a higher task allocation rate can ensure that the allocation is completed in one round of execution. In TSWCrowd, the number of rounds of algorithm execution may be higher, which will cause additional costs.

The centralized platform Mechanical Turk (MTurk) [8] needs to pay at least a 20% service fee to the platform to manage the crowdsourcing process. Therefore, the cost of our system to complete a crowdsourcing process is greatly reduced.

In conclusion, our system has good economic feasibility, which decentralizes the crowdsourcing process and significantly reduces the additional consumption of the crowdsourcing process.

5 Conclusions

The paper uses decentralized blockchain to replace the centralized crowdsourcing platform, thereby realizing a decentralized crowdsourcing system. In addition, a crowdsourcing task allocation algorithm is proposed, which realizes support for multi-skilled crowdsourcing tasks while ensuring a higher task allocation rate. Compared with traditional crowdsourcing, our system has two main advantages: (1) the task allocation process is transparent and fair, and (2) there is no centralized platform management crowdsourcing process, which can protect the privacy of users. The system uses the feature of the blockchain to realize the immutability of transactions and permanent storage, and the use of smart contract technology to ensure that the execution of the task allocation algorithm is fair and transparent. Moreover, the blockchain is an anonymous platform, which protects the privacy of users. Compared with the TSWCrowd blockchain-based crowdsourcing model, our system has a higher task allocation rate under the same conditions. While improving the completion rate of crowdsourcing tasks, it also reduces the additional cost of repeated calls

Table 2 Gas consumption of main function

Operation	Gas cost	Ether cost	USD cost
Greedy task allocation	5072974	0.02536	5.909
FIFO task allocation	2893366	0.01446	3.369
UserRegister	193025	0.000965	0.224
PostTask	276352	0.00138	0.321

to the allocation algorithm. And our system supports multi-skilled crowdsourcing tasks, which is more suitable for real spatial crowdsourcing application scenarios.

In future work, we will focus on optimizing the task allocation algorithms and user privacy protection. In the current method, each crowdsourcing task costs nearly $6.5, which is still too expensive for some small and micro crowdsourcing tasks. Although the blockchain platform is anonymous, there is still the risk of exposure [36] on the chain, which means that the user's private information can be analyzed through a clustering attack. We want to find a way to execute the transparent and fair task allocation algorithm on-chain while protecting users' privacy. This can make blockchain crowdsourcing truly significant advantages compared to traditional crowdsourcing.

Abbreviations

GPS	Global positioning system
WST	Worker selects task
TSW	Task selects worker
EVM	Ethereum virtual machine
DApps	Decentralized blockchain applications
R-SMB	Repeated-single-minded bidder
FIFO	first in first out
WMC	Worker manager contract
TMC	Task manager contract
TPS	Transaction per second
ETH	Ethereum

Acknowledgements
We sincerely thank the Reviewers and the Editor for their valuable suggestions.

Author contributions
WX proposed the new idea of the paper and participated in the outage performance analysis. HD performed the simulations and drafted the paper. JH, DL and YC conceived of the study, and participated in its design and coordination and helped to draft the manuscript. All authors have read and approved the final manuscript.

Funding
This work is supported in part by the Key R&D Program of Zhejiang (2022C03087) and R&D Program of ISTIZ (2021RD02).

Declarations

Availability of data and materials
Not applicable.

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
The authors declare that they agree with the consent for publication.

Received: 18 January 2022 Accepted: 1 June 2022
Published online: 21 June 2022

References

1. A. Chaer, K. Salah, C. Lima, P.P. Ray, T. Sheltami, Blockchain for 5g: opportunities and challenges, in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6 (2019). https://doi.org/10.1109/GCWkshps45667.2019.9024627
2. M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, R.H. Deng, Crowdbc: a blockchain-based decentralized framework for crowdsourcing. IEEE Trans. Parallel Distrib. Syst. **30**(6), 1251–1266 (2019). https://doi.org/10.1109/TPDS.2018.2881735
3. X. Zhang, G. Xue, R. Yu, D. Yang, J. Tang, Keep your promise: mechanism design against free-riding and false-reporting in crowdsourcing. IEEE Internet Things J. **2**(6), 562–572 (2015). https://doi.org/10.1109/JIOT.2015.2441031
4. Y. Lu, Q. Tang, G. Wang, Zebralancer: private and anonymous crowdsourcing system atop open blockchain, in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 853–865 (2018). https://doi.org/10.1109/ICDCS.2018.00087

5. K. Liu, W. Chen, Z. Zhang, Blockchain-empowered decentralized framework for secure and efficient software crowdsourcing, in *2020 IEEE World Congress on Services (SERVICES)*, pp. 128–133 (2020). https://doi.org/10.1109/SERVICES48979.2020.00039
6. Y. Lu, Q. Tang, G. Wang, On enabling machine learning tasks atop public blockchains: a crowdsourcing approach, in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 81–88 (2018). https://doi.org/10.1109/ICDMW.2018.00019
7. P. Cheng, X. Lian, L. Chen, J. Han, J. Zhao, Task assignment on multi-skill oriented spatial crowdsourcing. IEEE Trans. Knowl. Data Eng. **28**(8), 2201–2215 (2016). https://doi.org/10.1109/TKDE.2016.2550041
8. Amazon: Amazon Mechanical Turk. https://www.mturk.com/. Accessed 2021
9. Uber: Uber. https://www.uber.com/. Accessed 2021
10. TaskRabbit: TaskRabbit. https://www.taskrabbit.com/. Accessed 2021
11. L. Kazemi, C. Shahabi, Geocrowd: enabling query answering with spatial crowdsourcing, in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems. SIGSPATIAL '12*, Association for Computing Machinery, New York, NY, USA, pp. 189–198 (2012). https://doi.org/10.1145/2424321.2424346
12. A. Hu, Y. Gu, Mobile crowdsensing tasks allocation for mult-parameter bids, in *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 489–493 (2017). https://doi.org/10.1109/ITOEC.2017.8122344
13. R. Qiao, S. Yan, B. Shen, A reinforcement learning solution to cold-start problem in software crowdsourcing recommendations, in *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pp. 8–14 (2018). https://doi.org/10.1109/PIC.2018.8706279
14. Y. Tong, J. She, B. Ding, L. Chen, T. Wo, K. Xu, Online minimum matching in real-time spatial data: experiments and analysis, Proc. VLDB Endow (2016). https://doi.org/10.14778/2994509.2994523
15. T. Song, K. Xu, J. Li, Y. Li, Y. Tong, Multi-skill aware task assignment in real-time spatial crowdsourcing. GeoInformatica **24**(1), 153–173 (2020). https://doi.org/10.1007/s10707-019-00351-4
16. Y. Yin, Q. Huang, H. Gao, Y. Xu, Personalized apis recommendation with cognitive knowledge mining for industrial systems. IEEE Trans. Industr. Inf. **17**(9), 6153–6161 (2021). https://doi.org/10.1109/TII.2020.3039500
17. IBM-blockchain: smart-contracts. https://www.ibm.com/topics/smart-contracts. Accessed 2021
18. S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, F.-Y. Wang, Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Trans. Syst. Man Cybern. Syst. **49**(11), 2266–2277 (2019). https://doi.org/10.1109/TSMC.2019.2895123
19. H. Gao, Y. Zhang, H. Miao, R.J.D. Barroso, X. Yang, Sdtioa: modeling the timed privacy requirements of iot service composition: a user interaction perspective for automatic transformation from bpel to timed automata. Mobile Netw. Appl. **26**(6), 2272–2297 (2021). https://doi.org/10.1007/s11036-021-01846-x
20. X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A.B. Tran, S. Chen, The blockchain as a software connector, in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 182–191 (2016). https://doi.org/10.1109/WICSA.2016.21
21. M. Kadadha, R. Mizouni, S. Singh, H. Otrok, A. Ouali, A. Abcrowd: an auction mechanism on blockchain for spatial crowdsourcing, in *IEEE Access*, pp. 1–1 (2020). https://doi.org/10.1109/ACCESS.2020.2965897
22. L. Gao, T. Cheng, L. Gao, Tswcrowd: a decentralized task-select-worker framework on blockchain for spatial crowdsourcing. IEEE Access **8**, 220682–220691 (2020). https://doi.org/10.1109/ACCESS.2020.3043040
23. H. Gao, C. Liu, Y. Yin, Y. Xu, Y. Li, A hybrid approach to trust node assessment and management for vanets cooperative data communication: historical interaction perspective. IEEE Trans. Intell. Transp. Syst. (2021). https://doi.org/10.1109/TITS.2021.3129458
24. H. Gao, B. Qiu, R.J. Duran Barroso, W. Hussain, Y. Xu, X. Wang, Tsmae: a novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder. IEEE Trans. Netw. Sci. Eng. (2022). https://doi.org/10.1109/TNSE.2022.3163144
25. S. Yu, X. Chen, S. Wang, L. Pu, D. Wu, An edge computing-based photo crowdsourcing framework for real-time 3d reconstruction. IEEE Trans. Mob. Comput. **21**(2), 421–432 (2022). https://doi.org/10.1109/TMC.2020.3007654
26. L. Bandara, H. Rathnasinghe, E. Kavinda, C. De. Seram, M.M. Hansika, Intelligent crowd-sourced 5g heat-map with event-driven architecture, in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0982–0987 (2021). https://doi.org/10.1109/IEMCON53756.2021.9623247
27. Y. Zhang, Y. Gu, M. Pan, N.H. Tran, Z. Dawy, Z. Han, Multi-dimensional incentive mechanism in mobile crowdsourcing with moral hazard. IEEE Trans. Mob. Comput. **17**(3), 604–616 (2018). https://doi.org/10.1109/TMC.2017.2732982
28. H. Gao, W. Huang, T. Liu, Y. Yin, Y. Li, Ppo2: location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems. IEEE Trans. Intell. Transp. Syst. (2022). https://doi.org/10.1109/TITS.2022.3169421
29. Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, Z. Mai, Qos prediction for service recommendation with features learning in mobile edge computing environment. IEEE Trans. Cognit. Commun. Netw. **6**(4), 1136–1145 (2020). https://doi.org/10.1109/TCCN.2020.3027681
30. Ethereum: ethernodes. https://www.ethernodes.org/. Accessed 2021
31. H. Gao, J. Xiao, Y. Yin, T. Liu, J. Shi, A mutually supervised graph attention network for few-shot segmentation: the perspective of fully utilizing limited samples. IEEE Trans. Neural Netw. Learn. Syst. (2022). https://doi.org/10.1109/TNNLS.2022.3155486
32. Dandv: BlockchainOracle. https://en.wikipedia.org/wiki/Blockchain_oracle. Accessed 2021
33. web: TPS. https://cryptotps.com/tps-capacity-bitcoin-ethereum-ripple-xrp-tron-eos/. Accessed 2021
34. ethgasstation: GasStation. https://ethgasstation.info/. Accessed 2021
35. ETH: Solidity. https://solidity.readthedocs.io/en/latest/. Accessed 2021
36. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G.M. Voelker, S. Savage, A fistful of bitcoins: characterizing payments among men with no names. Commun. ACM **59**(4), 86–93 (2016). https://doi.org/10.1145/2896384

Publisher's Note