A Hashgraph-Based Knowledge Sharing Approach for Mobile Robot Swarm

Xiao Shu[1], Bo Ding[1(✉)], Jie Luo[1], Xiang Fu[1], Min Xie[1], and Zhen Li[2]

[1] College of Computer, National University of Defense Technology, Changsha, China
shuxiao19@nudt.edu.cn, dingbo@aliyun.com

[2] College of Information and Communication, National University of Defense Technology, Wuhan, China

Abstract. Common knowledge in a group of robots, i.e., the knowledge known by everyone or nearly everyone, can significantly promote the efficiency of robot collaboration. In a decentralized environment, it can be achieved through blockchain technology. However, traditional blockchain platforms such as Ethereum are based on Proof of Work (PoW), which requires huge amounts of computation and is not suitable for robots with limited computing resources. And the lack of a stable, fully-connected network will greatly reduce the performance of the traditional blockchain technology as well. To address these challenges, we propose a novel peer-to-peer knowledge sharing approach for mobile robot swarms in this paper. This approach is based on hashgraph, a distributed ledger technology that uses directed acyclic graphs to achieve consensus and does not need huge computational power. We also enhance hashgraph to adapt it to the mobile network environment with a limited communication range for each robot and dynamic network topology in the swarm. With a set of motivated scenarios of collective decision making, we verified the effectiveness of our approach and the results show that our approach helps robot swarm collaborate more efficiently with less computation and waste of resources than the approach based on the traditional blockchain.

Keywords: Robot swarm · Hashgraph · Common knowledge · Consensus algorithm

1 Introduction

A robot swarm is made up of a large group of locally interacting individuals with common goals, and it has been applied in many scenarios, such as site selection [6], and task assignment [4], etc. Common knowledge can be used to form a resultant force on the whole based on the local behavior of the individual to improve the collaborative efficiency, and it has been proved useful in applications such as task allocation in unknown environment [11], synchronization between different swarms [16]. However, it's difficult for swarm robots to support global synthesis of sensory information collected by the swarm without a centralized infrastructure [15].

To solve this problem, the traditional blockchains [20,24] can be adopted to provide the robot swarm with common knowledge in a decentralized, verifiable, and secure way,

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 407, pp. 158–172, 2021.
https://doi.org/10.1007/978-3-030-92638-0_10

which has been used in many applications [19,26]. The blockchain, such as Ethereum [29], can act as a peer-to-peer database and computing system in the swarm. Due to the characteristics of PoW [10], the most widely used consensus algorithm of traditional blockchains, robots can mine new blocks locally and store information in them. Despite the environment of limited communication range and dynamic network topology, the blockchains of different robots can be synchronized once robots come into the communication range, and a single point of failure will not affect the synchronization process of blockchains. Besides, the 51% computational power limitation and data verifiability can help the blockchain resist attacks from Byzantine robots efficiently.

Although the approach based on traditional blockchains shows an advantage in achieving common knowledge, it's not suitable for the mobile network environment of robot swarms. PoW requires robots to keep solving puzzles constantly through the reward mechanism to add new blocks to the blockchain [25], which means that electricity will be wasted on extra computations. In Bitcoin [20], PoW takes 10 min to generate a block [9], and most of the time is used for solving the puzzles, which is a huge burden for the robot of onboard computer and limited resources. Besides, the blockchain forks may occur if robots find the solutions to the PoW puzzles at almost the same time, or blocks are not spread through the entire network [26]. Robots in different clusters may take actions based on the forks of blockchains, which may cause unexpected situations. Discarding the forks will result in inefficient use of information and resources.

To address the above-mentioned challenges, we propose a novel peer-to-peer knowledge sharing approach for mobile robot swarms. This approach is based on hashgraph [3], a data structure and consensus algorithm. With no PoW, hashgraph allows the nodes to create many events per second and it doesn't call for high computational power. Besides, hashgraph is 100% efficient in information utilization because no events will be discarded. To adapt to the mobile network environment of the robot swarm, we enhance the gossip rules of hashgraph, that is, each node can only synchronize the local hashgraph with its neighbor nodes. In our hashgraph approach, each robot acts as a node of hashgraph and shares knowledge with neighbors through hashgraph synchronization. As the local hashgraph grows, the robot can achieve common knowledge and take action based on it. In the motivated scenarios of single-feature [27] and multi-feature [7] collective decision making, we test the efficiency of the hashgraph approach in consensus achievement and task allocation respectively. The experimental results show that the mobile robot swarm in the hashgraph approach can collaborate efficiently with less computational power and higher resource utilization than the traditional blockchain approach.

The rest of this paper is organized as follows. We give an overview of robot swarm, consensus algorithms in robot swarms, hashgraph in Sect. 2. We represent the motivated scenarios in Sect. 3 and the hashgraph approach in Sect. 4. Experiments are presented in Sect. 5. We conclude this paper in Sect. 6.

2 Related Work

2.1 Robot Swarm

The robot swarm is a typical distributed system consisting of a large number of simple robots, and the collective behavior of it emerges from numerous local interactions

between individuals without any central control [23]. It is increasingly applied to perform dynamic and complex tasks [1,18]. Ebert et al. [7] propose a decentralized algorithm and a dynamic task allocation strategy that allows the agents to lock in decisions on multiple features in a finite time. Wang et al. [28] propose an adaptive mechanism and design a group-based spatial formation algorithm for swarm robots to adapt to the dynamic environment. But the main focus of these algorithms is still how to manually set the behavior rules of each robot in advance, while how to obtain common knowledge to coordinate swarm behavior is not the point.

The idea of applying blockchain to robot swarms to provide common knowledge is first proposed in [9] and has been applied in some scenarios, such as trust management [14], monitoring quarantine areas [2], etc. A framework for ontology-oriented robots' coalition formation based on blockchain in cyberphysical systems is also proposed [19]. It should be pointed out that due to the high requirements of computational power and the low resource utilization, the blockchain is difficult to be applied to the mobile network environment of robot swarm.

2.2 Consensus Algorithms in Robot Swarms

The robot swarm is similar to the distributed system, and the consensus algorithm can be applied to enable individual units in the robot swarm to reach a common perspective of objectives and state of the world [12]. Many consensus algorithms have emerged in the field of distributed systems, such as Paxos [13], PBFT [5], etc., and some of them have been applied in applications, such as the multi-agent system based on Paxos [17]. But these traditional consensus algorithms often divide the nodes into several different roles, such as leaders, followers, etc., which doesn't meet the requirements of decentralization in a robot swarm. Besides, classical solutions to distributed consensus problems, such as two-phase commit [13], usually require frequent committing, acknowledging, and other interactions among a group of nodes, which is difficult to be applied in an environment of dynamic network topology and limited communication range.

Blockchain technology demonstrates that without a controlling authority, a group of agents can still reach an agreement on a particular state of affairs and record that agreement in a peer-to-peer network [9], and it has been applied in many studies of multi robots [2,14]. Proof of Work (PoW) is the proof that a certain amount of computational power was consumed to solve a hard puzzle that allows the adding of a block to the blockchain, the process of which is called mining. It is based on SHA256, which inputs a message of arbitrary length smaller than 2^{64} bits and produces as output a 256-bit message digest of the input, resulting in a huge waste of electricity. In Ethereum, the difficulty of mining puzzles can be adjusted to the hash power of the network [25], but the onboard computer of the robot determines that it can only solve less difficult puzzles. Due to the peer-to-peer network and limited communication range, the lower the difficulty of the puzzle, the more likely the robots are to mine new blocks at the same time, leading to the problem of forks. According to the rules of the blockchain, the information in the forks will be discarded, which is a waste of information and resources.

2.3 Hashgraph

In recent years, some new blockchain data structures and consensus algorithms have emerged, one of which is hashgraph [3], a data structure and consensus algorithm that uses Directed Acyclic Graphs(DAG) for time-sequencing transactions. As shown in Fig. 1, hashgraph is made up of several nodes and many events. Each node maintains a local hashgraph. The process of synchronizing their local hashgraph is named gossip about gossip. When Carol randomly tells another node, for example, David, everything she knows, David records the fact by creating a new event "d4", which contains the hash of his most recent event "d3", and the hash of Carol's most recent event "c2". At the same time, David stores a timestamp and some transactions into the event "d4". Thus it can be seen, hashgraph stores information about the history of how nodes have gossiped and all known events. Every node repeatedly chooses another node at random and gives it all the events that it doesn't yet know. As the process of gossip about gossip goes on, based on the local hashgraph, everyone can determine the consensus order of the events according to the virtual voting protocol.

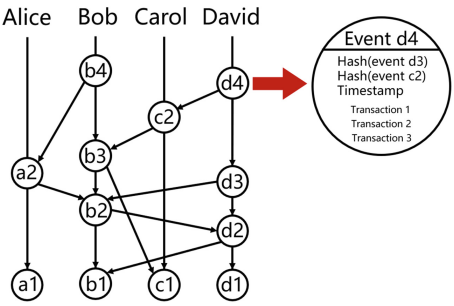

Fig. 1. Hashgraph of four nodes (Alice, Bob, Carol, David). Carol tries to gossip with David, David creates an event "d4" containing hashes of the events "c2" and "d3".

The process of gossip about gossip helps information spread exponentially fast and reliable, which ensures that every node will eventually know every event. Besides, the consensus is determined upon every node's local hashgraph, which greatly reduces the requirements for communication conditions. These characteristics make it possible to combine hashgraph with mobile robot swarms in an environment of dynamic network topology and limited communication range. Compared with blockchain, with no need for PoW, the throughput of hashgraph is high, and there is no problems of computational power and forks with it.

3 Motivated Scenarios

In this paper, we adopt collective decision making scenarios of swarm robots to illustrate our approach. In these scenarios, robots need to reach one or more common decisions among available options, e.g., selecting an escape route or judging the degree of

pollution in the cases of nuclear accident emergency rescue, which results in the problems of consensus achievement and task allocation.

The scenario environment is a $1.2\,\mathrm{m} \times 1.2\,\mathrm{m}^2$ area with a 25 mm thick gray border and contains one or more features (see Fig. 2). The single-feature scenario [27] is characterized by a grid consisting of several black and white tiles. The fill ratio of white tiles to the whole area represents the feature of this scenario. The multi-feature scenario, which is first proposed by Ebert et al. [7], is extended from the single-feature scenario. Several different single-feature layers are combined to form the scenario, and the fill ratio of the light color in each single-feature layer represents one of the features of that scenario. If the fill ratio of the light color is above 0.5, the value of that feature in the scenario is equal to 1, otherwise, it is equal to 0. The value of a feature represents that whether the light color or the black color is more frequent in the environment.

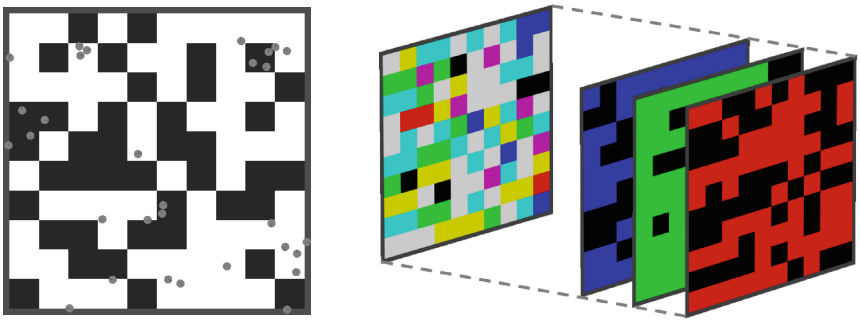

Fig. 2. Left: Robots are exploring the single-feature environment. Right: Generating multi-feature environment by overlaying three different single-feature layers. Colors are combined according to the standard RGB color model.

We adopt the simulator for the Kilobots [22] extended by Ebert et al. [7]. In this simulator, the Kilobot robot is a miniature mobile robot with a circular body of diameter 33 mm, having a linear moving speed of 33 mm/s and a turning speed of 0.2 π rad/s. The Kilobot is equipped with different light sensors that can detect different features, but it can only detect one feature at a time. The detection range of its sensors is limited to the point of its location and the communication range is limited to 3 bodylength. To determine the values of features, the robots walk randomly to explore and share knowledge through transmitting messages to neighbors, but the maximum transmission rate is limited to 10 messages/s. The goal of the swarm is to reach a consensus on the value of every feature efficiently.

For the scenarios, Ebert et al. [7] propose a decentralized collective decision-making algorithm and a dynamic task allocation strategy that allow the robots to reach common decisions on features in a finite time. In their approach (classical approach), while keeping walking randomly in the environment, the robots alternatively explore the environment to estimate the features and share the exploration results with local neighbors. Based on the shared knowledge from neighbors, each robot makes decisions on the features. As to the task allocation for different features, the robot judges the difficulty of

each feature according to the shared knowledge and dynamically selects an appropriate time to switch to the target feature. The experimental results show that when robots switch to the least certain feature every time before exploring the environment, there is an advantage in decision-making time.

4 Hashgraph-Based Knowledge Sharing Approach

Our hashgraph approach builds on the work of Ebert et al. [7], and the biggest difference is that we use hashgraph as the knowledge sharing medium of the mobile robot swarm. We first prove that the hashgraph consensus can be reached under the condition of a mobile network environment, then describe the hashgraph approach in two cases of single-feature and multi-feature scenarios.

4.1 Enhanced Hashgraph in the Mobile Network Environment

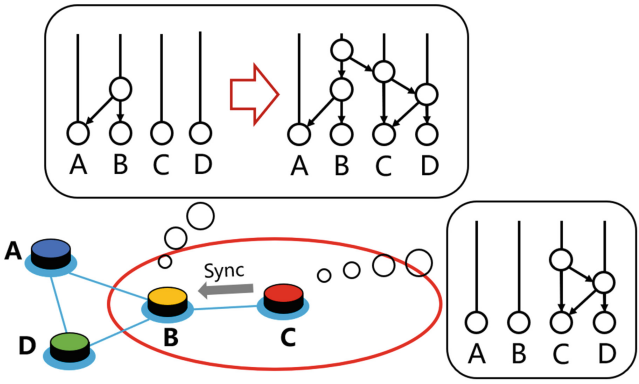

Fig. 3. The red circle represents the communication range of Robot C, while A and D are out of the communication range. Therefore, C can only synchronize the hashgraph with B, and B creates a new event to record this synchronization. (Color figure online)

In common hashgraph, a robot randomly chooses another one to synchronize the hashgraph. But in the mobile network environment of a robot swarm, we stipulate that a robot can only choose its neighbors, as shown in Fig. 3. Therefore, the synchronization process of the hashgraph approach is the same way that an epidemic is propagated throughout a group of individuals. How long it takes for the infection to reach every process has been considered in the "infect forever" model [8,21]:

$$R = log_{f+1}(n) + \frac{1}{f} log(n) + O(1) \tag{1}$$

In the equation, R represents the number of rounds necessary to infect the entire system, f represents the number of other nodes that each infectious node can contaminate in each round, n represents the total number of nodes in the system. Because the value of f and n are both greater than 0, the value of R is finite. Therefore, under such conditions, hashgraph can still be synchronized to each member of the swarm in a finite time. As the hashgraph grows, each node can finally reach the hashgraph consensus.

4.2 Hashgraph Approach in the Single-Feature Scenario

The process of a robot exploring the area is divided into two states: observation and dissemination. The robot first comes into the observation state, the duration of which is 60 s. In this state, the robot walks randomly to explore the feature and calculate the time spent in staying in black (t_b) and light (t_l) color, and receive messages from neighbors. At the end of this state, the robot takes the color corresponding to the larger value in t_b and t_l as its current estimate and updates its current confidence c by calculating the ratio between the time spent in the color of its current estimate and the sum of t_b and t_l. Then the robot turns into the dissemination state.

The duration of dissemination state is $120 \times c$ s, which means that the more confident the robot is in its estimate, the longer the dissemination time will be. In this state, besides receiving messages from others, each robot sends messages to its neighbors. The message contains its ID, estimate, and belief. At the end of each dissemination state, the value of belief b is updated based on the messages received in the past 180 s. In the values of estimates of these messages, the robot adopts the majority as its belief or selects a random belief if the count of each is equal. After this, the robot goes back to the observation state again and so on.

During both states of a robot, once it receives a message from its neighbor, which means they are in communication range, it will synchronize hashgraph with this neighbor and bundle the received message into the newest event at the same time, as shown in Fig. 3. They exchange knowledge and share the same copy of hashgraph in this way. As the hashgraph grows, each robot will judge if consensus has been reached with the virtual voting protocol. Once reaching a consensus on a group of events, each robot will make a decision $d = j$ on the feature if the data of these events satisfies the following two conditions:

$$\begin{cases} |N_1^e - N_0^e| > n \\ N_j^b > 0.5n \end{cases} \tag{2}$$

The variable n means the number of robots in the swarm; the variable N_1^e means the number of estimates of value 1 for the feature, and so on for N_0^e; the variable N_j^b means the number of robots whose newest belief is equal to j (j equals 0 or 1) for the feature. Once the robot makes the decision d, its belief b will be fixed to be d. The two formulas demand that both the number of estimates and that of the beliefs are above the corresponding thresholds respectively.

4.3 Hashgraph Approach in the Multi-feature Scenario

The algorithm in this scenario extends that for the single-feature scenario. Each robot maintains beliefs and makes decisions on each of the features with the same algorithm as in the single-feature scenario. The message it sends also contains the value of all the beliefs. Each robot can explore only one feature at a time and no prior knowledge about the difficulties of features is available, which leads to the problem of feature switching. Intuitively, the more difficult a task is, the more robots are required to perform it. In the multi-feature scenario, with the results of hashgraph consensus, the robot can roughly figure out a variable f_i that measures the difficulties of different features $i (i = 1, 2 \cdots M)$:

$$f_i = \begin{cases} 0 & d_i = 0 \; or \; 1 \\ \dfrac{N_i^r + 1}{|N_{i1}^e - N_{i0}^e| + 1} & else \end{cases} \tag{3}$$

As shown in this equation, the variable N_{i1}^e means the number of estimates of value 1 for the feature i, and so on for N_{i0}^e. N_i^r represents the number of robots that are exploring feature i in the recent past. The variable d_i, represents the decision of the robot on feature i and initially equals *null*. If d_i equals 1 or 0, it means the robot has already made a decision on feature i, therefore the corresponding f_i equals 0. A large f_i reflects that the N_i^r robots have no clear distinction in the estimation of the feature i, so feature i requires more robots. A small f_i brings the opposite conclusion. If every f_i equals 0, which means the robot has made decisions on all of the features, there's no need for the robot to switch features; if not, the tasks can be assigned according to the equation below:

$$A_i = \frac{f_i \times n}{\sum_{k=1}^{M} f_k} \tag{4}$$

The variable A_i represents that the number of robots that should be assigned to explore feature i. With this equation, the robots numbered 1 through n get to know which features they are assigned to explore: the robots numbered 1 to A_1 explore feature 1, the robots numbered $A_1 + 1$ to $A_1 + A_2$ explore feature 2, and so on, all the way to the last A_M robots are assigned to explore feature M. Even if the robot has made a decision on some feature, it may still be assigned to explore that feature to help others make decisions, which is the embodiment of self-organizing and collaboration among the swarm.

5 Experiments

We first verify whether the robot swarm in the hashgraph approach can effectively realize consensus achievement and task allocation in single-feature and multi-feature scenarios. Then, the effects of the moving speed and communication range of the robot on the consensus time of the hashgraph approach are tested. Finally, we compare the resource consumption of hashgraph and traditional blockchain approaches.

5.1 Single-Feature Experiments

Several simulation experiments are carried out to compare the efficiency of classical and hashgraph approaches under different fill ratios. 30 robots are assigned to explore the single-feature environment. The fill ratio of the color white in the experimental area equals r, which ranges from 0.53 to 0.9, while that of black correspondingly equals $1 - r$, ranging from 0.47 to 0.1. For each r value, we run 10 simulations and take the average (see Fig. 4).

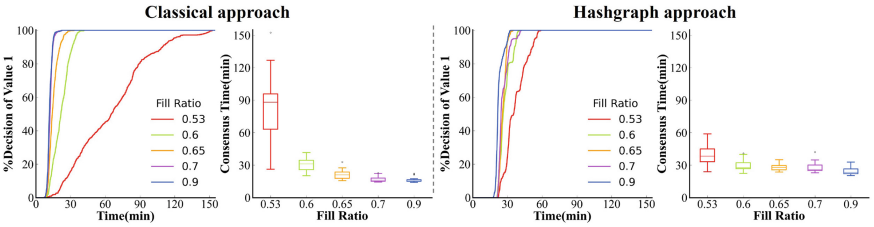

Fig. 4. Single-feature experiments: the percentages of robots that make decisions of value 1 as a function of time, and the time of reaching a consensus on the feature, over different fill ratios. Left: classical approach. Right: hashgraph approach.

In both approaches, the higher the fill ratio, the faster they make decisions, and they reach a consensus on the right decision over all of the fill ratios. In the results of the classical approach, members of the swarm start to make decisions at about 8 min and reach a consensus in 150 min, while the members of the hashgraph approach start to make decisions at about 23 min and reach a consensus in 60 min.

The experiments prove that the robots can reach consensus via hashgraph in the mobile network environment of swarm robots. In the hashgraph approach, because the robots make decisions based on the hashgraph consensus, which is delayed by the processes of gossip about gossip and virtual voting, the robots start to make decisions at a later time, but the curves of decision making rise at a faster slope than that of the classical approach, and finally, converge to the top in 60 min. Compare the box plots of the two approaches, we find that when the fill ratio is less than 0.6, robots of the hashgraph approach can reach a consensus faster than that in the classical approach, while that when the fill ratio is greater than 0.65 comes to the opposite conclusion. On the whole, the consensus time of the hashgraph approach is relatively stable, because its efficiency is more affected by the time to reach a hashgraph consensus, which is relatively fixed, while the consensus time of the classical approach is greatly affected by the fill ratio.

5.2 Multi-feature Experiments

In this subsection, we compare the effectiveness of the robots switching features in the hashgraph approach with that in the classical approach. We set the three features to red, green, and blue, and to distinguish the three features, we set the fill ratios of the features to 0.55, 0.8, and 0.65 respectively, as shown in the right graph of Fig. 2. In the

classical approach, the feature switching strategy that robots switch to the least certain feature before each observation state shows an advantage over other strategies, and we compare the hashgraph approach with this strategy. 30 robots are assigned to explore the three features and there are four different initial distributions: equally distributed and all distributed to one of the three features. We run 10 simulations for each of them. The results are shown in Figs. 5 and 6.

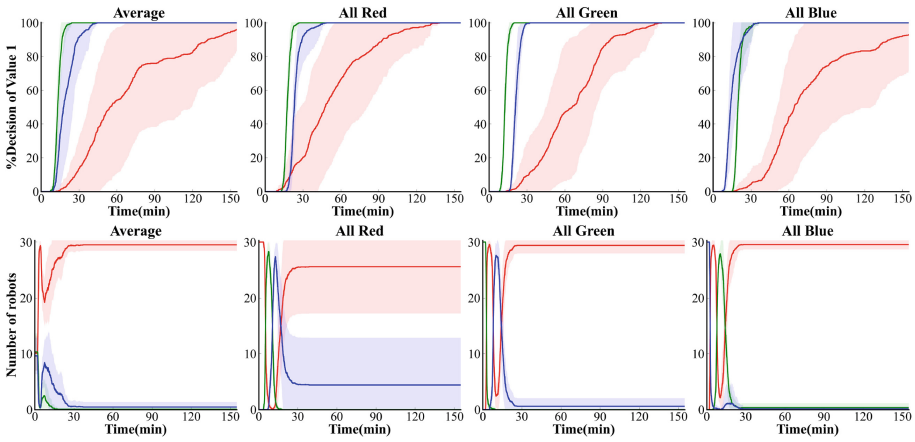

Fig. 5. Multi-feature experiments based on classical approach: the percentages of robots making decisions of value 1 on three features and the distribution of robots on three features respectively as a function of time, over different initial distributions (equally distributed, all distributed to red, green, and blue). The shadow represents the standard deviation (Color figure online).

As we can see in the top row of Fig. 5, in the classical approach, the robots start to make decisions early and reach consensuses quickly on two relatively simple features: green and blue, then all of them switch to explore the hardest feature: red. But their decision making converges slowly, and in some cases, they even can't reach an agreement on the feature of red in 150 min. In the top row of Fig. 6, the robots in the hashgraph approach start to make decisions at a relatively late time, but the curves rise at faster slopes and finally, they reach consensuses on all of the three features in 120 min.

The bottom rows of both Figs. 5 and 6 show the distribution of robots on three features over time. In the classical approach, from the final results, we can see that each robot switches feature efficiently: more robots are assigned to the harder feature. However, there is a problem of over-adaption with the approach. The curves rise and fall frequently and the ranges are large, which means that many robots switch between the three features repeatedly, leading to a waste of time and battery resources in practice. Without a central controller, robots switch features based on their local collected knowledge and follow the local switching strategy, which makes them unable to form a joint force as a whole.

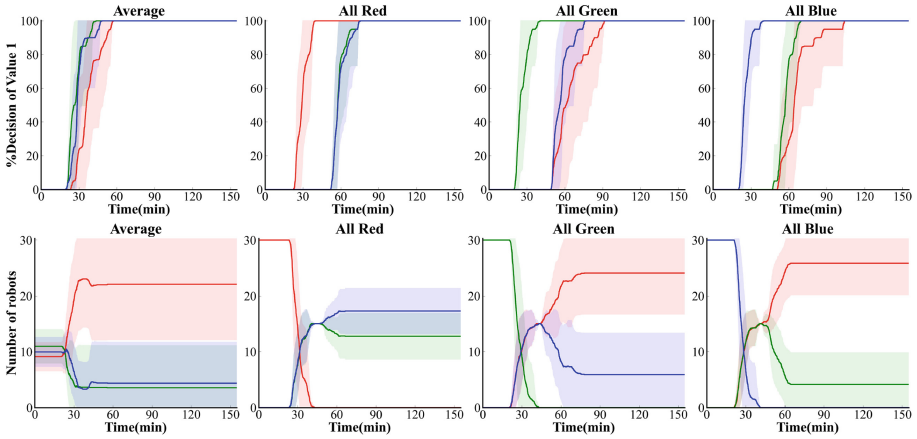

Fig. 6. Multi-feature experiments based on hashgraph approach: the percentages of robots making decisions of value 1 on three features and the distribution of robots on three features respectively as a function of time, over different initial distributions (equally distributed, all distributed to red, green, and blue). The shadow represents the standard deviation (Color figure online).

On the contrary, the curves of the hashgraph approach are much stable and fluctuate in a small range. Take the picture on the right of the bottom row in Fig. 6 as an example, all the robots are initially distributed to blue, after reaching the first round of hashgraph consensus, they switch to the other features. Instead of switching to the same feature, they are assigned equally to two features, then they adjust the tasks according to the results of the next round of hashgraph consensus, realizing efficient task allocation in a fully distributed way. Despite the limited communication range and dynamic network topology, hashgraph can still help robots achieve common knowledge to coordinate their actions well on the overall level.

5.3 Experiments on Consensus Time of Hashgraph Approach

We adopt 30 robots to explore the single-feature scenario to evaluate the performance of the hashgraph approach under different conditions, which is measured by the time of the robot swarm reaching a common decision on the feature. The fill ratio of the color white is fixed at 0.65, the communication range is set to 100, 300, 600, and 1000 mm, while the moving speed is set to 16, 32, and 48 mm/s. We run 10 simulations and take the average (see Fig. 7) for each condition.

As can be seen from Fig. 7, in the case of the same communication range, the greater the moving speed, the shorter the time to reach a consensus, but the difference is relatively small. On the contrary, the communication range has a great impact. When the communication range is limited to 100 mm, the swarm can reach a consensus in more than 20 min. While if the communication range reaches 1000 mm, which means that robots can reach almost every member of the swarm, the time is less than 2 min. The efficiency of the hashgraph approach tested above is also limited by the maximum transmission rate and the observation state during which the robots don't send messages.

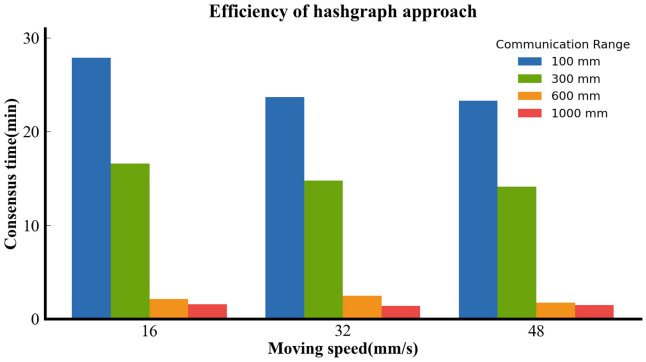

Fig. 7. Experiments on the consensus time of hashgraph approach: time to reach consensus on the feature of the environment under different communication ranges and moving speeds.

By comparison, a wider communication range means a shorter consensus time in the hashgraph approach, while in the traditional blockchain approach, the consensus time depends more on the speed of mining new blocks, and the communication range affects the broadcasting speed of newly mined blocks.

5.4 Experiments of CPU Utilization

In this subsection, we investigate the resource consumption of blockchain and hashgraph approaches by counting CPU utilization. The blockchain approach is derived from Strobel et al.'s work [25], which is based on Ethereum. In this approach, the robots keep mining blocks locally and store the received messages in the newly mined blocks. Blockchains will be synchronized through local interactions, helping the robots achieve common knowledge. The mining difficulty is set to a fixed value of 10^6, which is the same as in [25].

The experiments are carried on a personal computer furnished with an Intel processor of 8 cores, having 3.40 GHz speed and 8 GB RAM. We run 10 simulations and take the average for each approach (see Fig. 8).

As shown in Fig. 8, the blockchain approach consumes more resources than the hashgraph approach. The CPU utilization of the hashgraph approach keeps at about 80% from beginning to end. The CPU utilization of the blockchain approach starts at about 80% and keeps for about 40 min, during which robots start geth processes [25] to register to blockchains, explore the environment, and mine new blocks. After that, the curve rises rapidly and reaches its top, about 350%, which is caused by two factors: mining new blocks and dealing with forks. The mining difficulty is set relatively simple to adapt to the limited computational power of robots, but this leads to the emergence of many forks. Robots have to mine new blocks to restore the information in the forks to the mainchain of blockchains, which requires extra computation.

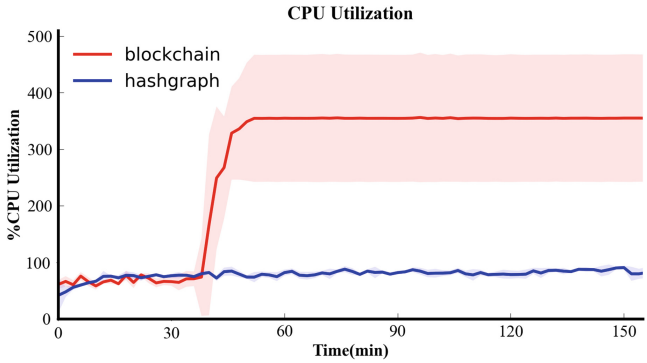

Fig. 8. CPU utilization of 30 robots based on blockchain approach and hashgraph approach as a function of time. The shadow represents the standard deviation.

6 Conclusions and Future Work

To address the shortcomings of the approach based on traditional blockchains in achieving common knowledge among mobile robot swarm, we propose hashgraph as the knowledge sharing medium for the mobile robot swarm in this paper. We describe the hashgraph approach and enhance the gossip rule of hashgraph, and then test the efficiency of the hashgraph approach in the motivated scenarios of single-feature and multi-feature collective decision making. With the enhanced consensus algorithms of hashgraph, we demonstrate that the hashgraph approach can efficiently help robot swarms reach consensuses and realize task allocation. The results show that compared with the blockchain approach, the hashgraph approach requires less computational power and has higher resource utilization.

As a private chain, the application scope of hashgraph is limited by its strict access mechanism. In addition, according to the hashgraph consensus algorithm, although it's easy to create events, the process of virtual voting after each round is relatively long, reducing the efficiency of reaching a consensus. In the future, we are going to verify our approach on physical robots, and consider adding Byzantine robots to test whether hashgraph can deal with malicious attacks in robot swarms. Besides, we will investigate the robustness of hashgraph in the case of a large number of failures of robots.

Acknowledgments. This work is partially supported by the major Science and Technology Innovation 2030 "New Generation Artificial Intelligence" project 2020AAA0104803 and Scientific Research Plan of National University of Defense Technology under Grant No. ZK-20-38.

References

1. Alhafnawi, M., Hauert, S., O'Dowd, P.: Self-organised saliency detection and representation in robot swarms. IEEE Robot. Autom. Lett. **6**(2), 1487–1494 (2021). https://doi.org/10.1109/LRA.2021.3057567

2. Alsamhi, S.H., Lee, B.: Blockchain-empowered multi-robot collaboration to fight COVID-19 and future pandemics. IEEE Access **9**, 44173–44197 (2021). https://doi.org/10.1109/ACCESS.2020.3032450
3. Baird, L.: The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance
4. Berman, S., Halász, Á., Hsieh, M.: Ant-inspired Allocation: Top-Down Controller Design for Distributing a Robot Swarm Among Multiple Tasks, pp. 243–274. CRC Press, Boca Raton (2016)
5. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002). https://doi.org/10.1145/571637.571640
6. Correll, N., Martinoli, A.: Modeling and designing self-organized aggregation in a swarm of miniature robots. Int. J. Robot. Res. **30**(5), 615–626 (2011)
7. Ebert, J.T., Gauci, M., Nagpal, R.: Multi-feature collective decision making in robot swarms. In: AAMAS 2018, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 1711–1719 (2018)
8. Eugster, P.T., Guerraoui, R., Kermarrec, A.M., Massoulie, L.: From epidemics to distributed computing. IEEE Trans. Comput. **37**, 2004 (2004)
9. Castelló Ferrer, E.: The blockchain: a new framework for robotic swarm systems. In: Arai, K., Bhatia, R., Kapoor, S. (eds.) FTC 2018. AISC, vol. 881, pp. 1037–1058. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-02683-7_77
10. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols. In: Joint Working Conference on Secure Information Networks: Communications and Multimedia Security (1999)
11. Jamshidpey, A., Afsharchi, M.: Task allocation in robotic swarms: explicit communication based approaches. In: Barbosa, D., Milios, E. (eds.) CANADIAN AI 2015. LNCS (LNAI), vol. 9091, pp. 59–67. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18356-5_6
12. Lafferriere, G., Williams, A., Caughman, J., Veerman, J.: Decentralized control of vehicle formations. Syst. Control Lett. **54**(9), 899–910 (2005)
13. Lamport, L.: Paxos made simple. In: ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001), pp. 51–58 (2001)
14. Li, J., Wu, J., Li, J., Bashir, A.K., Piran, M.J., Anjum, A.: Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks. IEEE Commun. Mag. **59**(7), 94–100 (2021). https://doi.org/10.1109/MCOM.001.2000419
15. Lumelsky, V., Harinarayan, K.: Decentralized motion planning for multiple mobile robots: the cocktail party model. Auton. Robot. **4**, 121–135 (1997). https://doi.org/10.1023/A:1008815304810
16. Majercik, S.M.: Initial experiments in using communication swarms to improve the performance of swarm systems. In: Kuipers, F.A., Heegaard, P.E. (eds.) IWSOS 2012. LNCS, vol. 7166, pp. 109–114. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28583-7_12
17. Mocanu, A., Bădică, C.: Bringing Paxos consensus in multi-agent systems. Association for Computing Machinery (2014)
18. Moussa, M., Beltrame, G.: On the robustness of consensus-based behaviors for robot swarms. Swarm Intell. **14**, 205–231 (2020). https://doi.org/10.1007/s11721-020-00183-1
19. Teslya, N., Smirnov, A.: Blockchain-based framework for ontology-oriented robots' coalition formation in cyberphysical systems. MATEC Web Conf. **161**(9), 03018 (2018)
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf
21. Pittel, B.: On spreading a rumor. SIAM J. Appl. Math. **47**(1), 213–223 (1987). https://doi.org/10.1137/0147013
22. Rubenstein, M., Ahler, C., Nagpal, R.: Kilobot: a low cost scalable robot system for collective behaviors. In: 2012 IEEE International Conference on Robotics and Automation, pp. 3293–3298 (2012). https://doi.org/10.1109/ICRA.2012.6224638

23. Schranz, M., Umlauft, M., Sende, M., Elmenreich, W.: Swarm robotic behaviors and current applications. Front. Robot. AI **7**, 36 (2020)
24. Shi, P., Wang, H., Yang, S., Chen, C., Yang, W.: Blockchain-based trusted data sharing among trusted stakeholders in IoT. Soft. Pract. Exp. (2019). https://doi.org/10.1002/spe.2739
25. Strobel, V., Castelló Ferrer, E., Dorigo, M.: Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to byzantine robots. Front. Robot. AI **7**, 54 (2020). https://doi.org/10.3389/frobt.2020.00054
26. Strobel, V., Castelló Ferrer, E., Dorigo, M.: Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In: AAMAS 2018, International Foundation for Autonomous Agents and Multiagent Systems, pp. 541–549 (2018)
27. Valentini, G., Brambilla, D., Hamann, H., Dorigo, M.: Collective perception of environmental features in a robot swarm. In: Dorigo, M., et al. (eds.) ANTS 2016. LNCS, vol. 9882, pp. 65–76. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44427-7_6
28. Wang, Q., Mao, X., Yang, S., Chen, Y., Liu, X.: Grouping-based adaptive spatial formation of swarm robots in a dynamic environment. Int. J. Adv. Robot. Syst. **15** (2018). https://doi.org/10.1177/1729881418782359
29. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014)