IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2021, accepted July 19, 2021, date of publication July 26, 2021, date of current version August 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3099771

Task Allocation and Utility Distribution Algorithms Based on Nash Bargaining Solution

MING L. FU, LV Q. CHEN, YU J. WAN, AND DE B. CHEN
School of Computer Science and Technology, Huaibei Normal University, Huaibei 235000, China

Corresponding author: Lv Q. Chen (l_q_chen@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61976101 and Grant 62006091, and in part by the University Natural Science Research Project of Anhui Province under Grant KJ2020A0033.

ABSTRACT In practical applications, there are many task allocation problems involving the participation of self-interested agents, including Witkey, crowdsourcing and electronic markets. In these cases, to improve the efficiency of task allocation, a reasonable distribution of utilities is critical. To the best of our knowledge, few studies have examined the complex task allocation and utility distribution of self-interested agents, and good solutions are lacking. To address this issue, the following works are done in this paper: first, based on a task allocation model for self-interested agents and by studying the Nash bargaining solution and the bargaining characteristics of the agents, an efficiency utility distribution algorithm satisfying individual rationality and budget effectiveness is proposed. Second, based on the best response strategy of the self-interested agent, a complex task allocation algorithm for multiple self-interested agents is proposed. Finally, the effectiveness of the proposed algorithm is verified by comparing the system revenues with other utility distribution and task allocation algorithms.

INDEX TERMS Nash bargaining solution, self-interested agent, task allocation, utility distribution.

I. INTRODUCTION

In recent years, the task allocation problem of multiagent systems has been extensively studied by scholars in related fields. Agents can be categorized as cooperative or self-interested. Cooperative agents perform tasks that maximize system revenue, even if their decisions adversely affect their own individual revenue. However, self-interested agents select tasks solely in terms of their own incomes, regardless of the impact of their decisions on other agents' benefits or system revenue. The coalition skill game studied in this paper is a simple and typical self-interested agent task allocation model. In real economic life, there are many similar task allocation situations, such as the rise of some large online recruitment markets and task management crowdsourcing platforms [1]–[3], in which task requesters are responsible for releasing tasks and providing compensation to the workers who complete the tasks. Workers with different skills can choose tasks that satisfy their specific needs and goals. They have the right to reject unsatisfying tasks assigned by the system. The workers and the task requesters are all self-interested. For a task allocation problem involving self-interested agents, the reasonable distribution of income is critical to obtaining larger system and individual revenues.

In the research field of multiagent task allocation in artificial intelligence, few studies have focused on self-interested agents. Relative to the task allocation of cooperative agents in a general sense, self-interested agent task allocation has unique properties (more than are listed here): (1) At any moment, self-interested agents always seek to maximize their own individual revenues when making decisions. Even if they know that other decisions will bring greater system revenues, they will ignore them. That is, the task allocation of self-interested agents must satisfy individual rationality [4], [5]. However, cooperative agents always prioritize system revenue within their capabilities (communication ability, computing power, etc.) when making decisions. (2) Self-interested agents, to maximize their own individual revenues, may intentionally make decisions that damage others' individual revenues or system revenue. (3) For a self-interested agent, after a task is completed, whether the utility distribution scheme is reasonable directly affects whether cooperation can continue. An unreasonable distribution scheme may affect the enthusiasm of self-interested

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

agents in participating in future task completion or even cause them to terminate task execution midway. For cooperative agents, utility distribution is not a problem that must be considered. At the same time, for a self-interested agent, utility distribution must also meet budget validity [6], [7]. (4) When communicating, to obtain better individual revenue, self-interested agents may withhold personal information, such as concealing their geographical location, exaggerating their cost, and even hiding the contents of the task. Little or no real information will directly affect the overall performance of the system. For cooperative agents, there is usually an implicit assumption that as long as the communication conditions permit, they are willing to transmit all their information to other agents or centralized managers without reservations. (5) The system revenue of self-interested agent task allocation cannot exceed the system revenue of cooperative agent task allocation under the same conditions.

Because of the above differences, some existing task allocation algorithms for non-self-interested agents cannot be directly used to solve task allocation problems for self-interested agents. In this paper, using the task allocation model of a coalition skill game as the research object, a utility distribution scheme based on the Nash bargaining solution is proposed. On this basis, this paper proposes a task allocation algorithm for self-interested agents based on the best response strategy that satisfies individual rationality. The algorithm proposed in this paper is named task allocation based on Nash bargaining solution (TANBS). The utility distribution algorithm in TANBS improves the fairness of the existing algorithms. Compared with the existing algorithms, the task allocation algorithm proposed in this paper considers individual rationality and ensures the stability of the task allocation results; at the same time, it can ensure a higher total revenue. It solves the problem where the existing algorithms cannot consider these three aspects simultaneously.

The remainder of this paper is structured as follows. Related works are discussed in section II. Section III presents the formal problem definitions considered in this paper. Section IV presents the basic concepts of TANBS and analyzes its convergence. The simulation results in section V show that our proposed algorithm can allocate tasks to self-interested agents effectively. Finally, the conclusion of our work is presented in section VI.

II. RELATED WORKS

The classic coalition skill game was proposed by Bachrach [8], [9]. It contains three sets: a service agent set, a skill set and a task set. Among them, the service agents are completely self-interested. Each service agent has one or more skills. Each task agent also corresponds to a set of skills, indicating which skills are needed. The task can only be completed if all the required skills are met. At the same time, each task also corresponds to a utility value, and the system revenue is defined as the sum of the utilities of the tasks that can be completed. The task allocation problem in the coalition skill game is as follows: under the premise of ensuring the individual rationality of the service agent, allocate tasks to service agents to obtain the maximum system revenue.

The following text analyzes the research status from two aspects: utility distribution algorithms and complex task allocation algorithms for self-interested agents.

A. UTILITY DISTRIBUTION ALGORITHMS

When the set of tasks that can be completed by all possible service agent coalitions is known and these task sets are independent of each other, then the solution concept of a cooperative game can be used to distribute utilities to service agents [10], [11]. However, in practical applications, the task sets that can be completed by service agent coalitions are usually unknown, and for n service agents, there are 2^n coalitions, which require considerable computation. At the same time, when a task is assigned to a service agent coalition, the task cannot be allocated to another coalition, so the task sets that different coalitions can complete are usually not independent of each other.

In CRA [12], the rate of return (RoR) concept was introduced to distribute the utilities. The RoR is defined as the ratio of the received reward and incurred cost for each service agent. To obtain the property of fairness, CRA tries its best to make all the service agents' RoRs the same. The fairness of the RoR lies in the setting that the service agent with more cost can obtain more reward. However, this is not conducive to the improvement of total system revenue. In contrast, to improve the total system revenue, tasks should be allocated to service agents at lower costs. The IRA algorithm can solve the problem of utility distribution and task allocation in coalition skill games, but when the number of skills required by any task agent is large, the time complexity is high [13]. With the development of the economy and internet technology, the number of service agents and tasks in crowdsourcing and other practical applications is growing rapidly, and the tasks to be completed also show a trend of increasing complexity. An increasing number of tasks require the cooperation of multiple service agents, so it is necessary to study the utility distribution and task allocation algorithms of complex tasks with low time complexity.

B. COMPLEX TASK ALLOCATION ALGORITHMS FOR SELF-INTERESTED AGENTS

Game theory is an effective method for studying the decision making of self-interested agents. Learning algorithms based on game theory mainly include best response (BR) [14], [15], fictitious play (FP) [16] and computationally efficient sampled FP (CESFP) [17]. Among them, the premise of the best response strategy to obtain better task allocation results is to distribute the utility reasonably, but the best response strategy does not consider the problem of utility distribution. In the fictitious play algorithm, the decisions of self-interested service agents are based on their own historical information, which will limit the improvement of task allocation results.

If the task sets completed by each service agent set are independent of each other, obtaining the task allocation

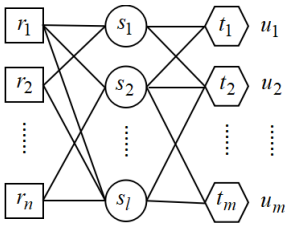


FIGURE 1. Graphic representation of the task allocation model.

scheme with the maximum system revenue is equivalent to solving the optimal coalition structure generation problem [18], [19]. However, it is very difficult to calculate the optimal coalition structure, regardless of whether it is a general coalition skill game model or a strictly restricted one [20]. Two kinds of computational intelligence algorithms—binary particle swarm optimization algorithms and binary differential evolution algorithms—have been proposed to solve the problem of coalition structure generation in coalition skill games. However, the research model is a characteristic function game, in which the coalition value of the service agent set is only determined by the task selected by its members. However, in the task allocation model of the coalition skill game studied in this paper, the coalition value of each service agent set not only depends on its member agents but also depends on nonmember agents [21].

Other similar studies on multiagent systems have focused on resource allocation [22]–[26] and single agent task allocation [27]–[32]. The difference between resource allocation and task allocation in the coalition skill game is that the resources owned by the service agent can be transferred, while the skills owned by the service agent in the coalition skill game cannot be transferred between the service agents [33]. Single agent tasks refer to tasks that can be completed without the cooperation of multiple service agents. There are single agent tasks in the coalition skill game model, but in this paper, more attention is given to the complex tasks that need the cooperation of multiple service agents [34].

III. TASK ALLOCATION MODEL AND ITS PROPERTIES

The coalition skill game model for self-interested agent task allocation includes three sets: service agent set $R := \{r_1, r_2, \cdots, r_n\}$, skill set $S := \{s_1, s_2, \cdots, s_l\}$ and task set $T := \{t_1, t_2, \cdots, t_m\}$, where $n = |R|$, $l = |S|$, $m = |T|$, and $|*|$ denotes the cardinality of set $*$. For $i \in \{1, 2, \cdots, n\}$ and $j \in \{1, 2, \cdots, l\}$, $RS_{i,j} = 1$ ($RS_{i,j} = 0$) indicates service agent r_i possesses (does not possess) skill s_j, $cost_{i,j}$ indicates the cost when service agent r_i completes the task with skill s_j, and if $RS_{i,j} = 0$, $cost_{i,j} = 0$ For $j \in \{1, 2, \cdots, l\}$ and $k \in \{1, 2, \cdots, m\}$, $ST_{j,k} = 1$ ($ST_{j,k} = 0$) indicates task t_k needs (does not need) skill s_j. The utilities of all tasks are denoted by a vector of $U := \{u_1, u_2, \cdots, u_m\}$. An intuitive graphic representation of the task allocation model is shown in Figure 1. The line between $r_i \in R$ and $s_j \in S$ indicates that user r_i possesses skill s_j. The line between $s_j \in S$ and $t_k \in T$ indicates that task t_k needs skill s_j. The numbers next to the tasks indicate their utilities.

The utility distribution schemes of all the task agents are denoted with TS. For $k \in \{1, 2, \cdots, m\}$ and $j \in \{1, 2, \cdots, l\}$, $TS_{k,j}(\tau)$ denotes the share distributed to s_j by t_k at time τ. If task t_k can be completed, r_i chooses task t_k and provides skill s_j, which will obtain the corresponding utility share $TS_{k,j}(\tau)$. $RTS(\tau)$ represents the tasks selected and skills provided by the service agent at time τ. $RTS_{i,0}(\tau)$ is the index of the task selected by $r_i \in R$ at time τ, $RTS_{i,0}(\tau) \in \{1, 2, \cdots, m\}$. For $i \in \{1, 2, \cdots, n\}$ and $j \in \{1, 2, \cdots, l\}$, $RTS_{i,j}(\tau) = 1$ ($RTS_{i,j}(\tau) = 0$) denotes r_i provides (does not provide) skill s_j for task $t_{RTS_{i,0}(\tau)}$ at time τ. If $RTS_{i,0}(\tau) = 0$, for $j \in \{1, 2, \cdots, l\}$, $RTS_{i,j}(\tau) = 0$.

Task agent t_k can be completed if all needed skills are provided. Given $RTS(\tau)$, the system revenue is defined as the sum of the utilities of the tasks that can be completed. The optimal task allocation to a multiple self-interested agent coalition skill game is an allocation of tasks to service agents that maximizes the system revenue.

In this paper, for the simplicity of analysis, the following assumptions are made: (1) each service agent has only one skill and is allowed to select at most one task at any time. (2) It is assumed that the unit of skill required by any task agent is 1. (3) At any time, the self-interested service agent will only choose the task that can bring it the maximum individual revenue. (4) For the tasks assigned by the system, the service agent has the right to reject and select a more satisfactory task.

The states of service agents and tasks at time τ are denoted as follows:

State of Service Agent: The state of service agent $r_i \in R$ at time τ is denoted by $r_i(\tau) :=< RS_{i,\cdot}, RTS_{i,\cdot}(\tau), cost_{i,\cdot} >$, where $RS_{i,\cdot}$ is the i^{th} row of RS and $RTS_{i,\cdot}(\tau) := \{RTS_{i,0}(\tau), RTS_{i,1}(\tau), \cdots, RTS_{i,l}(\tau)\}$, $cost_{i,\cdot}$ is the i^{th} row of $cost$.

State of Task Agent: The state of $t_k \in T$ at time τ is denoted by $t_k(\tau) :=< ST_{\cdot,k}, TSN_k, u_k, TS_{k,\cdot}(\tau) >$, where $ST_{\cdot,k}$ is the k^{th} column of ST. $TSN_k \in Z^+$ denotes the number of skills needed by t_k. u_k corresponds to the utility of t_k representing the value that completing the task is worth. $TS_{k,\cdot}(\tau)$ is the k^{th} row of $TS(\tau)$.

At time τ, for a service agent with skill s_j and a task agent that needs skill s_j, perform the following operations: first, order the cost of service agent using skill s_j in the order of small to large, and order the utility share of task agent assigned to skill s_j in the order of large to small. Then, start pairing from the first pair, if the cost is less than or equal to the utility share provided by the task, the match is successful. For skill s_j, the last pair of successfully paired "service agent-task" pairs is denoted as "r_s - t_s", and the corresponding cost and utility share are denoted as a_s and b_s, respectively, where s represents the number of successful pairings. The first pair of "service agent-task" unsuccessful pairs is denoted as "r_{s+1} - t_{s+1}", and the corresponding cost and utility share are recorded as a_{s+1} and b_{s+1}, respectively. If $s = 0$, then there is no successfully paired "service agent-task" pair; then, the

task should increase the share value assigned to skill s_j. If r_{s+1} does not exist, let $a_{s+1} = Inf$ (where Inf represents an infinite number), and if t_{s+1} does not exist, let $b_{s+1} = 0$. Then, the following attributes 1 and 2 hold.

Attribute 1: The external option of r_s is shown in formula (1):

$$gain_{as} = \begin{cases} 0, & a_s > b_{s+1} \\ b_{s+1} - a_s, & a_s \le b_{s+1} \end{cases} \tag{1}$$

The external option of t_s is shown in formula (2):

$$gain_{bs} = \begin{cases} 0, & a_{s+1} > b_s \\ b_s - a_{s+1}, & a_{s+1} \le b_s \end{cases} \tag{2}$$

The Nash bargaining solution of r_s and t_s is shown in formula (3):

$$\begin{aligned} nash_{js} &= a_s + \frac{gain_s + gain_{as} - gain_{bs}}{2} \\ &= b_s - \frac{gain_s + gain_{bs} - gain_{as}}{2} \end{aligned} \tag{3}$$

where $gain_s = b_s - a_s$.

Proof: The service agents that can provide skill s_j at time τ are sorted from smallest to largest according to their costs, the service agent ranked in position p is denoted r_p, and the cost is denoted a_p. The task agents are sorted from the largest to the smallest according to the utility shares they distribute to skill s_j, the task ranked in position p is denoted as t_p, and the corresponding utility share is b_p.

If r_s is unwilling to pair with t_s, r_s can be paired with t_{s+1} because t_{s+1} has no service agent that can successfully pair with it. When $a_s \le b_{s+1}$, the pairing succeeds; otherwise, the pairing fails. For r_s and t_p ($1 \le p \le s$ or $s+1 < p \le m$), t_p either is unwilling to pair with r_s, or even if it is willing to pair, the cost difference between r_s and t_p is not larger than that between r_s and t_{s+1}. This is because if $1 \le p < s$, then $a_p \le a_s$, $b_p - a_p \ge b_p - a_s$. In this case, t_p is more willing to pair with r_p than r_s. Similarly, when $s+1 < p \le m$, $b_p \le b_s$, then $b_p - a_s \le b_s - a_s$.

Thus, r_s has an external option:

$$gain_{as} = \begin{cases} 0, & a_s > b_{s+1} \\ b_{s+1} - a_s, & a_s \le b_{s+1}; \end{cases}$$

in the same way,

t_s has external option: $gain_{bs} = \begin{cases} 0, & a_{s+1} > b_s \\ b_s - a_{s+1}, & a_{s+1} \le b_s. \end{cases}$

In the negotiation of how the benefit $gain_s = b_s - a_s$ will be distributed, r_s will request at least $gain_{as}$, and t_s will request at least $gain_{bs}$. The residual income $gain_s - gain_{as} - gain_{bs}$ is averagely distributed between them. Therefore, the individual revenue obtained by r_s is

$$gain_{as} + \frac{gain_s - gain_{as} - gain_{bs}}{2} = \frac{gain_s + gain_{as} - gain_{bs}}{2}.$$

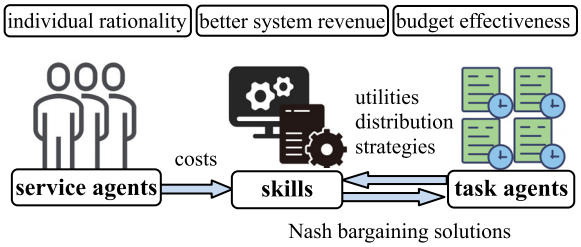


FIGURE 2. Utility distribution and task allocation process.

Similarly, the revenue obtained by t_s is:

$$gain_{bs} + \frac{gain_s - gain_{as} - gain_{bs}}{2} = \frac{gain_s + gain_{bs} - gain_{as}}{2}.$$

The Nash bargaining solution of r_s and t_s is:

$$\begin{aligned} &a_s + \frac{gain_s + gain_{as} - gain_{bs}}{2} \\ &= b_s - \frac{gain_s + gain_{bs} - gain_{as}}{2} \\ &= nash_{j,s}. \end{aligned}$$

Thus, Attribute 1 holds.

Attribute 2: $\forall p(1 \le p < s)$, $nash_{jp} = nash_{js}$.

Proof: The definitions of r_p, t_p, a_p and b_p are the same as the definitions in the proof of Attribute 1. When $p = s-1$, the external option of r_{p-1} is $gain_{a,s-1} = nash_{js} - a_{s-1}$. This is because if r_{s-1} is not paired with t_{s-1}, pairing with t_s can guarantee the maximum individual revenue. The Nash bargaining solution of t_s and r_s is $nash_{js}$, so if r_{s-1} is paired with t_s, it must be ensured that the revenue obtained by t_s is greater than or equal to $b_s - nash_{js}$. Then, the maximum individual revenue r_{s-1} can obtain is $nash_{js} - a_{s-1}$. Similarly, the external option of t_{s-1} is $gain_{b,s-1} = b_{s-1} - nash_{js}$.

From Attribute 1, it can be concluded that the following formula is true:

$$\begin{aligned} &nash_{j(s-1)} \\ &= a_{s-1} + \frac{gain_{s-1} + gain_{a,s-1} - gain_{b,s-1}}{2} \\ &= a_{s-1} + \frac{1}{2}(b_{s-1} - a_{s-1} + nash_{js} - a_{s-1} - b_{s-1} + nash_{js}) \\ &= a_{s-1} + \frac{1}{2}(2nash_{js} - 2a_{s-1}) \\ &= nash_{js}. \end{aligned}$$

Similarly, the following equation holds:

$$nash_{j(s-2)} = nash_{j(s-1)} \cdots .$$

Thus, $nash_{j1} = nash_2 = \cdots = nash_{j(s-1)} = nash_{js}$, so Attribute 2 holds.

IV. TASK ALLOCATION AND UTILITY DISTRIBUTION ALGORITHM DESCRIPTION

The process of utility distribution and task allocation is shown in Figure 2. The service agent adopts the best response

strategy to select task, which is described in section A. The centralized manager, such as crowdsourcing platform, calculates the Nash bargaining solution for each skill according to the costs and utilities obtained. The tasks distribute utilities according to the Nash bargaining solutions. Utility distribution strategies are described in section B. The whole frame of TANBS is given in section C, and its convergence is also demonstrated.

A. TASK SELECTION STRATEGY OF SERVICE AGENT

Let $t(i, \tau+1)$ denote the task selected by $r_i \in R$ at time $\tau+1$.

$$t(i, \tau+1) \leftarrow \underset{t_k \in T}{argmax} \sum_{s_j \in S \wedge need_{k,j}(\tau)} TS_{k,j}(\tau) \cdot RS_{i,j} \quad (4)$$

where $need_{k,j}(\tau)$ represents whether the skill s_j required by task t_k is provided by another service agent under the task selection state $RTS(\tau)$. This task selection method guarantees the individual rationality of the service agents.

B. UTILITY DISTRIBUTION STRATEGY OF TASK AGENT

According to Attributes 1 and 2, under the task selection state $RTS(\tau)$, the Nash bargaining solution of $s_j \in S$ at time τ can be computed with formula (5), where the definitions of s, a_s, b_s, a_{s+1} and b_{s+1} are the same as those in section 3.

$$Nash_j(\tau) = \begin{cases} \frac{a_s + b_s}{2}, & s \geq 1 \wedge a_{s+1} > b_s \wedge a_s > b_{s+1}; \\ \frac{b_s + b_{s+1}}{2}, & s \geq 1 \wedge a_{s+1} > b_s \wedge a_s \leq b_{s+1}; \\ \frac{a_s + a_{s+1}}{2}, & s \geq 1 \wedge a_{s+1} \leq b_s \wedge a_s > b_{s+1}; \\ \frac{a_{s+1} + b_{s+1}}{2}, & s \geq 1 \wedge a_{s+1} \leq b_s \wedge a_s \leq b_{s+1}; \\ -1, & s = 0 \wedge (a_1 = Inf \vee b_1 = -1); \\ a_1, & s = 0 \wedge a_1 \neq Inf \wedge b1 > 0. \end{cases} \quad (5)$$

With the Nash bargaining solution of each needed skill, tasks distribute their utilities according to formula (6).

$$TS_{k,j}(\tau) = \frac{Nash_j(\tau) \cdot u_k}{\sum_{s_j \in S \wedge ST_{j,k}=1} Nash_j(\tau)} \quad (6)$$

For the utility distribution scheme $TS(\tau)$ of task $t_k \in T$, $u_k = \sum_{s_j \in S \wedge ST_{j,k}=1} TS_{k,j}(\tau)$ holds, and if $ST_{j,k} = 1$, $TS_{k,j}(\tau) \geq 0$, which guarantees the budget validity of task allocation.

C. THE WHOLE FRAME OF TANBS

The whole frame of TANBS is shown in Algorithm 1, where $H_i^R(\tau) = 0$ denotes service agent r_i is performing a task and is not in the system at time τ. In contrast, $H_i^R(\tau) = 1$ denotes service agent r_i is still in the system at time τ. Similarly, $H_k^T(\tau) = 0$ denotes task agent t_k has acquired all the needed skills and is not in the system at time τ. $H_k^T(\tau) = 1$ denotes task agent t_k is still in the system at time τ. MSR denotes the maximum system revenue.

Algorithm 1 TANBS

Inputs: RS, ST, U, $cost$.

Outputs: the maximal total system revenue MSR and the corresponding RTS.

1: **Initialize** $TS(0)$: the utilities of task agents are distributed averagely among the needed skills. Initialize $RTS(0)$: the service agents take turns to select tasks according to formula (4) until all the tasks selected by the service agents in two consecutive rounds are completely the same. For $r_i \in R$, $H_i^R(0) \leftarrow 1$, and for $t_k \in T$, $H_k^T(0) \leftarrow 1$. ($\tau = 0$).
2: **WHILE** $\exists t_k \in T$ s.t. $H_k^T(\tau) = 1$ ($\tau \geq 1$)
3: For $s_j \in S$, compute the value of $Nash_j(\tau)$ according to formula (5);
4: For $t_k \in T$ ($H_k^T(\tau) = 1$), $s_j \in S$, compute $TS_{k,j}(\tau)$ according to formula (6);
5: the service agents take turns to select tasks according to formula (4), until all the tasks selected by the service agents in two consecutive rounds are completely the same;
6: **FOR** $t_k \in T(H_k^T(\tau) = 1)$
7: **IF** t_k can be completed
8: $H_k^T(\tau) \leftarrow 0$;
9: Delete task t_k and all service agents who select task t_k at time τ;
10: **END IF**
11: **END FOR**
12: Delete the task agents that cannot be completed;
13: If there is not any task agent is deleted in line 9 and 12,
delete $t_{\max} = \underset{t_d \in T \wedge H_d^T(\tau)=1}{agrmax} (u_d / TSN_d)$. A set of service agents needed by t_{max} are chosen with a Greedy Strategy and deleted.
14: **END WHILE**
15: Record the values of MSR, $TS(\tau)$ and $RTS(\tau)$.

The following text explains why the TANBS algorithm will end in finite steps. Lines 9, 12 and 13 ensure that the maximum number of iterations of the while loop for lines 2-14 is m. For lines 1 and 5, when the utility distribution schemes of all the tasks are fixed, the service agents take turns to select tasks that can bring them the maximum individual revenues according to the best response strategy until the tasks selected by all the service agents for two consecutive rounds are exactly the same. It is assumed that the unit of skill required by the task agent is 1. For any two service agents, either they are willing to cooperate to complete a task or select another different task because the current task cannot be completed. Therefore, the game model in TANBS belongs to the Weak Acyclic Games, and each Weak Acyclic Game has limited improvement feature [35], so TANBS converges.

V. SIMULATION RESULTS

Service agents in the task allocation model of coalition skill game are rational. When making decisions, the service agents

first consider how to maximize their individual income. Therefore, the task allocation algorithm for this problem must satisfy individual rationality, otherwise the task allocation result will not be stable. However, the task allocation algorithm ensuring individual rationality will inevitably affect system revenue. That is to say, individual revenue and system revenue are not consistent. Reasonable utility distribution can solve the inconsistency problem. Based on the above thought, the TANBS algorithm is divided into two steps: firstly, the Nash bargaining solution is used to distribute the utilities, and secondly, combined with the optimal response strategy, the task allocation algorithm can obtain stable task allocation results. Therefore, this section verifies the effectiveness of TANBS from the following two aspects: in section A, under 4 groups of data sets, the average run times and average system revenues of TANBS are compared with those of the other 3 algorithms. This section is to show that the proposed algorithm can guarantee higher system revenue under the premise of individual rationality and task allocation stability. Section B verifies the effectiveness of the utility distribution strategy proposed in this paper. This section is to show that the utility distribution scheme based on Nash bargaining solution is more equitable. Simulation environment: memory, 8.0 GB; CPU, Intel(R) Core(TM) i3-8100; main frequency, 3.6 GHz; operating system, Win 10.

For the data set, as far as we know, there is currently no standard database for the coalition skill game system. For the resource-constrained project scheduling problem, the project scheduling problem library [36] is a benchmark dataset, but the resources do not possess multiple skills. Another similar dataset is iMOPSE [37], [38], which was created based on the real-life project instances received from international enterprises (Volvo IT). The resources in iMOPSE have multiple skills (where the resources can be seen as service agents in coalition skill games), but each task in iMOPSE needs only one resource. Other datasets are generated artificially [39], [40]. Analogously, datasets 1 and 2 in this work were generated artificially and randomly, while datasets 3 and 4 were obtained by modifying iMOPSE. The dataset generation methods are described as follows.

Dataset 1 contains 15 sets of data generated with the following stochastic method (denoted as 1.1-1.15): $n = 100$, $l = 15$, $m = 100$. Each service agent has one skill, which is random. If $RS_{i,j} = 1$, $cost_{i,j} \leftarrow random(1, 10)$, where $random(a, b)$ represents a random integer in set [a, b]. The number of skills required by a specific task agent is $random(1, 15)$. For $t_k \in T$, $u_k \leftarrow TSN_k \times ramdom(1, m/2)$.

Dataset 2 contains 15 sets of data (denoted as 2.1-2.15), which were obtained with the following stochastic method: $n = 1000$, $l = 20$, $m = 200$. Each service agent has one skill, which is random. If $RS_{i,j} = 1$, $cost_{i,j} \leftarrow random(1, 10)$. The number of skills required by a specific task agent is $random(1, 15)$. For $t_k \in T$, $u_k \leftarrow TSN_k \times ramdom(1, m/2)$.

Dataset 3 and dataset 4 both contain 15 groups of data (denoted as 3.1-3.15 and 4.1-4.15), which were generated based on iMOPSE. The generation steps of dataset 3 and dataset 4 are described as follows: (1) a total of 360 service agents were obtained from the following datasets: 100_20_22_15, 100_20_23_9_D1, 100_20_45_15, 100_20_47_9, 100_20_65_9, 100_20_65_15, 200_40_45_9, 200_40_45_15, 200_40_90_9, 200_40_91_15, 200_40_130_9_D4, 200_40_133_15. (2) A total of 400 task agents were obtained from 200_40_91_15 and 200_40_133_15. (3) In each group of dataset 3, 300 service agents and 150 task agents were selected randomly. (4) In each group of dataset 4, 200 service agents and 300 task agents were selected randomly. (5) Some modifications were made to the extracted datasets: the salaries of resources and the duration of the each task were ignored. Only one skill is possessed by each service agent. If $RS_{i,j} = 1$, $cost_{i,j} \leftarrow random(1, 10)$. A total of $random(1, 11)$ skills were added to each task because each task in iMOPSE needs only one skill, which is not inconsistent with the case of this paper. For $t_k \in T$, $u_k \leftarrow TSN_k \times random(1, m/2)$.

TABLE 1. The average system revenues of GGA with different values of CP and MP.

	0.01	0.05	0.1	0.15	0.2	0.3	0.4	0.6	0.8
0.1	837.4	1051.3	1121.5	1189.9	1064.2	649.9	509.1	449.6	425.4
0.3	1028.3	1223.8	1382.4	1133.5	822.7	586.5	507.1	453.4	433.3
0.5	1059.7	1357.9	1381.6	960.6	764.8	573.4	500.6	456.6	435.5
0.6	1127.7	1396.3	1232.0	897.6	720.2	584.8	509.7	446.6	441.9
0.7	1211.2	1420.3	1171.7	876.6	710.7	555.2	494.6	460.3	427.2
0.75	1147.1	1453.0	1154.0	844.5	702.3	559.5	506.7	463.0	436.5
0.8	1257.2	1363.7	1048.9	847.1	695.0	547.9	500.6	453.3	433.0
0.85	1254.5	1398.2	1082.0	856.0	660.0	552.1	488.1	452.3	435.1
0.9	1199.6	1357.5	1045.3	815.2	665.3	547.2	482.4	455.2	430.8

A. ALGORITHM PERFORMANCE COMPARISON

Even if the individual rationality, the stability and the fairness are not considered, to find an optimal solution of multiply skill task allocation problem is still NP hard. Its time complexity is $O(l^n m^n)$. Therefore, this problem is usually solved with approximate optimal algorithms. In this section, the system revenues of TANBS are compared with other 3 approximate optimal algorithms for dataset 1, 2, 3, and 4: the General Genetic Algorithm (GGA), the Service and Adams' Algorithm (SAA) [33], and the Combinatorial Bids-based Algorithm (CBA) [41]. Service agents in CBA are self-interested, and those in GGA and SAA are not. GGA and SAA consider neither the individual rationality nor how the utilities are distributed. The goal of all their decisions is to achieve a higher system revenue. Therefore, from the perspective of system revenue alone, GGA and SAA have the advantage. Although CBA assumes that the service agent is self-interested, it cannot guarantee the stability of task allocation results.

Dataset 1.1 was used to obtain the optimal parameters of GGA. With different values of crossover probability (CP) and mutation probability (MP), GGA ran 100 times, and the average system revenues are shown in Table 1 (rows represent the values of MP, and columns represent the values of CP). From the simulation results, it can be seen that the largest

TABLE 2. The average system revenues and average run times of TANBS, GGA, CBA, and SAA under dataset 1.

	TANBS	time (s)	GGA	time (s)	CBA	time (s)	SAA	time (s)
1.1	2800	0.016	1450.8	171.9	2619	0.011	2789	0.0005
1.2	2921	0.019	1319.6	145.0	2749	0.012	3092	0.0003
1.3	3163	0.019	1309.6	145.4	2908	0.008	3073	0.0005
1.4	2715	0.021	1197.1	148.2	2662	0.013	2680	0.0004
1.5	3070	0.019	1535.7	168.9	3003	0.018	2969	0.0005
1.6	2682	0.019	1245.2	176.2	2494	0.009	2786	0.0006
1.7	2874	0.018	1452.5	202.4	2788	0.013	2926	0.0005
1.8	3118	0.017	1207.5	146.0	3078	0.009	3080	0.0006
1.9	2984	0.016	1416.3	163.1	2722	0.014	2889	0.0005
1.10	2084	0.015	1419.6	169.8	2115	0.007	2015	0.0007
1.11	3155	0.017	1176.2	138.3	3024	0.011	3130	0.0006
1.12	2441	0.018	1135.3	130.8	2027	0.009	2404	0.0005
1.13	2611	0.022	864.9	103.0	2690	0.008	2625	0.0006
1.14	3084	0.018	1023.1	152.4	3074	0.010	2947	0.0005
1.15	2564	0.021	1293.3	197.0	2445	0.008	2553	0.0004

TABLE 3. The average system revenues and average run times of TANBS, GGA, CBA, and SAA under dataset 2.

	TANBS	time (s)	GGA	time (s)	CBA	time (s)	SAA	time (s)
2.1	61248	6.93	13488	8469	60766	0.86	58631	0.004
2.2	56985	7.25	13379	9342	54798	0.83	57514	0.005
2.3	60247	6.55	13041	9376	59647	0.81	59151	0.004
2.4	67673	6.50	11996	8053	67797	0.78	66947	0.004
2.5	65251	6.94	13043	7474	64918	0.90	63088	0.005
2.6	63240	6.63	13520	7334	62393	0.76	63414	0.004
2.7	59680	6.42	15211	8561	59058	0.84	61046	0.004
2.8	63547	6.73	11677	7447	62558	0.80	63960	0.006
2.9	60548	6.69	12598	6025	59918	0.85	59570	0.004
2.10	63969	7.09	13298	7565	62907	0.83	63551	0.005
2.11	60684	6.94	14222	6866	60167	0.81	61901	0.004
2.12	55984	7.64	11937	7867	55640	0.83	56390	0.006
2.13	62590	7.76	12781	8153	62309	0.67	63434	0.003
2.14	62225	7.09	13688	8027	61585	0.84	61526	0.004
2.15	65272	7.16	11608	8658	65279	0.81	63739	0.005

TABLE 4. The average system revenues and average run times of TANBS, GGA, CBA, and SAA under dataset 3.

	TANBS	time (s)	GGA	time (s)	CBA	time (s)	SAA	time (s)
3.1	11021	0.36	6387.8	1813.9	10352	0.045	10819	0.002
3.2	11223	0.34	7349.8	1727.8	10446	0.043	11652	0.003
3.3	10827	0.37	6134.4	1948.4	10119	0.055	10419	0.002
3.4	13013	0.41	6526.6	1692.4	12395	0.066	11607	0.003
3.5	13482	0.33	6798.6	1577.9	12678	0.076	11563	0.002
3.6	11682	0.34	6684.0	2069.9	10508	0.052	10993	0.002
3.7	13002	0.36	6870.4	1515.5	11896	0.049	12659	0.003
3.8	11439	0.33	6551.6	1528.9	11117	0.057	10772	0.002
3.9	12051	0.37	6102.8	1594.3	11013	0.060	12530	0.002
3.10	13516	0.36	6552.6	1984.5	11340	0.070	12295	0.003
3.11	12421	0.36	6473.4	1623.3	12005	0.066	11681	0.002
3.12	12617	0.43	6612.4	1616.9	10845	0.051	11546	0.003
3.13	10587	0.44	5597.0	1832.7	9657	0.062	10322	0.002
3.14	13766	0.36	7119.8	1509.9	12840	0.060	13042	0.002
3.15	12253	0.33	5778.0	1944.2	10527	0.049	11533	0.002

TABLE 5. The average system revenues and average run times of TANBS, GGA, CBA, and SAA under dataset 4.

	TANBS	time (s)	GGA	time (s)	CBA	time (s)	SAA	time (s)
4.1	17335	0.35	8368.0	1477.1	15935	0.043	14743	0.002
4.2	17588	0.38	7442.3	1632.5	15340	0.045	15848	0.001
4.3	14841	0.37	7595.0	1537.7	13541	0.038	12814	0.002
4.4	16918	0.35	7067.4	1859.9	13518	0.035	14178	0.001
4.5	14504	0.34	7396.8	1327.8	12793	0.032	15449	0.002
4.6	19265	0.42	8187.4	1724.5	16865	0.046	18589	0.001
4.7	17751	0.38	7650.2	1372.3	17541	0.057	17062	0.002
4.8	19327	0.38	7773.4	1362.5	17964	0.038	19542	0.001
4.9	18018	0.45	7085.8	1687.0	16885	0.039	17594	0.001
4.10	13107	0.41	6853.8	1466.9	12561	0.038	13875	0.002
4.11	16776	0.37	6547.0	1734.4	14427	0.033	14301	0.002
4.12	15218	0.36	7190.0	1622.9	13396	0.033	16050	0.002
4.13	18341	0.37	7935.2	1629.5	18891	0.043	19047	0.002
4.14	14956	0.36	6854.6	1717.1	14022	0.032	14416	0.002
4.15	16443	0.38	7537.4	1269.0	16421	0.036	15518	0.002

TABLE 6. The average system revenues and average run times of TANBS, TADUA, and CRA under dataset 1.

	TANBP	time(s)	TAUDA	time(s)	CRA	time(s)
1.1	2800	0.016	2614	0.015	2043	0.134
1.2	2921	0.019	2750	0.018	2402	0.119
1.3	3163	0.019	2909	0.017	2205	0.168
1.4	2715	0.021	2666	0.021	1504	0.077
1.5	3070	0.019	2996	0.017	2107	0.086
1.6	2682	0.019	2500	0.018	1817	0.064
1.7	2874	0.018	2805	0.017	1437	0.104
1.8	3118	0.017	3065	0.017	2517	0.041
1.9	2984	0.016	2715	0.015	1274	0.138
1.10	2084	0.015	2112	0.015	2176	0.066
1.11	3105	0.017	3038	0.016	1787	0.094
1.12	2441	0.018	2023	0.017	1035	0.078
1.13	2611	0.022	2704	0.021	1502	0.092
1.14	3084	0.018	3076	0.017	1282	0.093
1.15	2534	0.021	2436	0.021	1652	0.074

TABLE 7. The average system revenues and average run times of TANBS, TADUA, and CRA under dataset 2.

	TANBS	time(s)	TAUDA	time(s)	CRA	time(s)
2.1	61248	6.93	60755	3.37	26236	249.5
2.2	56985	7.25	54736	3.55	28977	234.2
2.3	60247	6.55	59656	3.28	26953	203.9
2.4	67673	6.50	67799	3.13	35482	232.5
2.5	65251	6.94	64906	3.20	29019	323.1
2.6	63240	6.63	62409	3.18	29439	214.2
2.7	59680	6.42	59091	3.16	34008	215.1
2.8	63547	6.73	62624	3.34	23556	274.6
2.9	60548	6.69	59958	3.32	28629	287.4
2.10	63969	7.09	62939	3.23	30634	299.3
2.11	60684	6.94	60177	3.24	30790	227.3
2.12	55984	7.64	55689	3.42	23723	239.1
2.13	62590	7.76	62360	3.80	36573	273.7
2.14	62225	7.09	61604	3.54	23373	254.9
2.15	65272	7.16	65294	3.45	25407	278.2

average system revenue is obtained when CP is 0.75 and MP is 0.05. The other parameters of GGA were set as follows: the size of the population was 100, and the maximal number of iterations was 10000.

Under dataset 1, dataset 2, dataset 3 and dataset 4, the average system revenues and average run time of TANBS, GGA, CBA, and SAA are shown in Table 2, 3, 4 and 5, respectively. TANBS, CBA, and SAA were run 100 times, and GGA was run 30 times. Then, their average system revenues and average run times were calculated.

From the results in Table 2, 3, 4 and 5, it can be seen that in most cases, the average system revenues obtained by TANBS are better than those of the other 3 algorithms. It can also be concluded that the average run time of TANBS is shorter than that of GGA. Among them, because the search space of GGA is too large, its system revenues are the poorest,

TABLE 8. The average system revenues and average run times of TANBS, TADUA, and CRA under dataset 3.

	TANBS	time (s)	TAUDA	time (s)	CRA	time (s)
3.1	11021	0.36	10350	0.24	6281	5.84
3.2	11223	0.34	10436	0.24	8874	2.35
3.3	10827	0.37	10188	0.23	6028	4.61
3.4	13013	0.41	12465	0.18	7541	4.12
3.5	13482	0.33	12705	0.20	7640	3.30
3.6	11682	0.34	10511	0.18	6747	5.71
3.7	13002	0.36	11221	0.21	8522	4.29
3.8	11439	0.33	11127	0.21	9146	3.47
3.9	12051	0.37	10986	0.19	7312	3.68
3.10	13516	0.36	11288	0.23	7682	2.56
3.11	12421	0.36	11974	0.24	7015	6.13
3.12	12617	0.43	10847	0.22	6079	4.48
3.13	10587	0.44	9661	0.24	5913	5.39
3.14	13766	0.36	12818	0.17	8045	5.97
3.15	12253	0.33	10546	0.27	7032	3.47

TABLE 9. The average system revenues and average run times of TANBS, TADUA, and CRA under dataset 4.

	TANBS	time (s)	TAUDA	time (s)	CRA	time (s)
4.1	17335	0.35	15787	0.15	11390	2.08
4.2	17588	0.38	15389	0.17	10778	2.25
4.3	14841	0.37	13572	0.17	9568	2.01
4.4	16918	0.35	13511	0.12	11553	1.42
4.5	14504	0.34	12790	0.18	10204	1.42
4.6	19265	0.42	16877	0.16	13330	1.80
4.7	17751	0.38	17539	0.12	14870	2.16
4.8	19327	0.38	17952	0.18	14498	1.48
4.9	18018	0.45	16894	0.13	11224	3.01
4.10	13107	0.41	12305	0.14	9700	2.34
4.11	16776	0.37	13680	0.14	9153	3.67
4.12	15218	0.36	13398	0.13	12032	1.63
4.13	18341	0.37	18320	0.16	11295	3.31
4.14	14956	0.36	13617	0.16	11171	1.69
4.15	16443	0.38	16426	0.12	9923	2.80

and its run time is longest. The run time of SAA is shortest, but the service agents in SAA are cooperative; it can only allocate tasks but cannot distribute utilities, so the individual rationality of the service agent cannot be guaranteed in SAA. The bidding and pricing in the CBA algorithm can provide decision support for the utility distribution, but the task allocation result in CBA is not stable; that is, the service agent may give up the current cooperative task and choose another one that can bring it greater individual utility. The ''best response strategy'' in the first and fifth steps of the TANBS algorithm can ensure the stability of the final task allocation results. Therefore, from the experimental results, it can be further seen that the TANBS algorithm can not only solve the problem of utility distribution and maintain the stability of task allocation results but also obtain higher total system revenue in less running time.

B. EFFECTIVENESS VERIFICATION OF THE UTILITY DISTRIBUTION SCHEME IN TANBS

In the task allocation model of coalition skill game, the service agent is rational, so to obtain a stable task allocation result, it is necessary to guarantee the individual rationality. In addition, for rational service agents, to obtain a higher system revenue, distributing the utilities according to the service agents' power values is necessary. Therefore, this paper adopts the utility distribution strategy based on Nash bargaining solution. This section examines the influences of the utility distribution strategies on system revenue. The higher the system revenue is, the more reasonable and fair the utility distribution strategy is.

A new algorithm, called the Task Allocation Distributing Utilities Averagely (TADUA), was obtained by modifying TANBS so that the tasks distribute their utilities on average. The other steps were the same as TANBS. Another utility distribution scheme is CRA (consensus-based reward allocation, CRA). Under dataset 1, 2, 3 and 4, the three algorithms are run 100 times, and the average system revenues and average run times are shown in Table 6, 7, 8 and 9. From the simulation results, it can be seen that the total system revenues of TANBS are better than those of TADUD and CRA. This shows that by using the utility distribution strategy based on the Nash bargaining solution, even if the service agents are self-interested, the total system revenue can be improved.

If the service agent is cooperative, then how the utility is distributed after completing the task will not affect the total system revenue. However, for self-interested service agents, their goals are to maximize their own utilities. Therefore, whether the utility distribution is reasonable and fair will affect the improvement of the total revenue of the system to a certain extent. The fairness of utility distribution does not mean average distribution. For example, the utilities in the TADUA algorithm are distributed equally. Of course, it cannot be simply viewed that whoever has a higher cost of performing tasks can obtain a larger share of revenue, such as in the CRA algorithm. In CRA, the fairness of the RoR is not conducive to the improvement of system revenue. In contrast, to improve the total system benefits, tasks should be assigned to service agents with less cost. It can also be seen from the results in Table 6, 7, 8 and 9 that, from the perspective of the total system revenue, the effect of CRA is poor, and the running time is long, but TANBS can obtain a higher system revenue within a shorter time. This is because the TANBS algorithm not only considers the cost of completing the task but also considers the relationship between the supply and demand of the service.

VI. SUMMARY AND FUTURE WORK

This paper presents a task allocation algorithm for self-interested agents in coalition skill games. The service agents in this game are self-interested, and they always select tasks that can bring them the greatest individual revenues. The goal of the task is to obtain all the needed skills. Through the adjustment of its utility distribution scheme according to the Nash bargaining solution of the needed skills, the task distributes more shares of utility for current ''short supply'' skills. Based on this concept, this paper proposes the TANBS algorithm to solve the problems of utility distribution and task allocation including self-interested agents. Compared with

the existing algorithms, the utility distribution scheme based on the Nash bargaining solution not only considers the cost of completing tasks but also considers the right values of the service agents in the whole task allocation system, so its utility distribution result is fairer and more reasonable. This further ensures that the task allocation results can not only have stronger stability but also obtain higher total system revenue. TANBS solves the problem that the existing algorithm either cannot distribute the utility, or the utility distribution result is unreasonable, which leads to lower total system revenue and unstable task allocation results for rational service agents. The final simulation results verify its effectiveness. Future research work will focus on the following two aspects. First, this paper only considers the allocation of static tasks, but there are many dynamic task allocation problems in practical applications, such as Witkey and crowdsourcing. Next, we will study the utility distribution and task allocation in these open and dynamic environments. Second, a standard database of coalition skill games needs to be established.

REFERENCES

[1] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2296–2319, Sep. 2016.

[2] T. V. Le, A. Stathopoulos, T. Van Woensel, and S. V. Ukkusuri, "Supply, demand, operations, and management of crowd-shipping services: A review and empirical evidence," *Transp. Res. C, Emerg. Technol.*, vol. 103, pp. 83–103, Jun. 2019.

[3] T. Yong-Xin, Y. Ye, C. Yu-Rong, C. Lei, and W. Guo-Ren, "Survey on spatiotemporal crowdsourced data management techniques," *J. Softw.*, vol. 28, no. 1, pp. 35–58, 2017.

[4] Y. Xin, D. Gerberry, and W. Just, "Open-minded imitation can achieve near-optimal vaccination coverage," *J. Math. Biol.*, vol. 79, no. 4, pp. 1491–1514, Sep. 2019.

[5] H. M. Jin, L. Su, D. Y. Chen, H. P. Guo, K. Nahrstedt, and J. H. Xu, "Thanos: Incentive mechanism with quality awareness for mobile crowd sensing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1951–1964, Aug. 2019.

[6] M. Drechsler, "Performance of input- and output-based payments for the conservation of mobile species," *Ecol. Econ.*, vol. 134, pp. 49–56, Apr. 2017.

[7] I. Caragiannis and A. A. Voudouris, "The efficiency of resource allocation mechanisms for budget-constrained users," in *Proc. ACM Conf. Econ. Comput.* Ithaca, NY, USA: Cornell Univ., Jun. 2018, pp. 681–698.

[8] Y. Bachrach and J. S. Rosenschein, "Coalitional skill games," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, Estoril, Portugal, 2008, pp. 1023–1030.

[9] Y. Bachrach, D. C. Parkes, and J. S. Rosenschein, "Computing cooperative solution concepts in coalitional skill games," *Artif. Intell.*, vol. 204, pp. 1–21, Nov. 2013.

[10] Q. X. An, Y. Wen, T. Ding, and Y. L. Li, "Resource sharing and payoff allocation in a three-stage system: Integrating network DEA with the Shapley value method," *OMEGA-Int. J. Manage. Sci.*, vol. 85, pp. 16–25, Jun. 2019.

[11] Y. Bachrach, E. Elkind, E. Malizia, R. Meir, D. Pasechnik, J. S. Rosenschein, J. Rothe, and M. Zuckerman, "Bounds on the cost of stabilizing a cooperative game," *J. Artif. Intell. Res.*, vol. 63, pp. 987–1023, Dec. 2018.

[12] J. Hou, S. Luo, W. Xu, and L. Wang, "Fairness-based multi-task reward allocation in mobile crowdsourcing system," *IET Commun.*, vol. 13, no. 16, pp. 2506–2511, Oct. 2019.

[13] M.-L. Fu, H. Wang, and B.-F. Fang, "Coalitional skill games for self-interested robots with SVO," *Int. J. Robot. Autom.*, vol. 33, no. 5, pp. 1–11, 2018.

[14] B. C. Schipper, "Dynamic exploitation of myopic best response," *Dyn. Games Appl.*, vol. 9, no. 4, pp. 1143–1167, Dec. 2019.

[15] B. Swenson, C. Eksin, S. Kar, and A. Ribeiro, "Distributed inertial best-response dynamics," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4294–4300, Dec. 2018.

[16] N. Kamra, U. Gupta, K. Wang, F. Fang, Y. Liu, and M. Tambe, "Deep fictitious play for games with continuous action spaces," in *Proc. 18th Int. Conf. Auton. Agents Multi-Agent Syst.*, Montreal, QC, Canada, May 2019, pp. 2042–2044.

[17] B. Swenson, S. Kar, and J. Xavier, "A computationally efficient implementation of fictitious play for large-scale games," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, Paris, France, Aug. 2015, pp. 1–16.

[18] T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings, "Coalition structure generation: A survey," *Artif. Intell.*, vol. 229, pp. 139–174, Dec. 2015.

[19] T. Genin and S. Aknine, "Coalition formation strategies for self-interested agents in task oriented domains," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, Toronto, ON, Canada, Aug. 2010, pp. 205–212.

[20] Y. Bachrach, R. Meir, K. Jung, and P. Kohli, "Coalitional structure generation in skill games," in *Proc. 24th AAAI Conf. Artif. Intell. (AAAI)*, Atlanta, GA, USA, Jul. 2010, pp. 703–708.

[21] Y. Liu, G.-F. Zhang, Z.-P. Su, F. Yue, and J.-G. Jiang, "Using computational intelligence algorithms to solve the coalition structure generation problem in coalitional skill games," *J. Comput. Sci. Technol.*, vol. 31, no. 6, pp. 1136–1150, Nov. 2016.

[22] D. Li, Q. Yang, W. Yu, D. An, Y. Zhang, and W. Zhao, "Towards differential privacy-based online double auction for smart grid," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 971–986, Aug. 2019.

[23] M. N. Faqiry and S. Das, "Double auction with hidden user information: Application to energy transaction in microgrid," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 11, pp. 2326–2339, Nov. 2019.

[24] J. N. Qin, X. Fu, S. M. Peng, J. Huang, and S. Huang, "Asymmetric bargaining model for water resource allocation over transboundary rivers," *Int. J. Environ. Res. Public Health*, vol. 16, no. 10, pp. 1–23, 2019.

[25] L. Feng, Q. Yang, K. Kim, and K. S. Kwak, "Two-timescale resource allocation for wireless powered D2D communications with self-interested nodes," *IEEE Access*, vol. 7, pp. 10857–10869, 2019.

[26] Y. Liang, W. Wei, and C. Wang, "A generalized Nash equilibrium approach for autonomous energy management of residential energy hubs," *IEEE Trans. Ind. Informat.*, vol. 15, no. 11, pp. 5892–5905, Nov. 2019.

[27] N. Cao, S. Brahma, B. Geng, and P. K. Varshney, "Optimal auction design with quantized bids for target tracking via crowdsensing," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 5, pp. 847–857, Oct. 2019.

[28] H. Cai, Y. Zhu, Z. Feng, H. Zhu, J. Yu, and J. Cao, "Truthful incentive mechanisms for mobile crowd sensing with dynamic smartphones," *Comput. Netw.*, vol. 141, pp. 1–16, Aug. 2018.

[29] P. Zhou, W. Chen, S. Ji, H. Jiang, L. Yu, and D. Wu, "Privacy-preserving online task allocation in edge-computing-enabled massive crowdsensing," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7773–7787, Oct. 2019.

[30] P. Xiong, D. Zhu, L. Zhang, W. Ren, and T. Zhu, "Optimizing rewards allocation for privacy-preserving spatial crowdsourcing," *Comput. Commun.*, vol. 146, pp. 85–94, Oct. 2019.

[31] D. Yuan, Q. Li, G. Li, Q. Wang, and K. Ren, "PriRadar: A privacy-preserving framework for spatial crowdsourcing," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 299–314, 2020.

[32] C. Martin, D. Ivana, G. Iosifidis, and M. Bouroche, "Adaptive reward allocation for participatory sensing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–15, Jan. 2018.

[33] T. C. Service and J. A. Adams, "Coalition formation for task allocation: Theory and algorithms," *Auton. Agents Multi-Agent Syst.*, vol. 22, no. 2, pp. 225–248, Mar. 2011.

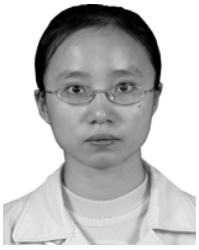
[34] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robot. Auton. Syst.*, vol. 90, pp. 55–70, Apr. 2017.

[35] K. R. Apt and S. Simon, "A classification of weakly acyclic games," *Theory Decis.*, vol. 78, no. 4, pp. 501–524, Apr. 2015.

[36] R. Kolisch and A. Sprecher, "PSPLIB—A project scheduling problem library: OR software-ORSEP operations research software exchange program," *Eur. J. Oper. Res.*, vol. 96, no. 1, pp. 205–216, 1997.

[37] B. P. Myszkowski, E. M. Skowroński, and K. Sikora, "A new benchmark dataset for multi-skill resource-constrained project scheduling problem," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Łódź, Poland, Nov. 2015, pp. 129–138.

[38] P. B. Myszkowski, M. E. Skowroński, Ł. P. Olech, and K. Oślizło, "Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem," *Soft Comput.*, vol. 19, no. 12, pp. 3599–3619, Dec. 2015.
[39] T. Hegazy, A. K. Shabeeb, E. Elbeltagi, and T. Cheema, "Algorithm for scheduling with multiskilled constrained resources," *J. Construct. Eng. Manage.*, vol. 126, no. 6, pp. 414–421, Dec. 2000.
[40] A. Santosm and P. A. Tereso, "On the multi-mode, multi-skill resource constrained project scheduling problem—A software application," in *Soft Computing in Industrial Applications*, no. 96. Berlin, Germany: Springer, 2011, pp. 239–248.
[41] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Apr. 2005, pp. 1145–1150.



MING L. FU was born in Sichuan, China, in 1981. She received the B.Eng. and M.Eng. degrees in network engineering and pattern recognition and intelligent systems from the Sichuan University of Science & Engineering, and the Ph.D.Eng. degree from the Hefei University of Technology, in 2018. She is currently a Teacher with the School of Computer Science and Technology, Huaibei Normal University, Huaibei, China. Her current research interests include multiagent systems, game theory, and emotional intelligence.

LV Q. CHEN was born in Anhui, China, in 1982. He received the B.Eng. degree in electronic information engineering from Huaibei Normal University, Huaibei, China, and the M.Eng. degree in pattern recognition and intelligent systems from the Sichuan University of Science & Engineering. He is currently a Teacher with the School of Computer Science and Technology, Huaibei Normal University. His current research interests include multiagent systems, evolutionary algorithms, and machine learning.

YU J. WAN received the B.Eng. degree from the School of Information, Huaibei Normal University, Huaibei, China, in 2019, where he is currently pursuing the master's degree in software engineering. His research interests include game theory and evolutionary computation.

DE B. CHEN received the Ph.D. degree from the School of Computer Science, Nanjing University of Science and Technology, Nanjing, China, in 2008. He is currently a Full Professor with Huaibei Normal University, Huaibei, China. His current research interests include evolutionary computation, global optimization, multiobjective optimization, and neural networks.