Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8834879, 11 pages
https://doi.org/10.1155/2021/8834879

Research Article

Min-k-Cut Coalition Structure Generation on Trust-Utility Relationship Graph

XiangLong Kong,[1] **XiangRong Tong**,[1,2] **and YingJie Wang**[1,2]

[1]*School of Computer and Control Engineering, Yantai University, Yantai 264005, China*
[2]*Yantai Key Laboratory of High-End Ocean Engineering Equipment and Intelligent Technology, Yantai University, Yantai 264005, China*

Correspondence should be addressed to XiangRong Tong; txr@ytu.edu.cn

Received 28 August 2020; Revised 8 October 2020; Accepted 13 March 2021; Published 14 April 2021

Academic Editor: Maode Ma

Trust relationships have an important effect on coalition formation. In many real scenarios, agents usually cooperate with others in their trusted social networks to form coalitions. Therefore, the trust value between agents should constrain the utility of forming coalitions when cooperating. At the same time, most studies ignore the impact of the number of coalitions in coalition structure. In this paper, the coalition formation of trust-utility relationship in social networks is researched. Each node represents an agent, and the trust-utility networks that connect the agents constrain coalition formation. To solve the task assignment problem, this paper proposes a greedy algorithm which is based on the edge contraction. Under the premise of ensuring the agent's individually rationality, this algorithm simulates the formation process of coalitions between agents through continuous edge contraction and constrains the number of forming coalitions to k to solve the problem of coalition structure. Finally, the simulation results show that our algorithm has great scalability because of the ability of solving the coalition structure on a large-scale agent set. It can meet the growing demand for data intensive applications in the Internet of things and artificial intelligence era. The quality of the solution is much higher than other algorithms, and the running time is negligible.

1. Introduction

Coalition formation (CF) is an important research area in Multiagent Systems (MAS) [1]. It is known that combining several agents together can achieve goals that every individual cannot achieve alone, or get better utility [2]. Most of CF researches are based on models in the field of cooperative game theory, e.g., characteristic function game [3, 4].

Although these models can be used in the study of coalition formation, they are usually too abstract to be used in the actual cooperation scenarios. Most of them use graphs to represent the constraints of the relationship between agents [5–7], and each node in the graph represents an agent, while the edge represents the relationship between agents. Agents can form coalitions only when they are the vertices of the connected subgraphs in a graph, which is called the problem of *Graph Coalition Structure Generation* (GCSG); that is, in the process of Coalition Structure Generation (CSG), the connectivity between agents that make up the coalition is required. Such constraints exist in most real scenarios, such as social trust constraints, physical constraints [8], and wireless communication constraints [9]. The proposed algorithm in this paper uses trust constraint; agents usually look for others with high trust and utility to cooperate, which is suitable for most scenarios. For example, in the field of privacy protection [10–14], if the trust value or utility between agents is higher, the more privacy data the agents know about each other. Then, the agents in the coalition will not disclose the information between each other and play the role of privacy protection.

In addition, most previous studies about the GCSG problem assumed that agents can form any number of coalitions and failed to consider to bind the number of coalitions in a coalition structure. However, in many real scenarios, the number of coalitions should be constrained. For example, in the field of task assignment [15, 16], agents in MAS may

need to cooperate to complete all tasks. Since the resources or capabilities of a single agent to perform tasks are limited, it is necessary to assign tasks to the coalition. This requires to constrain the number of coalitions consistent with the number of tasks, so as not to waste resources or fail to complete tasks. In addition, when performing tasks, agents usually do not cooperate with unfamiliar agents, but they can establish contact with other team members through the introduction of trusted agents.

To solve the above problems, we develop a solution for the GCSG problem in the context of task assignment and provide a model which can be used to solve the problem of a large number of agents in the real world. We use trust and utility to discuss CF, where the weight of the edge represents the strength of the cooperative relationship between agents, and its weight is represented by two tuples, which are the trust value and utility value, respectively. A connection between disconnected agents can be established through trust transfer. Based on edge contraction, this paper proposes an *Edge Contraction Coalition Formation* (ECCF) algorithm to solve the GCSG problem. This algorithm refers to the min-k-cut algorithm, constrains the number of formed coalitions to k, and generates the coalition structure under the set conditions. The process of coalition formation between agents is simulated by continuous edge contraction. Each edge contraction process includes the following: (i) delete an edge from the graph, and (ii) merge two nodes previously connected by this edge. Merging two nodes is corresponded to merging two coalitions represented by these nodes, from two coalitions to one coalition, to ensure the generation of a coalition structure with exact k coalitions. The ECCF algorithm is an approximate algorithm with $O(n^2e)$ time complexity. In addition, the ECCF algorithm is a greedy algorithm essentially, which can quickly find the feasible solution of the set coalition structure in polynomial time. Finally, ECCF can be applied in large-scale systems. It can well meet the demand of the growing data intensive applications in the Internet of things and artificial intelligence era. The simulation results show that the ECCF algorithm can always find a feasible coalition structure in a short time.

The main contributions of this paper are summarized as follows:

(1) Trust is transitive for agents, and each agent is independent. The *six degrees of separation* and *trust transfer* are used to establish the trust relationship, so as to obtain a trust-utility relationship graph between agents, which can be applied in large-scale agent sets in most real scenarios

(2) In order to solve the problem of task assignment, we propose a new polynomial time complexity algorithm ECCF, which is based on edge contraction and uses the min-k-cut method to segment the trust-utility network graph and generate the coalition structure. Under the condition of ensuring the agent's individual rationality and generating accurate k coalitions, the coalition structure with k coalitions in the given situation is quickly generated

(3) The simulation results demonstrate the efficiency and runtime of ECCF algorithm. The ECCF algorithm can be applied in large-scale MAS, and its solution quality is much higher than other algorithms; furthermore, its runtime is negligible

The rest of the paper is organized as follows. Section 2 discusses the relationship between our work and existing literature, and Section 3 introduces some Preliminary Definitions. Section 4 formally defines the ECCF, and Section 5 discusses our empirical evaluation. Finally, Section 6 concludes the paper.

2. Related Work

In this section, we will discuss the CF, graph theory techniques, and trust transitivity, respectively.

2.1. Coalition Formation. Many algorithms have been developed to solve traditional CSG problems. Yeh [17] first proposed the method based on Dynamic Programming (DP) to solve the CSG problem. The exact coalition structure can be obtained through DP. Rahwan et al. [4] developed an anytime algorithm based on the integer partition (IP). Michalak et al. [18] combined the IP algorithm and the DP method to improve the classical CSG algorithm ODP-IP. However, the traditional solutions can only process dozens of agents. To improve the scalability of the algorithm, Ueda et al. [19] have examined alternative function representations, which can reduce the computational complexity of the associated CF problem. Wu et al. [20, 21] sampled the coalition structure graph based on the Monte-Carlo tree search method and iteratively expanded the search tree and proposed a scalable anytime method called CSG-UCT. The CSG-UCT algorithm needs to be performed once for each iteration. The Monte-Carlo tree search process requires a large number of iterations in the process of finding the optimal solution. When the number of agents increases, the time required is greater. However, these models may not be applicable to the real world, such as the scenario in this paper that agents connect through the trust network; that is, they cannot be applied to the GCSG problem.

In this context, graph-based cooperative game has attracted the attention of researchers in various fields [22–25]. Voice et al. [26] proposed an algorithm for GCSG. However, the assumption made by voice can only be applied to the characteristic function of the independent disconnected members (IDM) property. It was hard to be applied in most real scenarios. Filippo et al. [27] proposed a branch and bound algorithm CFSS (Coalition Formation for Sparse Synergies), which can solve the problem of GCSG and provide anytime approximate solution of quality guarantees. However, all of these works assumed that any number of coalitions can be formed, and there is no constraint on the number of coalitions in the coalition structure.

2.2. Graph Theory Techniques and Trust Transitivity. The ECCF algorithm forms the coalition structure of agent set through edge contraction and constrains the number of coalitions. This graph theory technology and is famous for its

application in the algorithm of solving minimum cut problem [28]. Filippo et al. [27] for the first time applied edge contraction to enumerate feasible coalition structure and proposed a CFSS algorithm. At the same time, the search tree is used to represent the search space of the coalition structure. The tree generated by the search space contains all coalition structures on the graph; a CFSS algorithm was proposed, which can prune the search space and solve the coalition structure with quality guarantees at any time when it is used to solve the $m + a$ function coalition structure. However, this algorithm is only suitable for solving the $m + a$ function coalition structure, and it can not be used in the context of task assignment.

In the CF game, under the background of task assignment, a social network is mostly used as the constraint of cooperation between agents, which requires connectivity between agents that make up the coalition. Sless et al. [29] avoided this constraint by setting the organizer. Organizers are required to have the ability to promote the connection between unrelated agents. Organizers can add edge sets to social networks to connect unrelated agents, which depends on the social status of organizers and their own experience.

In MAS, the research on trust attracted increasing attention of researchers [30–32]. When two parties cooperate, trust is a subjective evaluation of one party to the other. Trust is not only subjective but also asymmetric and transitive. Trust is an important factor in decision-making when an agent chooses who to cooperate with to complete the task. Generally, the higher the trust level, the easier it is to generate a long-term stable cooperative relationship. Trust relationship depends on the network to spread. In the real world, the transitivity of trust can make the unfamiliar agents communicate and promote the cooperation between agents. In this way, in the process of CSG, there is a relationship between agents. At the same time, any coalition generated also has connectivity. Thus, our approach establishes a connection between disconnected agents through trust transfer and then solves the set coalition structure through edge contraction.

3. Preliminary Definitions

Let $G = (A, E, \rho, \omega)$ represent a directed weighted graph, where $A = \{a_1, a_2, \cdots, a_n\}$ is a finite, non-empty set of agents. Each edge $(a_i, a_j) \in E$ has two weights ρ and ω, which represent trust and utility relationship between agents, respectively. $\rho_{i,j}$ represents the trust value of agent a_i from a_j, and $\omega_{i,j}$ represents the utility value of agent a_i from a_j. Agents without edge connection can get the trust value and utility value through transfer calculation.

Let $f_{\rho,\omega}(a_i, a_j)$ represent the trust-utility relationship function obtained by considering the trust relationship and utility relationship between agent a_i and agent a_j, which represents the ability of smooth cooperation between agents. $f_{\rho,\omega}(a_i, a_j)$ could be expressed by

$$f_{\rho,\omega}(a_i, a_j) = \frac{P_{i,j}\Omega_{i,j} + P_{j,i}\Omega_{j,i}}{2}. \tag{1}$$

where P and Ω are the trust relationship and utility relationship between agents after transmission. The trust-utility relationship can be used to get the trust-utility relationship graph between agents $G_f = (A, E, f)$. Only the trust relationship between agents is considered when ω is zero or the same between agents in cooperation. Only the utility relationship is considered in turn when ρ is zero or the same between agents in cooperation.

Let $\mu(a_i, C)$ denote the utility $a_i \in C$; its value is the sum of the utility among a_i and other members in C. It is shown by

$$\mu(a_i, C) = \sum_{a_i \in C} \omega(a_i, a_j). \tag{2}$$

It is denoted as μ_i. The utility of an agent is given by other agents who have trust-utility relationship with him in the coalition. The utility of a coalition is composed of the utility of all agents in the coalition. $V(C)$ is used to represent the value of $C \in A$ in the coalition, and the value is defined as the values of all agents in C. $V(C)$ is obtained by

$$V(C) = \sum_{a_i \in C} \mu(a_i, C). \tag{3}$$

Assume that the required number of tasks to be completed by agents is k, $0 < k < n$. $S_k(A)$ indicates that the agent set A is divided into a set of partitions containing k nonempty subsets. Each case in $S_k(A)$ is called the coalition structure, which is denoted as CS.

$\gamma_{CS} : A \longrightarrow \mathrm{CS}$ represents the mapping function of the agent set to its coalition, i.e., $\mathrm{CS} \in S_k(A)$, $C_i \in \mathrm{CS}$, and $a_i \in C_i$, then $\gamma_{\mathrm{CS}}(a_i) = C_i$. The utility of agents in different coalition structures is shown by Figure 1. The social utility of coalition structure CS is defined as SW(CS), which is shown by

$$\mathrm{SW}(\mathrm{CS}) = \sum_{C \in CS} V(C). \tag{4}$$

Definition 1 (*block*). Given a coalition structure $\mathrm{CS} = \{C_1, C_2, \cdots, C_k\}$, $\exists B \in A$, if $\forall a_i, a_j \in B$, $\forall a_i{}' \in \gamma_{CS}(a_i)$, $\forall a_j{}' \in \gamma_{CS}(a_j)$ satisfied $f_{\rho,\omega}(a_i, a_j) > f_{\rho,\omega}(a_i, a_i{}')$ and $f_{\rho,\omega}(a_i, a_j) > f_{\rho,\omega}(a_j, a_j{}')$; then, a coalition B is *block*.

According to Definition 1, for any two agents in a coalition, if there is another coalition B containing the two agents and the rewards of agent in B are greater than the rewards of the current coalition, that is, these two agents tend to leave the current coalition, and B is called *block*. Clearly, *block* B is an unstable factor for CS.

From Figure 2, we can see that if there is a coalition structure $\mathrm{CS} = \{\{a_1, a_2, a_3\}, \{a_4\}, \{a_5, a_6\}\}$, a coalition $B = \{a_4, a_5\}$. According to Definition 1, we can see that coalition B is a *block*.

Note that even if the optimal CS is obtained, there may be *blocks* in the solution. The movement of agents in the original coalition may form any number of coalitions. Clearly, we must exclude this situation in the model. Therefore, a *k -block*

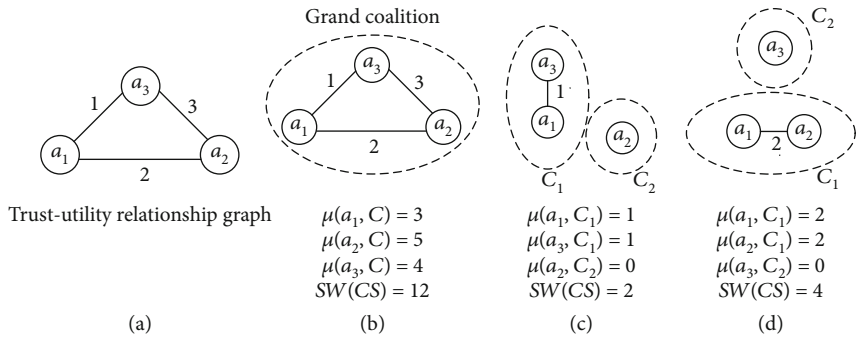


Figure 1: The utility of agents in different coalition structures.

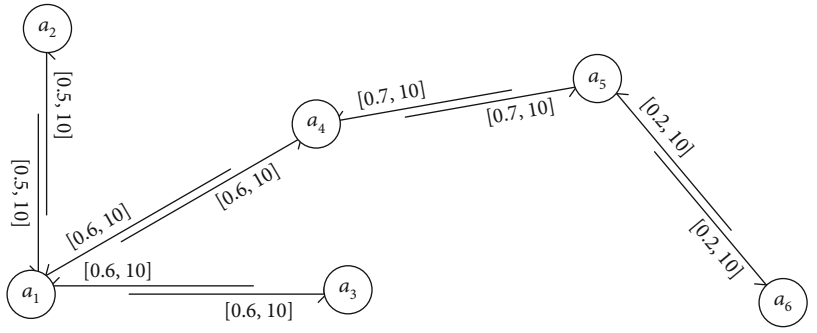


Figure 2: Trust network.

coalition must be defined so that the final coalition structure contains exactly k coalitions.

Definition 2 (*k -block coalition*). Given a coalition structure $CS = \{C_1, C_2, \cdots, C_k\}$, $\exists B \in A$, if $\forall a_i, a_j \in B$, $\forall a_i{}' \in \gamma_{CS}(a_i)$, $\forall a_j{}' \in \gamma_{CS}(a_j)$ can satisfy $f_{\rho,\omega}(a_i, a_j) > f_{\rho,\omega}(a_i, a_i{}')$ and $f_{\rho,\omega}(a_i, a_j) > f_{\rho,\omega}(a_j, a_j{}')$. At the same time, $\exists C_m \in CS$, $C_m \in B$; then a coalition B is *k-block*.

In Figure 2, it is known that there are coalition structure $CS = \{\{a_1, a_2, a_3\}, \{a_4\}, \{a_5, a_6\}\}$ and *block* $B = \{a_4, a_5\}$, $\exists C_m = \{a_4\}$, making $C_m \in CS$ and $C_m \in B$. According to Definition 2, B is a *3-block coalition*.

Definition 2 is an extension of Definition 1. Coalition structure after any agent movement must contain exact k coalitions. Therefore, it is necessary to ensure that agents tend to join the existing coalition; that is, agents with movement tendency in the coalition structure will only join the current coalition to another formed coalition and will not form a new coalition. In addition, it is necessary to ensure that there is only one coalition. If there are multiple such coalitions, the number of coalitions in the generated coalition structure will be less than k. If there is less than one such coalition, the final coalition structure will contain $k + 1$ coalitions. Only if there is one coalition, the movement of agents will guarantee the coalition structure to contain accurate k coalitions.

Definition 3 (*individual rationality*). Given a coalition structure $CS \in S_k(A)$, if each agent gets at least the same utility from CS to its own, then CS satisfies individual rationality, i.e., $\forall a \in C_i$, $i \in (1, \cdots, k)$, $\mu(a_i, C_i) \geq \mu(a, \{a\})$.

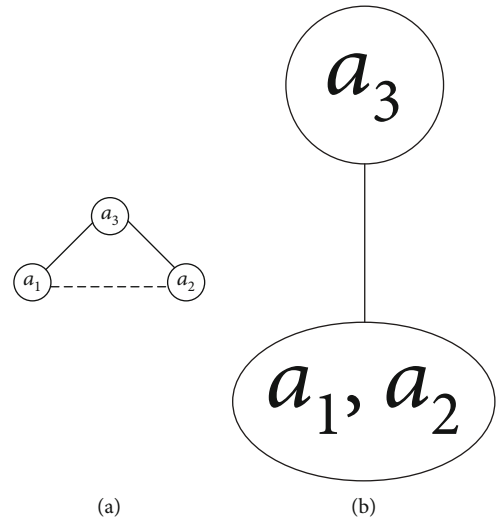


Figure 3: Example of an edge contraction.

Definition 4 (*trust transfer*). Given the trust network, trust can be transferred. As shown in Figure 3, assuming that agent a_1 trusts agent a_2 and agent a_2 trusts agent a_4, it can be determined that agent a_1 may trust agent a_4. There are two ways for *trust transfer*: (i) take the minimum value, and (ii) take the weighted average value. In a large network system, the

Input: Trust network $G = (A, E, \rho, \omega)$, number of coalitions parameter k from CS
Output: Coalition structure CS
1: Calculate the trust and utility after transfer
2: For all agent $a_i, a_j \in A$
3: $P_{i,j} = \min\{\rho_{ij}^{(k)}, k = 2, 3, \cdots, 5\}$.
4: $\Omega_{i,j} = \min\{\omega_{ij}^{(k)}, k = 2, 3, \cdots, 5\}$.
5: End for
6: Calculate the trust and utility relationship and get the trust-utility relationship graph
7: For each edge $(a_i, a_j) \in E$
8: $f_{\rho,\omega}(a_i, a_j) = (P_{i,j}\Omega_{i,j} + P_{j,i}\Omega_{j,i})/2$.
9: End for
10: Perform the edge contraction operation and generate the image segmentation coalition
11: For all Agent $a_i, a_j \in A$
12: For i =1…n-k
13: $(a_i, a_j) \leftarrow \max\{f_{\rho,\omega}(a_i, a_j)\}$
14: $Contract(a_i, a_j)$
15: For all $a \in A$
16: $f_{\rho,\omega}(a, m) = f_{\rho,\omega}(a_i, a) + f_{\rho,\omega}(a_j, a)$
17: End for
18: End for
19: For each edge $(a_i, a_j) \in E$
20: Remove (a_i, a_j)
21: End for
22: End for
23: **return**CS

Algorithm 1: ECCF

weighted average will make the trust value tend to zero, so we take the minimum value in the transfer process, that is,

$$P_{i,j} = \min\left\{\rho_{ij}^{(k)}, k = 2, 3, \cdots, 5\right\}. \quad (5)$$

In the same way, the utility between agents is expressed as follows:

$$\Omega_{i,j} = \min\left\{\omega_{ij}^{(k)}, k = 2, 3, \cdots, 5\right\}. \quad (6)$$

According to the trust and utility function, trust and utility relationship between the agents can be obtained after coherently considering the trust relationship and utility relationship, so as to obtain the trust-utility relationship graph $G_{f=}(A, E, f)$ between the agents, where A represents the agent set, the members are generally a_i, E is the edge set, and each edge e has weight f that represents the trust-utility relationship value between the agents obtained through the trust-utility function.

The trust network in Figure 2 can provide an example of trust transfer. According to the equations, we can get $P_{15} = 0.6$, $\Omega_{15} = 10$, and $f_{15} = 6$.

Definition 5 (*edge contraction*). Given the trust-utility graph $G_f = (A, E, f)$, $(a_i, a_j) \in E$, the result of contraction for e is the graph G_f' obtained by deleting e and corresponding vertices a_i and a_j and, at the same time, adding new vertices $m = a_i \cup a_j$. In addition, each edge connecting with a_i or a_j in G_f will be connected to m in G_f', then merging the parallel edges (i.e., the edges connected to the combined two vertices) and the corresponding weights of parallel edges (i.e., the trust-utility relationship).

Intuitively, each edge contraction merges two vertices, corresponding to merging two coalitions represented by these nodes. Note that the order of the contraction of edges can be in any given order and get the same result, i.e., contracting e first, then contracting e' which produces the same graph to contracting e' then contracting e. The resulting graph after contracting represents a feasible coalition structure, where the coalition corresponds to the vertices of the graph.

When performing edge contraction, the number of subsets of edges in the graph is greater than the number of feasible coalition structures. This redundancy is due to the contraction of any two or three edges resulting in the same coalition structure. In this paper, the edge with the largest weight is firstly selected for contraction operation. When the weight of the edge is different, there will be no redundant operation. When there are multiple edges with the largest weight, any edge with the largest weight will be contracted randomly, and the edge with the largest weight will be also selected for operation in the next contraction. So there will be no redundant coalition structure.

Figure 3 shows the contraction of the edge (a_1, a_2) in the trust-utility graph $G_f = (A, E, f)$, which results in a new

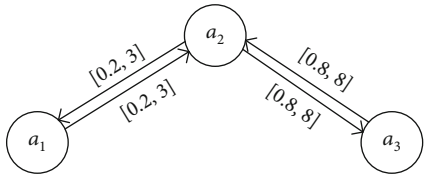


Figure 4: Simple trust network.

Table 1: Initialization trust-utility.

	a_1	a_2	a_3
a_1		[0.2, 3]	
a_2	[0.2, 3]		[0.8, 8]
a_3		[0.8, 8]	

Table 2: Postprocessing trust-utility.

	a_1	a_2	a_3
a_1		[0.2, 3]	[0.2, 3]
a_2	[0.2, 3]		[0.8, 8]
a_3	[0.2, 3]	[0.8, 8]	

vertex $m = a_i \cup a_j$ connected to the vertex a_3 and merges the parallel edges (a_1, a_3) and (a_2, a_3).

4. Maximize the Utility of Coalition Structure

This section mainly discusses the problem of the coalition structure that tries to maximize the utility given a trust-utility relationship graph. Agents in the graph are reasonably assigned to coalitions to maximize the utility of coalition structure.

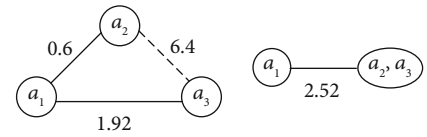
Definition 6 (*k-cut*). $\mathrm{Cut}_k(C_1, C_2, \cdots, C_k)$ means that agents in the trust-utility relationship graph are divided into k disjoint parts according to the trust-utility relationship, where each part forms a coalition, and k coalitions are formed. $\mathrm{Cut}_k(C_1, C_2, \cdots, C_k)$ is called k-cut.

Brânzei and Larson [33] first noted the relationship between the social welfare maximization coalition structure and the min-k-cut of the graph; that is, the coalition structure with the number k of maximizing utility is the partition of the min-k-cut graph. This is reasonable because the sum of the weights of all edges is constant. By minimizing the utility of edges outside the coalition, the utility within the coalition can be maximized. Therefore, the min-k-cut algorithm can be used to find the maximum utility coalition structure. Based on the above researches, Sless et al. [29] proved that min-k-cut is in P for nearly positive graphs and a fixed k. However, the algorithm has high complexity and can only be applied in nearly positive graphs, while our algorithm could solve approximate solutions with low complexity.

Based on the concept of edge contraction in graph theory, an approximate algorithm is proposed. Given k, the edge contraction method is used to perform n - k number of edge contraction. Therefore, it is ended with k nodes. Intuitively, each node after the contraction is a coalition. Therefore, though deleting the remaining edge, it can form k-connected subgraphs, i.e., k coalitions.

Table 3: Trust-utility function postrelationship.

	a_1	a_2	a_3
a_1		0.6	0.6
a_2	0.6		6.4
a_3	0.6	6.4	



(a) Before contraction (b) After contraction

Figure 5: Trust-utility relationship graph.

In each iterative contraction, the edge with the largest weight is selected in the trust-utility graph, and define operation $\mathrm{Contract}(a_i, a_j)$. $\mathrm{Contract}(a_i, a_j)$ means to delete points $a_i$and a_j and edge (a_i, a_j), and add a new nodes $m = a_i \cup a_j$, for any $a \in A$; there is $f_{\rho,\omega}(a, m) = f_{\rho,\omega}(m, a) = f_{\rho,\omega}(a_i, a) + f_{\rho,\omega}(a_j, a)$.

4.1. ECCF. In this section, we give a comprehensive consideration for the trust relationship and utility relationship between the agents in the trust network G; then $P_{i,j}$ and $\omega_{i,j}$ are calculated according to the trust network G. Then, the trust-utility relationship between the agents is calculated according to the values of $P_{i,j}$ and $\omega_{i,j}$. Moreover, the contract of edge is performed according to f to complete the image cutting. Finally, coalition structure is obtained. The specific steps are as shown by Algorithm 1.

Given the trust network $G = (A, E, \rho, \omega)$, the number of vertices is n, the $P_{i,j}$, $\Omega_{i,j}$, and $f_{i,j}(a_i, a_j)$ are the trust relationship, the utility relationship, and the trust-utility relationship after comprehensive consideration, respectively. Firstly, input the trust network and calculate the trust value and utility value between agents according to the trust transfer formula. Secondly, according to the results of the first step, the trust and utility are considered synthetically, and the trust-utility relationship between agents is obtained by using the trust-utility relationship function. Thirdly, according to the trust-utility relationship size, the proposed algorithm is performed, and according to the operation of edge contraction, n - k times edge contraction is performed, then the remaining weight is deleted, and the image is cut. The agent set is divided into k disjoint sets, i.e., k coalitions, and the final CS is obtained.

In each iteration of ECCF algorithm, the edge with the largest trust-utility relationship between agents is selected to contract. The CS obtained by the ECCF algorithm is an approximate solution, which is essentially a greedy algorithm. The ECCF algorithm does not pursue the optimal

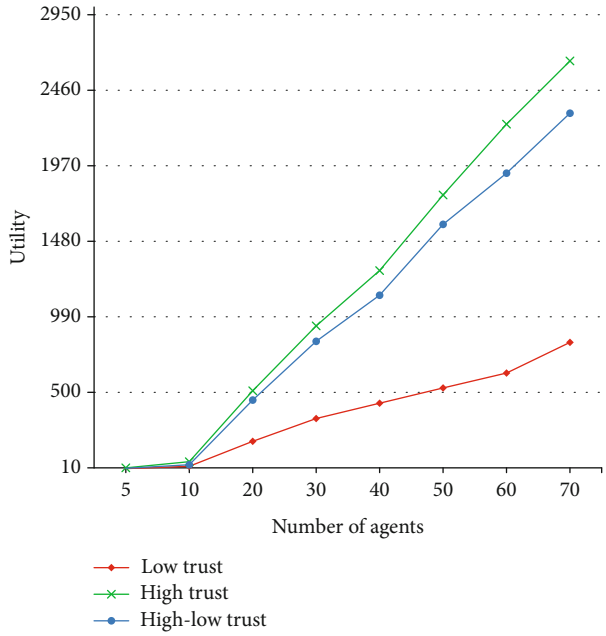


Figure 6: The influence of trust on utility.

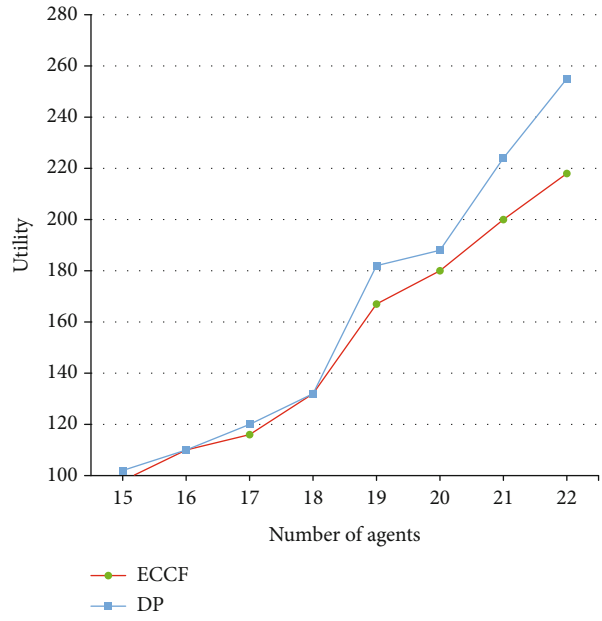


Figure 7: The influence of n on utility.

solution, and backtrack only hopes to get a more satisfactory solution quickly and makes the optimal selection based on the current situation, without considering all possible overall situations. It could save a lot of time to find the best solution.

4.1.1. Time Complexity Analysis. In the ECCF algorithm, the first step uses the trust transfer formula to obtain the trust value and utility value between agents. The time complexity is $O(e)$. In the second step, the trust-utility function formula is used to coherently consider calculate the trust-utility relationship between agents. The time complexity is $O(n^2)$. In the third step, the *for* loop in the algorithm performs the contract operation n - k times. In each contract iteration, the ECCF algorithm is used to traverse all edges in the graph and find the largest weight edge. The time complexity is $O(e)$. Finally, merge parallel edges and the corresponding weights of parallel edges. The complexity is at most $O(2n)$, so the complexity of the third step is $O(n^2\ e)$. The total time complexity of the algorithm is $O(n^2\ e)$.

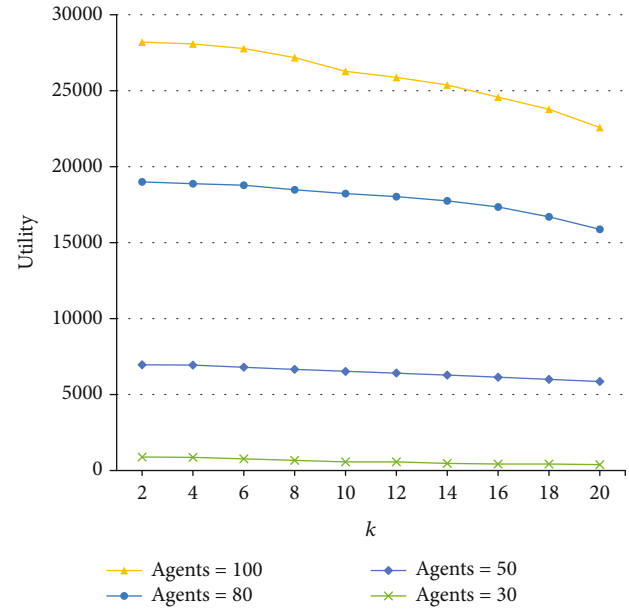


Figure 8: The influence of n and k on utility.

4.1.2. Example of Algorithm Solving Process. Figure 4 shows an ECCF algorithm for a given simple trust network.

According to the ECCF algorithm, the solution process is explained as follows:

In the first step, the settings are initialized, which are shown in Table 1.

In the second step, according to the transfer formula, the trust and utility after processing are shown in Table 2.

In the third step, we use the trust-utility function $f_{\rho,\omega}(a_i, a_j) = (P_{i,j}\Omega_{i,j} + P_{j,i}\Omega_{j,i})/2$ to get the relationship shown in Table 3.

The ECCF algorithm coherently considers the impact of the trust relationship and utility relationship between agents, which is consistent with the real social situation, and obtains the trust-utility relationship graph shown by Figure 5.

In the fourth step, based on the ECCF algorithm, let $k = 2$; we contract the maximum edge (i.e., 6.4) and delete the remaining trust-utility relationship (i.e., 2.52) to get the coalition structure $\{\{a_1\}, \{a_2, a_3\}\}$; then the utility is 12.8.

5. Simulation Experiments and Analysis

5.1. Experimental Environment and Comparison Algorithm. In this section, the simulation experiments are designed to demonstrate the effectiveness of the ECCF algorithm. We use a random graph for experimental data. The experimental environment is Windows 7 with 64 bit operating system,

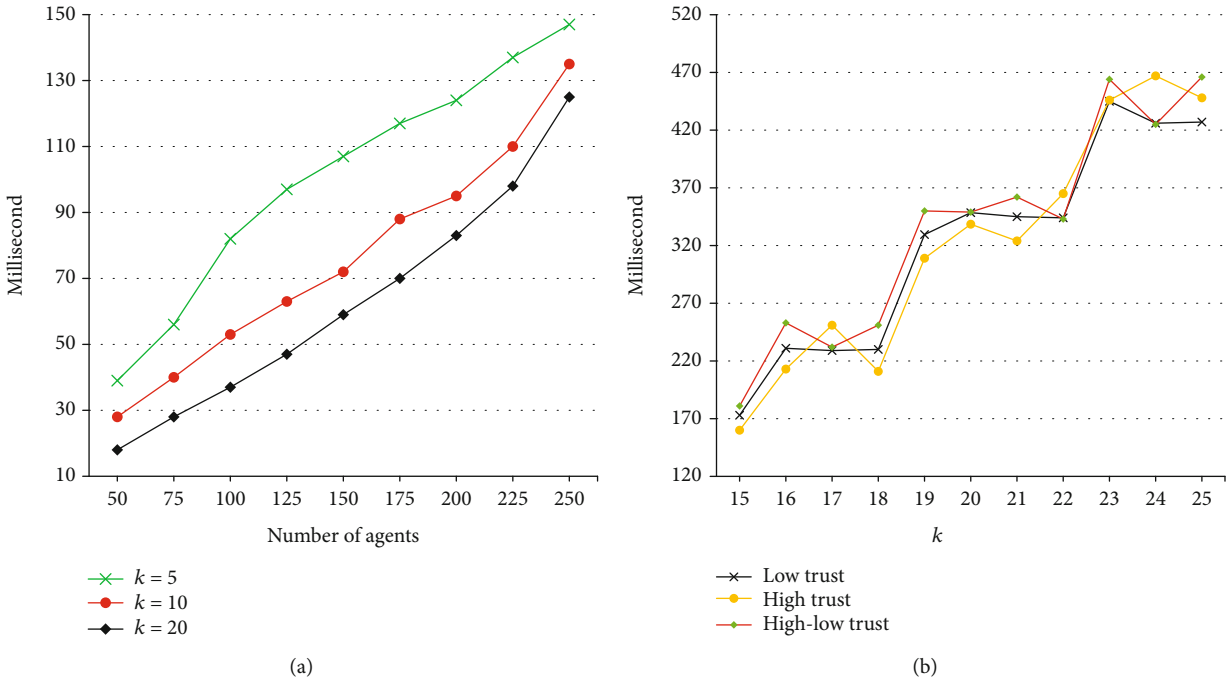


Figure 9: Relationship between the number of agents and runtime.

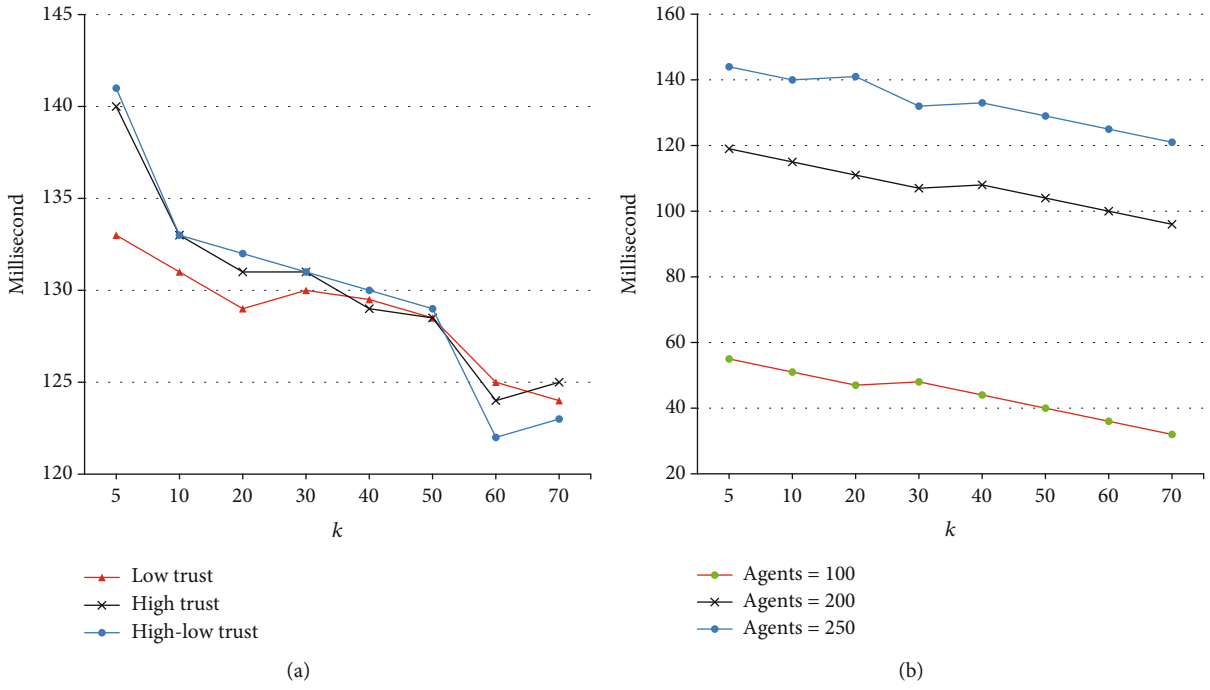


Figure 10: Relationship between the number of coalitions and runtime.

8GB memory, 3.2GHz main frequency, and i5-6500 processor. The programming language is java, which runs on eclipse.

We chose the classic CSG algorithm as the comparison algorithm, with the DP algorithm and ODP-IP algorithm. In addition, we also implement the efficient algorithm of Saran and Vazirani [34] which is called SV (Saran and Vazirani) algorithm. The SV algorithm is an approximate algorithm for solving min-k-cut problem. The algorithm recursively selects the edge with the smallest weight to cut until the graph is divided into k parts.

We use utility and runtime as indexes to evaluate the coalition structure. On the one hand, we study the influence of trust, the number of agents is n, and the number of coalitions is k on the utility of CS. On another hand, we study the influence of the number of agents n and the number of coalitions k on runtime.

5.2. Experimental Results and Performance Analysis

5.2.1. Effect of ρ, n, k on Utility. This section mainly discusses the influence of trust, the number of agents n, and the number of coalitions k for the ECCF algorithm on the utility of the coalition structure. Among them, $0 < \rho < 0.5$ is a low trust relationship, $0.5 < \rho < 1$ is a high trust relationship, and $0 < \rho < 1$ is a high-low trust relationship.

Figure 6 shows the relationship between trust and utility among agents in the trust-utility graph of the ECCF algorithm, where the x-coordinate represents the number of agents and the y-coordinate represents the utility of the ECCF algorithm. It is easy to know that trust and utility between agents show a linear relationship. The utility generated by the agent increases with the increase of the trust relationship. The greater the trust value between agents, the greater the utility of generating the coalition structure. In addition, the utility of the coalition structure is related to the number of agents. Figure 7 shows the relationship between agents and the utility of the coalition structure in ECCF algorithm. In general, the utility of the coalition structure increases with the number of agents. Through the analysis of Figure 7, it is easy to known that the utilities of the ECCF algorithm and the DP algorithm increase with the number of agents; the increase range is basically the same. Figure 8 shows the relationship among k coalitions in CS and the utility of coalition structure. The x-coordinate shows the number of coalitions in CS, and the y-coordinate shows the utility of CS. With the increase of k, the utility of ECCF algorithm decreases correspondingly, because the number of edges that need to be contracted decreases correspondingly and the number of edges that need to be deleted finally increases, so the final remaining utility decreases correspondingly.

5.2.2. Effect of n and k on Runtime. This section mainly discusses the effect of the number of agents n and the number of tasks k on runtime of the generated coalition structure. Through the trust-utility relationship graph, the coalition structure of different agent sets is solved, and the runtime required to generate the coalition structure is recorded. The results are shown in Figure 9. The x-coordinate is the number of agents, and the y-coordinate is the runtime of the Coalition Structure Generation; the unit is millisecond.

Through the analysis of Figure 9, it is easy to know that with the increase of the number of agents, the runtime of ECCF algorithm increases correspondingly and the runtime is not affected by the trust relationship. According to Figure 10, it is easy to know that with the increase of the number of k, the runtime of the ECCF algorithm decreases correspondingly, because the number of edges that need to be contracted decreases correspondingly, so the calculation amount decreases correspondingly. Similarly, it is not affected by the trust relationship; that is, the trust between agents only affects the utility of the formed coalition and does not affect the runtime of the coalition.

5.2.3. Algorithm Runtime Comparison. This section mainly discusses the changes on the runtime of the ECCF algorithm, DP algorithm, and ODP-IP algorithm with the increase of the number of agents. The comparison results are shown in Figure 11. The x-coordinate represents the number of agents, and the y-coordinate represents the runtime.

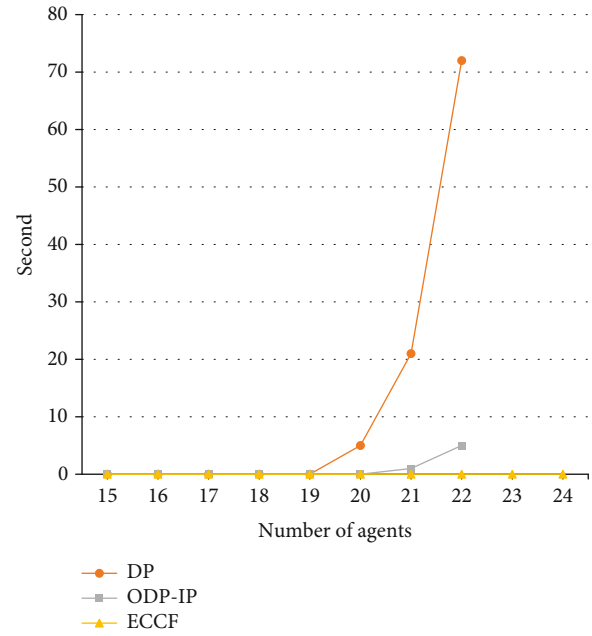


Figure 11: Runtime comparison of three algorithms.

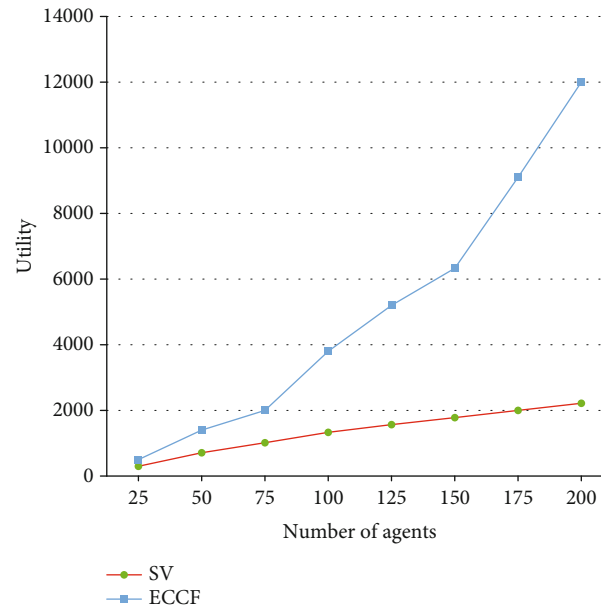


Figure 12: Utility comparison of two algorithms.

In Figure 11, it can be see that the runtime of the ECCF algorithm is far less than those of the DP algorithm and ODP-IP algorithm. Since the DP algorithm gradually decomposes the current solution into an optimal partition and records the intermediate solution to find the optimal coalition structure, the running time is relatively high. In addition, with the increase of the number of agents, the effect of time comparison is more obvious.

5.2.4. Algorithm Utility Comparison. From Figure 12, we can see that the utility of the coalition structure obtained by the ECCF algorithm is greater than that by the SV algorithm. In addition, with the increase of the number of agents, the effect of utility is more obvious. Because the SV algorithm recursively deletes the edge of the minimum trust-utility relationship until k coalitions are obtained. Then, the minimum trust-utility relationship among agents in the generated k coalitions will also be deleted, which will cause the waste of resources. The utility generated by the SV algorithm is less than that by the ECCF algorithm in most cases.

6. Conclusions

In this paper, we use a given number of coalitions to analyze the trust and utility of CSG. This paper proposed an ECCF algorithm based on edge contraction, which can solve the GCSG problem well. Firstly, trust is transitive, and the trust and utility are transferred according to the trust transfer. Then, the image of the transferred trust network is segmented to obtain a feasible coalition structure. Finally, the ECCF algorithm can be applied to large-scale systems, and the feasible solution can be obtained quickly.

In future works, we will analyze the search problem of the coalition core in the coalition structure. Another potential research direction is the study of the number of agents in coalition. In order to remove the coalition with a small number of agents in the solution, the size of the coalition can be limited.

Data Availability

Data are available on request through contacting long984384142@163.com.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article. This research was financially supported by the National Natural Science Foundation of China under Grant No. 62072392 and the China Postdoctoral Science Foundation under Grant No. 2019T120732 and Grant No. 2017M622691.

References

[1] M. Rogna, "Coalition formation and bargaining protocols: a review of the literature," *Journal of Economic Surveys*, vol. 33, no. 1, pp. 26–251, 2019.

[2] E. Elkind, T. Rahwan, and N. R. Jennings, "Computational coalition formation," *Annals of Nuclear Energy*, vol. 6, no. 3, pp. 329–380, 2013.

[3] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, no. 1-2, pp. 209–238, 1999.

[4] T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci, "An anytime algorithm for optimal coalition structure generation," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, pp. 521–567, 2009.

[5] R. B. Myerson, "Graphs and cooperation in games," *Mathematics of Operations Research*, vol. 2, no. 3, pp. 225–229, 1977.

[6] Y. Huang, M. Chen, Z. Cai, X. Guan, T. Ohtsuki, and Y. Zhang, "Graph theory based capacity analysis for vehicular ad hoc networks," in *IEEE Global Telecommunications Conference (GLOBECOM 2015)*, pp. 1–5, San Diego, CA, USA, December 2015.

[7] D. Miao, Z. Cai, J. Yu, and Y. Li, "Triangle edge deletion on planar glasses-free RGB-digraphs," *Theoretical Computer Science*, vol. 788, pp. 2–11, 2019.

[8] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[9] K. Bakht, F. Jameel, Z. Ali et al., "Power allocation and user assignment scheme for beyond 5g heterogeneous networks," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 2472783, 11 pages, 2019.

[10] T. Liu, Y. Wang, Y. Li, X. Tong, L. Qi, and N. Jiang, "Privacy protection based on stream cipher for spatiotemporal data in IoT," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7928–7940, 2020.

[11] Z. Sun, Y. Wang, Z. Cai, T. Liu, X. Tong, and N. Jiang, "A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing," *International Journal of Intelligent Systems*, 2021.

[12] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[13] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, July 2019.

[14] Z. Cai, X. Zheng, and J. Yu, "A differential-private framework for urban traffic flows estimation via taxi companies," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492–6499, 2019.

[15] B. Huang, Z. Li, Y. Xu et al., "Deep reinforcement learning for performance-aware adaptive resource allocation in mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 2765491, 17 pages, 2020.

[16] Y. Wang, Z. Cai, Z.-H. Zhan, B. Zhao, X. Tong, and L. Qi, "Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1033–1046, 2020.

[17] D. Y. Yeh, "A dynamic programming approach to the complete set partitioning problem," *BIT Numerical Mathematics*, vol. 26, no. 4, pp. 467–474, 1986.

[18] T. P. Michalak, T. Rahwan, E. Elkind, M. Wooldridge, and N. R. Jennings, "A hybrid exact algorithm for complete set partitioning," *Artificial Intelligence*, vol. 230, pp. 14–50, 2016.

[19] S. Ueda, A. Iwasaki, V. Conitzer, N. Ohta, Y. Sakurai, and M. Yokoo, "Coalition structure generation in cooperative games with compact representations," *Autonomous Agents and Multi-Agent Systems*, vol. 32, no. 4, pp. 503–533, 2018.

[20] F. Wu and S. D. Ramchurn, "Monte-Carlo tree search for scalable coalition formation," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 407–413, Yokohama, Japan, July 2020.

[21] S. Arib, S. Aknine, and T. Cazenave, "Nested Monte-Carlo search for multi-agent coalitions mechanism with constraints," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, pp. 80–88, Kuala Lumpur, Malaysia, November 2015.

[22] Z. Cai, G. Lin, and G. Xue, "Improved approximation algorithms for the capacitated multicast routing problem," in *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON 2005)*, pp. 136–145, Kunming, China, August 2005.

[23] Z. Cai, Z.-Z. Chen, and G. Lin, "A 3.4713-approximation algorithm for the capacitated multicast tree routing problem," *Theoretical Computer Science*, vol. 410, no. 52, pp. 5415–5424, 2009.

[24] Z. Cai, R. Goebel, and G. Lin, "Size-constrained tree partitioning: a story on approximation algorithm design for the multicast k-tree routing problem," in *Combinatorial Optimization and Applications. COCOA 2009. Lecture Notes in Computer Science, vol 5573*pp. 363–374, Springer, Berlin, Heidelberg.

[25] Z. Cai, R. Goebel, G. Lin et al., "Size-constrained tree partitioning: approximating the multicast k-tree routing problem," *Theoretical Computer Science*, vol. 412, no. 3, pp. 240–245, 2011.

[26] T. D. Voice, S. D. Ramchurn, and N. R. Jennings, "On coalition formation with sparse synergies," in *Proceedings of the 11th International Con-ference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pp. 223–230, Valencia, Spain, June 2012.

[27] B. Filippo, F. Alessandro, C. Jesus, J. Rodríguez-Aguilar, and S. D. Ramchurn, "Algorithms for graph-constrained coalition formation in the real world," *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 4, pp. 1–24, 2017.

[28] M. Ghaffari, K. Nowicki, and M. Thorup, "Faster algorithms for edge connectivity via random 2-out contractions," in *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, pp. 1260–1279, Salt Lake City, UT, USA, January 2020.

[29] L. Sless, N. Hazon, S. Kraus, and M. Wooldridge, "Forming k coalitions and facilitating relationships in social networks," *Artificial Intelligence*, vol. 259, pp. 217–245, 2018.

[30] G. Han, J. Du, C. Lin, H. Wu, and M. Guizani, "An energy-balanced trust cloud migration scheme for underwater acoustic sensor networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1636–1649, 2020.

[31] C. Kolias, W. Meng, G. Kambourakis, and J. Chen, "Security, privacy, and trust on internet of things," *Wireless Communications and Mobile Computing*, vol. 2019, 3 pages, 2019.

[32] Y. Wang, G. Yin, Z. Cai, Y. Dong, and H. Dong, "A trust-based probabilistic recommendation model for social networks," *Journal Network Computer Applications*, vol. 55, pp. 59–67, 2015.

[33] S. Brânzei and K. Larson, "Coalitional affinity games," in *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2009)*, pp. 1319-1320, Budapest, Hungary, May 2009.

[34] H. Saran and V. V. Vijay Vazirani, "Finding k cuts within twice the optimal," *SIAM Journal Computing*, vol. 24, no. 1, pp. 101–108, 1995.