Task Allocation on Layered Multi-Agent Systems: When Evolutionary Many-Objective Optimization Meets Deep Q-Learning

Mincan Li, Zidong Wang, *Fellow, IEEE*, Kenli Li, Xiangke Liao, Kate Hone, and Xiaohui Liu

***Abstract*—This paper is concerned with the multi-task multi-agent allocation problem via many-objective optimization for multi-agent systems (MASs). First, a novel layered MAS model is constructed to address the multi-task multi-agent allocation problem that includes both the original task simplification and the many-objective allocation. In the first layer of the model, the deep Q-learning method is introduced to simplify the prioritization of the original task set. In the second layer of the model, the modified shift-based density estimation (MSDE) method is put forward to improve the conventional Strength Pareto Evolutionary Algorithm 2 (SPEA2) in order to achieve many-objective optimization on task assignments. Then, an MSDE-SPEA2-based method is proposed to tackle the many-objective optimization problem with objectives including task allocation, makespan, agent satisfaction, resource utilization, task completion, and task waiting time. As compared with existing allocation methods, the developed method in this paper exhibits an outstanding feature that the task assignment and the task scheduling are carried out simultaneously. Finally, extensive experiments are conducted to 1) verify the validity of the proposed model and the effectiveness of two main algorithms; and 2) illustrate the optimal solution for task allocation and efficient strategy for task scheduling under different scenarios.**

***Index Terms*—Evolutionary computation, many-objective optimization, multi-agent systems, task allocation, deep Q-learning.**

I. Introduction

THE past two decades have witnessed a great deal of research attention devoted to multi-agent systems (MASs) [7], [8], [13], [16], [19], [31], [55], [57]. Task allocation in MASs aims at finding appropriate agents who can independently or cooperatively conduct specific tasks [2], [25]. In the existing literature, task allocation has been classified into various categories according to three different perspectives: 1) from the tasks' viewpoint, task allocations can be divided into single-task and multi-task assignments; 2) in terms of the agents (more generally robots), there are single-robot and multi-robot allocations; and 3) by means of assignment style, task allocations can be grouped into instantaneous and time-extended ones. Based on the aforementioned classifications, the two most important assignments, namely, single-task multi-robot tasks and instantaneous assignment (ST-MR-IA) [48] as well as multi-task and multi-robot allocation (MTMRA), have attracted persistent research attention concerning the task allocation issue for MASs.

The purpose of the ST-MR-IA problem is to establish a unique agent coalition for every task. It is worth noting that, for each task, one agent cannot join more than one coalition at a time. Accordingly, it is crucial to find an ideal coalition formation algorithm [52]. So far, a number of coalition formation algorithms have been developed (see e.g. [53]) and applied in many areas such as disaster response [26] and urban search with rescue environment [43]. Nevertheless, existing coalition formation algorithms might be incapable of dealing with the MTMRA problem due mainly to the complexity of constraints [47], [49].

In order to cope with the foregoing complexity issue, a number of approaches have been put forward in the literature to tackle the MTMRA problem. Generally, the MTMRA procedure in MASs can be carried out within noncooperative or cooperative environments. In the noncooperative circumstance, selfish agents pursue their own profits by providing resources as seller agents or consuming resources as consumer agents, regardless of others' rewards in the process of task assignment, and such a behavior is referred to as the market-based task allocation. This kind of allocation method, as mentioned in [14], has advantages in improving the allocation efficiency and enhancing the agent cooperation with satisfactory total utility. In the cooperative case, breaks and conflicts often occur in the assignment process. In order to mitigate the occurrence of undesired breaks and conflicts, many negotiation-based allocation methods have been proposed with examples including bilateral negotiation [20], time-constrained negotiation in dynamic open grid environments [29], and resource negotiation in manufacturing systems [30].

The aforementioned allocation algorithms (i.e. market- and negotiation-based allocation methods) have proven to be effective in certain circumstances for MASs. Nevertheless, using these traditional algorithms alone would not be of much help in dealing with the increasingly demanding requirements as well

This work was supported in part by the the National Key Research and Development Program of China under Grant 2020YFB2104000, the National Natural Science Foundation of China under Grants 61625202, 61751204 and 61860206011, the European Union's Horizon 2020 Research and Innovation Programme under Grant 820776 (INTEGRADDE), the Royal Society of the UK, and the Alexander von Humboldt Foundation of Germany. (*Corresponding author: Kenli Li.*)

M. Li and K. Li are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China; and also with the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China. (emails: limc@hnu.edu.cn; lkl@hnu.edu.cn)

Z. Wang, K. Hone and X. Liu are with the Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, United Kingdom. (emails: Zidong.Wang@brunel.ac.uk; Kate.Hone@Brunel.ac.uk; Xiaohui.Liu@brunel.ac.uk)

X. Liao is with the Collaborative Innovation Center of High Performance Computing, National University of Defense Technology, Changsha 410073, China. (e-mail: xkliao@nudt.edu.cn).

as the complex-dependency-induced constraints on practical task assignments. Fortunately, the nowadays deep learning algorithms are well known for their capabilities of handling complicated situations, and therefore are well suited to be applied to deal with allocation issues for MASs. Some up-to-date deep learning methods include, but are not limited to, Deep Q-Network (DQN) [42], Convolutional Neural Network [9], Feedforward Neural Network [56] and Recurrent Neural Network [1], which are ideal candidate algorithms for further improving the efficiency of task allocation and execution [33]. These networks in deep-learning process usually combined with back-propagation [22], and Stochastic Gradient Descent (SGD) [3] methods. Especially, deep Q-learning method usually gets excellent performance in dealing with uncertain status and exponential solution searching space in task allocation problem. On the one hand, for the exponential searching space of the task dividing problem, deep Q-learning has a deep Q-network to training and searching, which help to accelerate finding the optimal solution. On the other hand, for the uncertain status in task dividing, the complex computation in every iteration can be solved by the approximate calculation in the Q-network of deep Q-learning method.

In addition to the allocation, scheduling serves as another crucially important aspect for task allocation for MASs [17]. So far, a variety of scheduling methods have been developed for task allocation in order to ensure short makespan, high success rate and few conflicts [34]. Unfortunately, the allocation and scheduling issues have seldom been taken into *simultaneous* consideration and, so far, the relevant results have been scattered in the literature. For MASs, the task allocation is essentially a multi/many-objective issue that involves multiple yet possibly conflicting performance requirements on makespan, agent satisfaction, resource utilization, task completion, and task waiting time, etc. Clearly, it is difficult to meet all the requirements via solving a single-objective assignment problem and, therefore, there is a practical need to study the multi/many-objective assignment issues for MASs in the context of optimization. Over the past few decades, the evolutionary algorithms (EAs) have shown superiority in dealing with multi/many-objective optimization problems and stirred a great deal of research attention [39]. As such, a seemingly natural idea is to utilize the latest EAs to cope with the typically multi-objective multi-constraint allocation/scheduling issues with MASs.

In fact, the evolutionary multi-objective optimization (EMO) algorithms have begun to receive some initial research interest in accommodating different task requirements in MASs. For example, in [51], a novel EA-based classification algorithm has been explored to develop an agent decision-making strategy. In [5], multi-objective optimization methods have been introduced to optimize the prioritized tasks in MASs. A hybrid EMO algorithm has been applied in [18] for the stochastic flow shop scheduling problem with aim to minimize the makespan and the total tardiness. Unfortunately, the EMO algorithms themselves might be unable to effectively handle optimization problems with *more than three objectives*, which are customarily referred to as the *many*-objective optimization problems. As such, it makes practical sense to initiate an investigation on the constrained task allocation problem by developing an evolutionary many-objective optimization algorithm.

Motivated by the above discussions, the main aim of this paper is to design an evolutionary many-objective optimization algorithm for task allocation of MASs. This problem appears to be especially difficult as we are inevitably confronted with the following three essential challenges: 1) how to design a dedicated encoding scheme for the operations of crossing and mutating in accordance with a series of performance requirements on the task allocation? 2) how to improve, to a great extent, the traditional EAs and EMO algorithms via designing strict density estimation methods? and 3) how to deal with the complex dependency relationships (among tasks in the original task database) which would bring in substantial difficulties in designing evolutionary many-objective optimization algorithm? It is, therefore, the main purpose of this paper to meet the challenges.

This paper is concerned with multi-task multi-agent allocation problem via many-objective optimization for MAS, where the five objectives include admissible makespan, agent satisfaction, resource utilization, task completion, and task waiting time in the task allocations. It is worth noting that agents are purposely divided into two layers with aim to improve the allocation efficiency. The main contributions of this paper are highlighted as follows:

1) A novel layered MAS model is designed for two main allocation purposes: the first layer is established to divide tasks and simplify prioritizations, and the second layer is constructed for task allocation and scheduling.
2) A deep Q-learning algorithm is, for the first time, to deal with the task dividing problem so that the task prioritization can be effectively simplified to facilitate the subsequent assignment.
3) A modified shift-based density estimation (MSDE) strategy is designed for the Strength Pareto Evolutionary Algorithm 2 (SPEA2) for the purpose of implementing many-objective optimization in the task assignments with aim to a) play an adequate trade-off between the addressed five objectives; b) maintain the diversity of allocation solutions in the elite selection process; and c) achieve a satisfactory overall accuracy.
4) The proposed MSDE-SPEA2-based method enables us to simultaneously conduct task assignments and task scheduling for the entire task set, where the assignment results include not only the specific agent groups for every task but also the execution order for the task set.

The rest of this paper is organized as follows. Section II discusses the related work of optimization algorithms for task allocation and assignment problems for MASs. The problem formulation is given in Section III. In Section IV, the models and the proposed algorithms for the task allocation problem of MASs are provided. Section V presents the experiment results, related analysis, and the comparisons with some other state-of-the-art approaches. We draw conclusions and point out future research topics in Section VI.

II. Related Work

In recent years, the multi-task allocation problem has been extensively investigated for MASs from various aspects including rewards of agents [11], searching efficiency of solutions [50] and specific allocation of applications [23]. The profits dividing issue has been paid particular attention within the task allocation framework especially in the noncooperative environment. For example, the auction algorithms [45] have been applied to deal with the market-based assignment issues with no agent cooperation, and satisfactory results have been obtained. Nevertheless, a single point of failure in the auction algorithms could result in a broken system and, in order to prevent such a single point of failure from happening, the negotiation-based methods have been introduced to deal with the multi-task allocation problem. Negotiation-based methods (e.g. uncertain deadline negotiation [20], time-constraint negotiation [29] and game theory-based negotiation [12]) are known to be capable of effectively avoiding breaks or conflicts.

In available literature, the self-organization behavior of agents has been considered in some task allocation methods in the noncooperative environment. For instance, it has been shown in [54] that agents can find suitable tasks and services according to self-incentive rules. In the cooperative circumstance, most game-theory-based task allocations [58] have focused on the collective profits dividing, see e.g. [32] for the proposed Shapley principle for collective profits dividing in task allocation. With the growing size of the database, the number of coalition structures of agents is increased at an exponential rate which, in turn, leads to low searching speed. In this regard, the latest EAs could be exploited to accelerate the process of searching coalition structures in task allocations.

Due to their distinct merits of strong robustness, wide applicability and rapid searching capability, the EAs have shown tremendous potential in dealing with the global multi-agent optimization problems in practical applications [21]. For example, in [6], the task deadlock problem (caused by agent decision) has been avoided through dynamic and variable pheromone placement methods. In [24], heuristic initialization rules and repair strategy mechanisms have been applied to a genetic algorithm with a fast convergence. Nevertheless, traditional EAs alone might be unable to simultaneously tackle the multiple objectives in task allocation. In order to coordinate the competitive and possibly conflicting objectives, the EMO algorithms have been introduced to address the agent task allocation problems. One distinguished feature of the EMO algorithms is their ability to obtain a Pareto approximation set with fast convergence and superior diversity, thereby offering a set of optimal solutions [15]. On this account, the EMO algorithms have been widely applied in MASs, see e.g. [51] for the optimization of the agent decision making process, and other applications include mobile tracking [44], agents optimal path searching [27] and agent cooperation with automated negotiation [28].

As discussed previously, deep learning methods are in an ideal position to be deployed to tackle the task allocation problems that are complicated by the sophisticated prioritization and dependency among tasks. Furthermore, deep Q-learning (DQL) algorithms, which combine the merits of deep learning and reinforcement learning, are particularly suitable for addressing the complicated task allocation problems. In fact, the importance of DQL algorithms has already begun to be recognized from both industry and academy with respect to general optimization problems. For example, a DQL-based resource assignment method has been put forward in [4] to improve the ultra-reliable low-latency communication service in the industrial wireless network, where the proposed method could overcome the difficulty resulting from the dynamic and complex variations of network nodes. In [60], a DQL-based transmission mechanism has been proposed to maximize the system throughput by a suitable strategy on buffer transmission. Very recently, a deep reinforcement learning algorithm has been developed [41] on the agent communication learning problem with hope to reduce redundant message sending.

III. Problem Formulation

In this section, the task allocation problem is formulated with detailed descriptions on the task, the agent and the allocation targets. Let us start by describing some key concepts required for formulating the evolutionary many-objective optimization problem for task allocation on layered MASs.

Task: The ith task, denoted as t_i, is a 7-tuple given by

$$(ID_{t_i}, TY_{t_i}, IF_{t_i}, CT_{t_i}, RT_{t_i}, V_{t_i}, TD_{t_i})$$

where ID_{t_i} is the identification of the task (i.e. $ID_{t_i} = i$), TY_{t_i} represents the task type,

$$IF_{t_i} = \{\{id_1, p_{(t_i,t_{id_1})}\}, \{id_2, p_{(t_i,t_{id_2})}\}, \ldots, \{id_m, p_{(t_i,t_{id_m})}\}\}$$

stands for the prioritization of t_i, $CT_{t_i} = \{ct_1^{t_i}, ct_2^{t_i}, \ldots, ct_l^{t_i}\}$ describes t_i's requirement vectors of capability, $RT_{t_i} = \{rt_1^{t_i}, rt_2^{t_i}, \ldots, rt_n^{t_i}\}$ denotes t_i's requirement vectors of resource, V_{t_i} means the maximum reward of finishing task t_i, and TD_{t_i} indicates the maximum duration time of t_i's implementation.

To be more specific, the task types include the dependent type $TY_{t_i} = 0$ and the independent type $TY_{t_i} = 1$. A task in the dependent type requires more than one agent to finish the task, while the independent task requires only one agent. Every task has its prioritization which is denoted as the vector IF_{t_i}. When t_i has been divided, t_i will have a positive influence on every t_{id_ξ} $(\xi = 1, 2, \ldots, m)$ by different probabilities. $(id_1, id_2, \ldots, id_m)$ includes *m* identifications of tasks that must be executed after t_i is finished. $\{id_\xi, p_{(t_i,t_{id_\xi})}\}$ $(\xi = 1, 2, \ldots, m)$ reflects the probability of the positive influence from task t_i on task t_{id_ξ} because of t_i's dividing. Here, the positive influence means that, when t_i has been divided, it is successful to combine t_i's subtask t_{i_λ} with t_{id_ξ}. Furthermore, CT_{t_i} has l kinds of capabilities and RT_{t_i} has n kinds of resources. In **Task**, if a task has no requirement for the η^{th} kind of capability or resource, the corresponding $ct_\eta^{t_i}$ and $rt_\eta^{t_i}$ will be "0".

First-Layer Agent: The jth-agent a_j on the first-layer is a 3-tuple $(ID_{a_j}, IF_{a_j}, LD_{a_j})$, where ID_{a_j} is the identification of the agent $(ID_{a_j} = j)$, $IF_{a_j} = \{\{id_1, p_{(a_1,a_{id_1})}\},$

$\{id_2, p_{(a_2,a_{id_2})}\}, \ldots, \{id_m, p_{(a_i,a_{id_m})}\}\}$ copies the information of IF_{t_i} and LD_{a_j} is a flag reflecting whether the task is divided or not, that is, $LD_{a_j} = 1$ (respectively, $LD_{a_j} = 0$) means that the task is divided (respectively, not divided).

Second-Layer Agent: The kth-agent a_k on the second-layer is a 4-tuple $(ID_{a_k}, RO_{a_k}, CA_{a_k}, RA_{a_k})$, where ID_{a_k} represents the identification of the agent ($ID_{a_k} = k$), RO_{a_k} is the role of agent a_k, $CA_{a_k} = \{ca_1^{a_k}, ca_2^{a_k}, \ldots, ca_l^{a_k}\}$ and $RA_{a_k} = \{ra_1^{a_k}, ra_2^{a_k}, \ldots, ra_n^{a_k}\}$ indicate agent a_k's capability vector and resource vector, respectively. Specifically, $RO_{a_k} = 0$ means that agent a_k is a dependent agent which must cooperate with at least one agent, and $RO_{a_k} = 1$ indicates that a_k is independent. An independent agent means that it can finish one task by cooperating with others or not.

The following definitions are useful in quantifying the requirements/objectives for the addressed allocation/scheduling problem.

Definition 1: The execution time of the task t_i, which is allocated to the agent coalition C_i, is denoted by τ_{t_i} and defined as follows:

$$\tau_{t_i} = TD_{t_i} \frac{1}{1 + \rho_{t_i,C_i}} \tag{1}$$

with

$$\rho_{t_i,C_i} = \frac{\sum_{\eta=1}^{l} (ct_\eta^{t_i} - \overline{ct}^{t_i})(\overline{ca}_\eta^{C_i} - \overline{ca}^{C_i})}{\sqrt{\sum_{\eta=1}^{l} (ct_\eta^{t_i} - \overline{ct}^{t_i})^2} \sqrt{\sum_{\eta=1}^{l} (\overline{ca}_\eta^{C_i} - \overline{ca}^{C_i})^2}} \tag{2}$$

where $\overline{ct}^{t_i}$ is the average value of t_i's capability requirements (i.e. $\overline{ct}^{t_i} = \frac{1}{l} \sum_{\eta=1}^{l} ct_\eta^{t_i}$), $\overline{ca}_\eta^{C_i}$ is the average value of all η^{th} capability requirements in the agent coalition C_i (i.e. $\overline{ca}_\eta^{C_i} = \frac{1}{|C_i|} \sum_{k=1}^{|C_i|} \overline{ca}_\eta^{a_k}$ and $\overline{ca}^{C_i} = \frac{1}{l} \sum_{\eta=1}^{l} \overline{ca}_\eta^{C_i}$), and ρ_{t_i,C_i} denotes the ability evaluation of agent members in coalition C_i.

Definition 2: The agent satisfaction index represents agent's evaluation of its reward. Agent a_k's satisfaction index in coalition C_i for task t_i is defined as

$$s_{a_k} = \frac{(V_{t_i} - cost_{C_i}) - (V_{t_i} - cost_{(C_i \setminus \{a_k\})})}{V_{t_i} - cost_{(a_k,t_i)}} \tag{3}$$

where s_{a_k} denotes agent a_k's satisfaction index, $cost_{C_i}$ indicates the cost for all agents in the coalition C_i to complete the task t_i, and $cost_{(C_i \setminus \{a_k\})}$ represents the cost for the agents except a_k in the coalition C_i to complete the task.

Task allocation problem in this paper is focused on balancing the five objectives in task allocation as mentioned, and the coalition cost is a part of the calculation in agent satisfaction. Hence, the complex calculation of coalition cost for every agent combination is replaced by data generator programming. The data generated by the data generator obey the superadditivity principle. When calculating the cost of every task, the related data will be accessed in the experiments database.

Definition 3: The *makespan* of a task set T is defined by

$$makespan = \sum_{i=1}^{|T|} \tau_{t_i} - \tau_0 \tag{4}$$

where τ_{t_i} is the task t_i's execution time and τ_0 refers to all overlapping execution time of tasks.

Definition 4: The resource utilization RU_{t_i} of the coalition C_i for task t_i is defined as follows:

$$RU_{t_i} = \frac{1}{n} \sum_{\eta=1}^{n} \frac{rt_\eta^{t_i}}{\frac{1}{|C_i|} \sum_{k=1}^{|C_i|} ra_\eta^{a_k}} \tag{5}$$

where $rt_\eta^{t_i}$ and $ra_\eta^{a_k}$ are defined in **Task** and **Second-layer agent**, respectively.

Definition 5: The task allocation success ratio of a task set T is defined by:

$$SR_T = \frac{|SU|}{|T|} \tag{6}$$

where $|SU|$ is the number of successfully allocated tasks, and $|T|$ indicates the size of the task set. An assignment is said to be successful if 1) the corresponding agent coalition meets the task's requirements; 2) there is no conflict among agents; and 3) the sum of t_i's waiting time and t_i's execution time does not exceed the maximum duration time TD_{t_i}.

Definition 6: The total task waiting time is defined as follows:

$$WT = \sum_{i=1}^{|T|} w\tau_{t_i} \tag{7}$$

where $w\tau_{t_i}$ represents the waiting time of task t_i.

In the second-layer model, we will design the scheduling order for all tasks which gives a sequence number for every task. Tasks can be executed according to their sequence number. Without loss of generality, t_i's sequence number is marked as SN_{t_i}. The waiting time $w\tau_{t_i}$ of t_i can be calculated when there are more than one task whose sequence number equals to $SN_{t_i} - 1$, otherwise, $w\tau_{t_i} = 0$. For example, if $SN_{t_i} = 5$ and $SN_{t_1} = SN_{t_2} = SN_{t_3} = 4$, the waiting time can be calculated by $w\tau_{t_i} = \max\{\tau_{t_1}, \tau_{t_2}, \tau_{t_3}\} - \min\{\tau_{t_1}, \tau_{t_2}, \tau_{t_3}\}$.

This paper aims to find an optimal allocation for tasks and design an ideal task scheduling to coordinate the following five objectives: 1) the makespan, 2) the agent satisfaction, 3) the resource utilization, 4) the task completion, and 5) the task waiting time in the task allocation.

IV. Main results

In this section, a two-layer MASs model is established for task assignments, followed by the presentation of the main algorithms.

A. Task Allocation Procedure Model of Layered MASs

The assignment procedure of task allocation on the layered MASs is displayed in Fig. 1, which includes two main steps. In the first step, the deep Q-learning method is introduced for the first-layer agents to simplify the prioritization of the original task set. In this step, the prioritization of tasks is reduced as much as possible. In the second step, the MSDE-SPEA2-based method is applied to harmonize the five objectives under the constraints.

It is worth noting that the information of first-layer agents relies on the task set T, and thus the number of first-layer agents equals to $|T|$. Different from the first-layer agent, the second-layer agents are given in the agent set A, and the

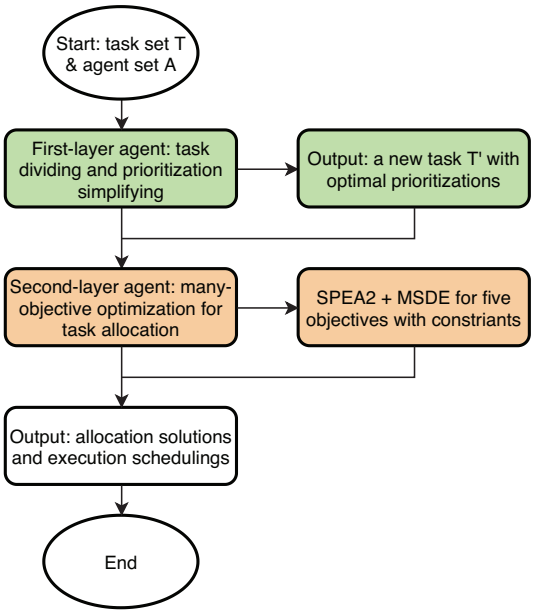


Fig. 1. Task allocation procedure model of layered MASs

number of second-layer agents is $|A|$. According to the layered allocation procedure in Fig. 1, our main algorithm is provided in Algorithm 1 as follows.

Algorithm 1 Main Algorithm

Input: An agent set A and a task set T

1: Simplify the prioritization of T by task dividing to generate a new task set T' on the first layer of the model.
2: Assign and schedule the new task set T' through the agent set A by means of MSDE-SPEA2-based algorithm under the constraints on the second layer of the model.

Output: Allocation coalition set AC and scheduling order O.

B. Algorithms for the Task Allocation

In this section, the corresponding algorithms for the task allocation problem are presented.

1) First-layer agents: task dividing and prioritization simplifying: The task dividing and prioritization simplifying problem is an uncertain problem, which is shown in Fig. 2. t_2, t_3 and t_4 can be executed when t_1 finishes. t_1 has three different probabilities of positive influence on t_2, t_3 and t_4. To be specific, if t_1 is divided, subtask t_{1_1} will be combined with task t_2 with probability $p_{(t_1,t_2)}$, and if the combination is successful, the new task is marked as t'_2. Then the prioritization of (t_1, t_2) is simplified. It should be mentioned that t_1 can be divided successfully, while, the success of tasks' combination is uncertain because of the corresponding probabilities. Therefore, it is an uncertain task dividing problem. Reflect this uncertain problem to our experiments, and there will be a parameter to indicate the current probability of the two tasks which will be combined. The parameter has a random value, and according to this value, the combinations can be tackled

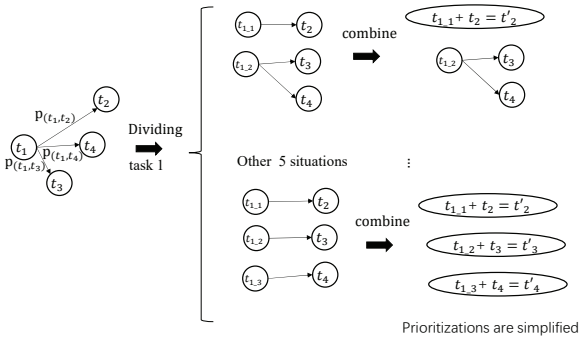


Fig. 2. An example of task dividing and prioritization simplifying

successfully or not. When all subtasks of t_1 are tackled, the prioritizations of t_1 is simplified. Similar situation applies to t_3 and t_4. In this case, t_1's dividing can lead to 7 different results. According to the result of t_1's dividing, the combined tasks can be divided but the others cannot. Based on this rule, whether a task can be divided is dependent on the situation of the last task and the corresponding probability of positive influence. Thus, in order to find out a task dividing set DS to simplify prioritization as much as possible, Algorithm 2 is introduced into this step.

In order to make Algorithm 2 easily understandable, we first introduce some notations involved in the Q-learning method as follows:

- *States*: the task queue of the current dividing set $DS^{'}$ is defined as a state. New tasks will be added into $DS^{'}$ and $DS^{'}$ will update the queue during every iteration step. The terminal state means that no task can be divided.
- *Actions*: an action means that new tasks are added into the set $DS^{'}$.
- *Rewards*: the rewards rely on the latest action. According to the action, several new tasks have been added into $DS^{'}$. The reward of this action is the extent of simplified prioritization caused by these new tasks.
- *Policy*: according to the greedy policy, the task which has the maximum rewards will be chosen in every iteration.

Next, let us introduce the deep Q-learning dividing algorithm in Algorithm 2. Q-network calculates the approximate action-value functions by initializing the replay memory set, the two value functions, and the first vector inputting. According to the greedy policy, the best action will be executed, and the replay memory set and the value of action-value functions will be updated. Through the gradient backpropagation on Q-network, the parameters are updated. Thus, when the maximum iteration is reached, the algorithm can output the task dividing set.

The optimal dividing set DS can be obtained by deep Q-learning dividing algorithm. Then, we can divide the tasks in DS and obtain the new task set T'. We introduce an auxiliary parameter p to facilitate the implementation of the dividing process. If task t_i is to be divided and $p > p_{(t_i,t_\eta)}$, the subtask t_{i_λ} of t_i will be combined with t_η successfully. If task t_i is divided and if $p < p_{(t_i,t_j)}$, the subtask will fail to be combined with task t_η. The dividing rules of t_j are set as:

Algorithm 2 Deep Q-learning Dividing Algorithm

Input: The set of first-layer agents, the iteration number t_{max}, action set AC, decay factor γ, budget B.

Initialize the replay memory set $M = \phi$, initialize Q-network, initialize action value Q and target action value $\widehat{Q}$.
for i=1 to B **do**
  Initialize $DS^{'}$ with an agent a_j, calculate its eigenvector $\phi(DS)$
  for i=1 to t_{max} **do**
    Input $\phi(DS^{'})$ into Q-network and choose a best action AC which has the maximum Q-value by greedy idea
    Execute the action AC and update $DS^{'}$, calculate the new eigenvector $\phi(DS^{'})$ and the reward R
    Update the replay memory set M by $\phi(DS^{'})$, AC, R, $\phi(DS^{'})$, and update $DS = DS'$
    Sample randomly from M and calculate the target action value $\widehat{Q}$
    if the current state is terminal state **then**
      $\widehat{Q} = R$
    else $\widehat{Q} \leftarrow R + \gamma \max Q(\phi(DS'), AC')$
    end if
    Execute the gradient descent with respect to the Q-network, then update the network parameters
  end for
end for

Output: The optimal dividing set DS.

a) if $p > p_{(t_i,t_\eta)}$, every non-zero $rt_\varsigma^{t_\eta}$ in RT_{t_η} will be updated by $rt_\varsigma^{t_\eta} = rt_\varsigma^{t_\eta} + p_{(t_i,t_\eta)} rt_\varsigma^{t_i}$, and the rule of the vector CT_{t_η} is same as RT_{t_η}, i.e. $ct_\varsigma^{t_\eta} = ct_\varsigma^{t_\eta} + p_{(t_i,t_j)} ct_\varsigma^{t_i}$;
b) if $p > p_{(t_i,t_\eta)}$, $V_{t_\eta} = p_{(t_i,t_\eta)} V_{t_i} + V_{t_\eta}$ and $TD_{t_\eta} = p_{(t_i,t_\eta)} TD_{t_i} + TD_{t_\eta}$; and
c) if $p \le p_{(t_i,t_\eta)}$ and $ct_\varsigma^{t_\eta} = 0$, we have $ct_\varsigma^{t_i} = 0$, and if $p \le p_{(t_i,t_\eta)}$ and $rt_\varsigma^{t_\eta} = 0$, we have $rt_\varsigma^{t_i} = 0$. By dividing all tasks in DS, a new task set T' is obtained.

2) Second Layer Agents: Task Allocation and Scheduling: With the new task set T' and the second-layer agent set A, the MSDE-SPEA2-based algorithm is proposed to deal with many-objective optimization problem for task allocation and scheduling. First, the five objective functions are defined as follows.

The *makespan* function f_1 is formulated by:

$$f_1(\tau) = \sum_{i=1}^{|T|} \tau_{t_i} - \tau_0 \tag{8}$$

where $\tau = \{\tau_{t_1}, \tau_{t_2}, \ldots, \tau_{t_{|T|}}\}$.

The task waiting time function f_2 is given by:

$$f_2(WT) = \sum_{i=1}^{|T|} w\tau_{t_i} \tag{9}$$

where $WT = \{w\tau_{t_1}, w\tau_{t_2}, \ldots, w\tau_{t_{|T|}}\}$.

The agent satisfaction function f_3 is as follows:

$$f_3(SA) = 1 \Big/ \left\{ \frac{1}{|T|} \sum_{i=1}^{|T|} \frac{\sum_{j=1}^{|C_i|} s_{a_k}}{|C_i|} \right\} \tag{10}$$

where s_{a_k} can be calculated by Equation (3).

The task completion function f_4 is defined as:

$$f_4(SR_T) = \frac{|T|}{|SU|} \tag{11}$$

The resource utilization function f_5 is:

$$f_5(RU) = 1/\{\frac{1}{|T|} \sum_{i=1}^{|T|} RU_{t_i}\} \tag{12}$$

where RU_{t_i} can be calculated by Equation (5).

Considering the five objective functions given above, our optimization function is

$$\text{minimize } F(OS)$$

where $F(OS) \triangleq (f_1(\tau), f_2(WT), f_3(SA), f_4(SR), f_5(RU))$ and OS denotes a solution of the task allocation.

Next, the constraints of task allocation are set as follows.

Type constraint: according to the task type and agent role defined in **Task** and **Second-Layer Agent**, every assignment should not violate the requirements of the task type and agent type, that is, there is a type constraint given as

$$g_1(OS) = 0, \tag{13}$$

where

$$g_1(OS) \triangleq \sum_{i=1}^{|T|} b_{t_i} + \sum_{k=1}^{|A|} b_{a_k}.$$

Here, b_{t_i} reflects whether the allocation coalition violates the requirements of t_i's type. If the requirements of t_i's type is violated, $b_{t_i} = 1$, otherwise, $b_{t_i} = 0$. b_{a_k} reflects whether the allocation coalition violates the a_k's type. If the requirements of a_k's type is violated, $b_{a_k} = 1$, otherwise, $b_{a_k} = 0$.

Time constraints: The sum of t_i's waiting time and t_i's execution time cannot be longer than the maximum duration time. Thus, for every task t_i, we have the following condition:

$$w\tau_{t_i} + \tau_{t_i} \le TD_{t_i} \tag{14}$$

where $w\tau_{t_i}$, τ_{t_i} and TD_{t_i} indicate the waiting time, execution time and the maximum duration time of task t_i, respectively. Then, one has the constraint function as follows:

$$g_2(OS) = 0 \tag{15}$$

where $g_2(OS) \triangleq \sum_{i=1}^{|T|} et_{t_i}$, in which et_{t_i} reflects if task t_i satisfies (14). If (14) is violated, $et_{t_i} = 1$, otherwise, $et_{t_i} = 0$.

Agent constraint: An agent cannot join more than one coalition at a time, and therefore we have the agent constraint as

$$g_3(OS) = 0 \tag{16}$$

where $g_3(OS) \triangleq \sum_{i=1}^{|T|} c_{t_i}$, in which c_{t_i} indicates the number of agent conflicts of task t_i.

Resource constraint: If one task is assigned to one coalition, the coalition should satisfy the requirement that at least one

agent can meet the condition $ra_\eta^{a_k} \geq rt_\eta^{t_i}$ $(\eta = 1, 2, \ldots, n)$. Therefore, we have

$$g_4(OS) > 0,\ \forall t_i \in T \tag{17}$$

where $g_4(OS) \triangleq \sum_{k=1}^{|C_i|} sr_{a_k}$, in which OS denotes one solution of the task allocation and sr_{a_k} reflects if the agent can meet the task's resource requirements. If a_k does not meet the requirements, $sr_{a_k} = 0$, otherwise, $sr_{a_k} = 1$. $\sum_{k=1}^{|C_t|} sr_{a_k}$ actually refers to the number of agents which can meet the task's resource requirements.

According to above constraint conditions, the constraint violation value can be calculated as

$$CV(OS) = g_1(OS) + g_2(OS) + g_3(OS) + \frac{1}{g_4(OS)} \tag{18}$$

In order to solve the task allocation problem under the enforced constraints by the MSDE-SPEA2-based algorithm, the dominance relations among solutions obey the following rule:

$$OS_i \text{ dominates } OS_j,$$
$$if \begin{cases} OS_i \text{ is a feasible solution and } OS_j \text{ is not;} \\ OS_i \text{ and } OS_j \text{ are not feasible solutions} \\ \quad \text{and } CV(OS_i) < CV(OS_j); \\ OS_i \text{ and } OS_j \text{ are feasible solutions} \\ \quad \text{and } OS_i \text{ Pareto dominates } OS_j. \end{cases} \tag{19}$$

It is worth noting that a Pareto solution set

$$OSS = \{OS_1, OS_2, \ldots, OS_W\}$$

can be obtained by MSDE-SPEA2-based algorithm.

To implement the many-objective optimization of task allocation under constraints in the second layer, MSDE-SPEA2-based algorithm is shown in Algorithm 3.

Algorithm 3 MSDE-SPEA2-based Algorithm

Input: An agent set A, a task set T' and the maximum parameter W

1: Initialize population S with random solutions and empty the external nondominated set S'.
2: Copy nondominated members of S to S'.
3: Remove dominated solutions within S' by solutions also in S'.
4: **if** the number of nondominated solutions in S' beyond the maximum W **then**
5: prune S' by $k-neighbor$ of clustering with the shift-based density estimation method $MSDE$
6: **end if**
7: Calculate the fitness values on five objective functions for each individual in S and S'.
8: Select individuals from $S + S'$ to fill the mating pool through binary tournament selection.
9: Do the crossover and mutation operators on task allocation individual rules.
10: **if** reach the maximum iteration number **then**
11: stop
12: **else** go to Line 2
13: **end if**

Output: The optimal solution set OSS.

In Algorithm 3, the modified shift-based density estimation (MSDE) method (before the clustering step in the MSDE-SPEA2-based algorithm) is realized in line 5. In the MSDE method, the individual's moving distance can be decided by the weight parameter of every objective function. For example, as shown in Fig. 3, it is assumed that f_1's weight w_1 is larger than f_2's weight w_2. Then, the moving of every node on f_1-axis will follow the basic SDE method, that is, the value of every node on f_1-axis is equals to the target node's f_1 value. After that, the value of every node on f_2-axis will be the value of w_2/w_1 times the f_2 value of the target node. To be more specific, in the first axis, it is easy to see that $f_1(A) > f_1(B)$ and $f_2(A) < f_2(B)$. Thus, there are no obvious differences between the two individuals A and B on the two functions. Then, these two nodes can be evaluated by MSDE method and the process of their shift-based density estimation is shown in the second and third axis. For the second axis, the target node is A and the node B is the target node in the third axis. In order to estimate target node "$T = (i_0, j_0)$", the other nodes marked by "$P = (i, j)$" should be moved to "$P' = (i', j')$" by the following rules:

$$(i', j') = \begin{cases} (i, j_0), & \text{if } i > i_0 \ \&\ j < j_0 \ \&\ w_1 > w_2 \\ (\frac{w_2}{w_1} i_0, j), & \text{if } i < i_0 \ \&\ j > j_0 \ \&\ w_1 > w_2 \\ (i, \frac{w_1}{w_2} j_0), & \text{if } i > i_0 \ \&\ j < j_0 \ \&\ w_1 < w_2 \\ (i_0, j), & \text{if } i < i_0 \ \&\ j > j_0 \ \&\ w_1 < w_2 \end{cases} \tag{20}$$

In Fig. 3, for the middle axis, according to the target node "A", other nodes move by obeying the first two rules in Equation (20). To demonstrate the satisfactory results, the new positions are represented by hollow circles. In the right axis, according to the target node "B", other nodes move to the corresponding positions of the hollow circles by following the same rules in Equation (20). As can be seen easily, node "A" has a better clustering evaluation value than node "B". The eliminations of those nodes which have bad performance are marked by "$\times$". The MSDE method ensures the efficiency and accuracy of individual density evaluation.

When it comes to the specific task allocation, the chromosome encoding plays a significant role in the crossover and mutation operators. In order to tackle the task allocation and task scheduling issue simultaneously, a feasible solution to the encoding is shown in Fig. 4. The crossover and mutation processes are displayed in Fig. 5 and Fig. 6, respectively. In Fig. 4, every individual includes two chromosomes. One chromosome represents the coalition combinations for every task, and the other one denotes the scheduling order for the whole task set. First, in order to unify the length of the two chromosomes, the length is set as $|T|$. In the first chromosome, the range of values for each bit is $[1, 2^{|A|}]$, which can be seen as a binary representation. This binary representation includes all kinds of agent coalitions. Then, the second chromosome indicates the scheduling order of the corresponding task. Some of the values of each bit might be the same because some tasks could be executed simultaneously.

The crossover operator of chromosomes is shown in Fig. 5. According to the cut-off position, which is chosen randomly,

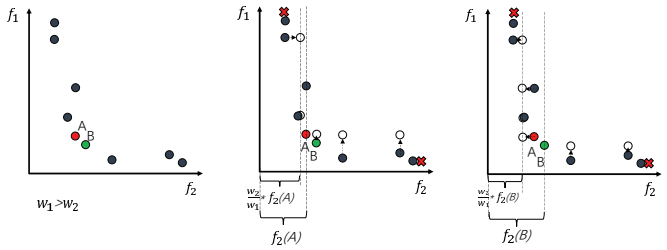


Fig. 3. An example of MSED individual moving for density estimation

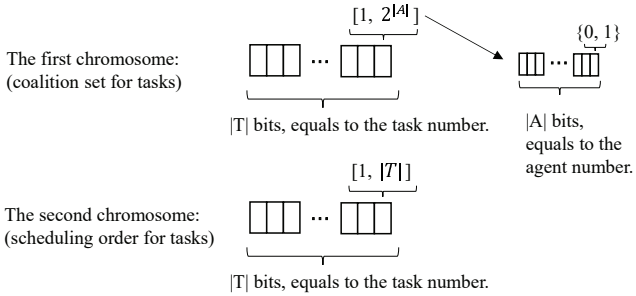


Fig. 4. The encoding details for one feasible solution

the first 7 bits of chromosomes are exchanged in Fig. 5. The crossover operator includes not only coalition information but also the corresponding scheduling sequence. Therefore, as a result, the exchanged chromosomes' scheduling information might be messy because of the missing sequence numbers. Because of the number of tasks is fixed, the scheduling sequence numbers before crossover are fixed and consecutive. After the crossover operator, the numbers in exchanged second chromosomes may not be consecutive, because some numbers are missing. For example, in Fig. 5, the scheduling sequence of individual i has the missing number "3, 5, 7", and individual j has no "1, 2".

To deal with this issue, we should supplement the missing sequence numbers. First, in the second chromosome, one bit will be chosen randomly among the bits with the same value. Next, the chosen bit replace the missing sequence number. The corresponding crossover algorithm is demonstrated in Algorithm 4. In line 4, I_{j_i} indicates the i^{th} bit of the j^{th} chromosome. The exchanging process is implemented in lines 2-6, and the replacing process is shown in lines 7-16. By the process described in Algorithm 4, in Fig. 5, three bits of individual i are chosen randomly and replaced by a random one among the missing sequence numbers of individual i. The similar operator applies to individual j.

The process of mutation is demonstrated in Fig. 6. The mutation includes coalition mutation, scheduling mutation, and coalition-scheduling pair mutation. According to the mutation probability δ, choose $\delta|T|$ bits randomly in every chromosome. For example, in Fig. 6, one bit is chosen for coalition mutation, one bit for scheduling mutation, and two coalition-scheduling pairs for couple mutation. As for coalition mutation, first, the value of the chosen bit is translated into a binary queue. Then, add "1" to any bit. Finally, convert the resultant binary queue into decimal. For the scheduling mutation, one bit is

Algorithm 4 Crossover Operator

Input: Two individual codings: I and J.

1: Randomly choose a cut-off position c_pos.
2: **for** i=0 to c_pos **do**
3: **for** j=1 to 2 **do**
4: Exchange I_{j_i} and J_{j_i}.
5: **end for**
6: **end for**
7: **for** the second chromosome in every exchanged individual encoding **do**
8: Add the missing numbers of this chromosome in set M
9: **for** i=1 to $|T|$ **do**
10: **if** the number of bits (which equal to i) > 1 **then**
11: Randomly chose one bit among the bits equal to i.
12: Replace the value of this bit by a random member in M.
13: Delete the random member in M.
14: **end if**
15: **end for**
16: **end for**

Output: Two exchanged individual codings: I' and J'.

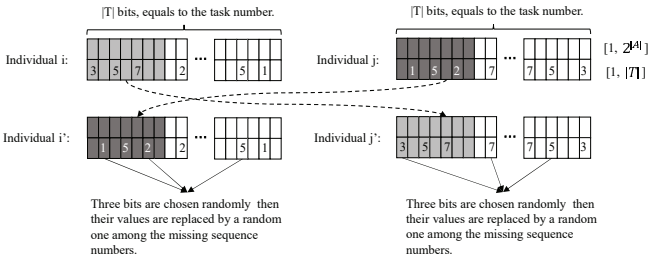


Fig. 5. The details of crossover operator

chosen randomly, and its value will replace the value of the bit that should be mutated. For coalition-scheduling pair mutation, the coalition part obeys the coalition mutation, and the scheduling part is replaced by a different scheduling sequence number. The mutation process is exhibited in Algorithm 5. The coalition mutation, scheduling mutation, and coalition-scheduling pair mutation are demonstrated in lines2-6, lines7-9 and lines10-12, respectively.

An optimal solution set OSS can be obtained by Algorithm 3, while, it is indeed that only one solution can be executed for a specific task allocation problem finally. Therefore, it is still significant to figure out a method of choosing a final solution. The specific screening steps are as follows:

1) For the first screening, the evaluation of every solution s_i in OSS can be calculated by $EF(s_1) = 0.8 * f_1(s_1) + 0.5 * f_2(s_2) + 0.75 * f_3(s_3) + 0.6 * f_4(s_4) + 0.5 * f_5(s_5)$. Then, put the solutions which have the biggest evaluation value into one set FL. If $|FL| > 0.05 * (the\ number\ of\ initialize\ population)$, the set FL needs second screening, else go to 4).
2) For the second screening, the calculation is changed into

Algorithm 5 Mutation Operator

Input: One individual coding: I.

Randomly choose $\delta|T|$ bits on every chromosome including at least one coalition-scheduling pair.
for each single bit which is chosen in the first chromosome **do**
    Change the bit's value into a binary queue.
    Randomly choose one position in this queue and do the "NOT" operator.
    Change this queue into a decimal number and replace the previous value of this bit.
end for
for each single bit which is chosen in the second chromosome **do**
    Change its value into another random bit's value.
end for
for each bit in coalition-scheduling pair which is chosen in the second chromosome **do**
    Replace its value by a random sequence number.
end for

Output: One mutated individual coding: I'.

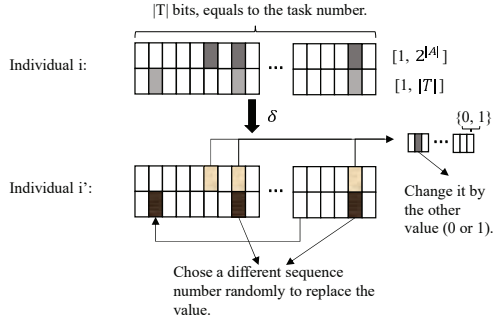


Fig. 6. The details of mutation operator

$EF(s_1) = 0.8*f_1(s_1)+0.5*f_2(s_2)+0.75*f_3(s_3)+0.6*f_4(s_4)$. Then, updated FL with the solutions which have the biggest evaluation value in the second calculation. If $|FL| > 0.05*(the\ number\ of\ initialize\ population)$, the set FL needs third screening, else go to 4).

3) For the third screening, calculate the evaluation by $EF(s_1) = 0.8 * f_1(s_1) + 0.75 * f_3(s_3) + 0.6 * f_4(s_4)$, then update FL.
4) Choose a solution in FL randomly as the final solution.

Due to our experiment data, no more than three screenings, the size of FL can be decreased.

V. Experiments and Analysis

In this section, the proposed procedure of many-objective optimization for MASs is tested to show the effectiveness of the developed algorithm by evaluating the task allocation efficiency. In the first layer, the experiment results on task dividing and prioritization simplifying are given by exploiting the deep Q-learning method. In the second layer, the performance tests are shown for many-objective optimization of task allocation from two aspects. Finally, comprehensive evaluations are given to prove the effectiveness of the whole task allocation system. The whole framework is implemented in C++ and Python, and run in the environment of the TensorFlow 2.0 CPU vision.

A. Experiments on the First-layer-agent

This subsection focuses on the results of task dividing and prioritization simplifying. The experiment is implemented via two steps. The first step is to copy the task's prioritization and probability information to the first-layer agents, and the second step is to execute Algorithm 1.

1) Experimental settings: The task prioritization is generated for the task graph instances (inspired by the Wiki-Vote database which avoids the circle in the topology graph). The values of the probabilities of positive influence are randomly produced by our data generator.

In the deep Q-learning algorithm, we set the maximum time as $t_{\max} = 1000$, decay factor $\gamma = 0.7$, learning rate $r = 0.001$, greedy explore rate $\epsilon = 0.1$, and sample number $m = 30$. In the MSDE-SPEA2-based algorithm, the weights of the five objective functions are chosen as $0.8, 0.5, 0.75, 0.6, 0.5$, respectively. The whole experiments are implemented in an offline environment.

2) Performance of deep Q-learning based on MASs: We run the experiment 30 times for different numbers of tasks in our database. Fig. 7 shows the statistical data on average reduced prioritization and the reduced percentage of divided task population. By comparing with the traditional PSO algorithm and Greedy Heuristic (GH) method, it is not difficult to find that the adopted deep Q-learning algorithm has a perfect performance in reducing the prioritization, which is independent of task size. Such an approach reduces $30\sim40\%$ of the task prioritization. As the number of tasks increases, the amount and complexity of the prioritization increase exponentially, and classic PSO and GH method cannot satisfy the requirement of searching ability and related calculations. Note that these two methods only reduce $7\sim20\%$ and $4\sim11\%$ of the task prioritization, respectively.

On the other hand, the running time plays an important role in performance evaluation. The normalized running time of the three algorithms is shown in Fig. 8. In contrast to PSO and GH algorithms, DQN has exhibited a distinct advantage in task dividing calculation, especially when the number of task increases. It can be seen in Fig. 8 that the PSO and GH both have a steep increasing tendency. Based on the Q-Network, the approximate calculation of $Q - value$ accelerates the algorithm, which becomes obvious when the task population exceeds 150.

B. Experiments on the Second-layer agent

The experiment results are shown in Fig. 9-11. Fig. 9 displays the average values of the five objective function under different task numbers with fixed 500 agents. Fig. 10 presents the results under different agent numbers with fixed 80 tasks by running the MSDE-SPEA2-based algorithm 200 times. As shown in Fig. 9, the average percentage of the makespan (f_1) under the fixed number of tasks increases as the number of task increases. The reason is that the increase in task size leads

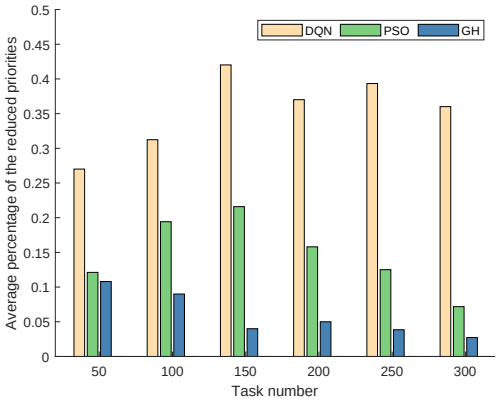


Fig. 7. The performance of deep Q-learning method on different task numbers

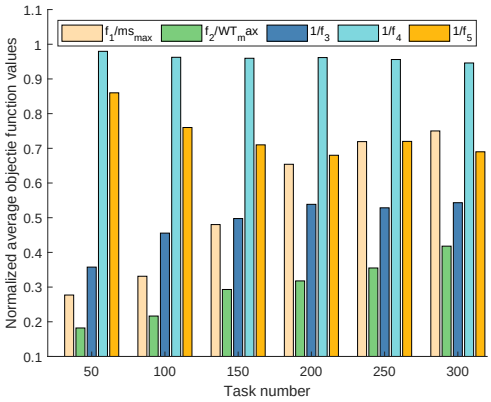


Fig. 9. The objective values through SPEA2 + MSDE with fixed agent set

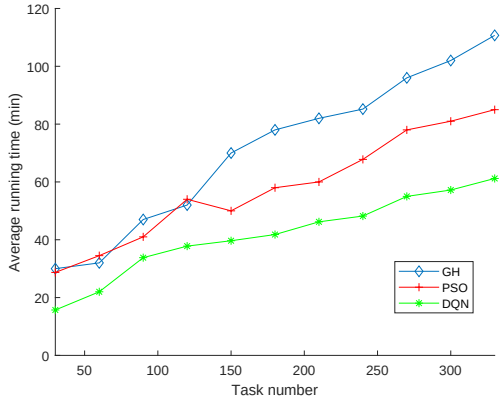


Fig. 8. The performance of deep Q-learning method on different task numbers

to a decrease in the number of tasks that can be scheduled simultaneously. In other words, in the case that the number of agents is fixed, the smaller the task size is, the less time the total scheduling would cost. It is not difficult to find that the success rate of the allocation of different task sets has a stable range (i.e., $96\sim98\%$), which shows that the developed MSDE-SPEA2-based algorithm has an advantage in improving the accuracy. In addition, the agent satisfaction and the resource utilizations stay within a suitable range, which means that our method has a merit in keeping the balance between the agent satisfaction and the resource utilizations for different task sets.

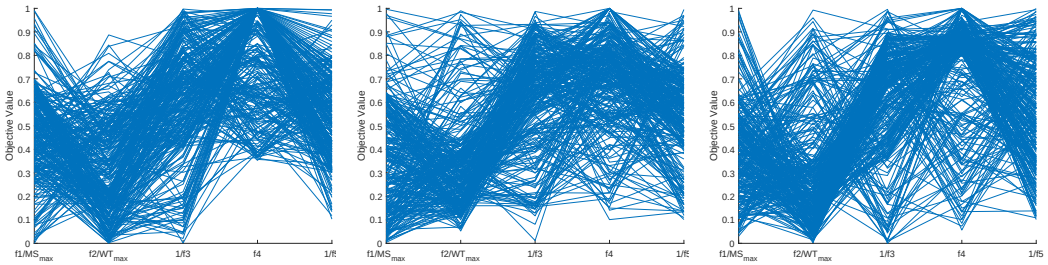
As we can see in Fig. 10, the performance of f_1 implies that the makespan has a decreasing tendency with the increase in the number of agents. The reason is that available agents cooperate with each other to finish one task for obtaining efficient results, and this reduces the execution time of a single task while increasing the number of tasks that can be executed simultaneously. Similar to Fig. 9, the success rate in Fig.10 also stays at a relatively high level. In addition, we can see that the agent satisfaction increases at the beginning episode but decreases when the agent amount breaks 200. As such, we see a perfect balance between the coalition size and the number of tasks within *a certain range* of the number of agents, which means that coalitions can produce high rewards for higher satisfaction $(f3)$ as well as higher resource utilization $(f5)$. Once the number of agents is out of this range, much more agents are assigned to one task, which results in low satisfaction and low resource utilization. On the other hand, the high diversity of the coalition (caused by the large agent size) provides diverse choices for tasks, which reduces the conflicts among coalitions effectively. Furthermore, in light of better coalitions for tasks, the optimized scheduling can be conducted, as evidenced by the decline tendency of $(f2)$ in Fig.10.

In order to further show the effectiveness of the proposed MSDE method, the performance of MSDE-SPEA2-based algorithm and SPEA2 for the task allocation with 50 tasks and 100 agents is shown in Fig. 11. By running MSDE-SPEA2-based algorithm and SPEA2 algorithm 50 times, respectively, it is obvious that MSDE-SPEA2-based algorithm has an advantage on population selection, which leads to an excellent result on the two objective functions (f_1 and f_2 here for explanation purposes).

The final results of the MSDE-SPEA2-based method are presented in Fig. 12. Three kinds of task allocation data are set for testing the convergence of the MSDE-SPEA2-based method. It is obvious that the solutions can converge into the Pareto front. In Fig. 12, a good balance between convergence and diversity is achieved by the MSDE-SPEA2-based method. It should be noticed that, some of the lines are a little messy because the solutions are discrete. The task allocation problem is a complex combination problem between agents and tasks, which means the solutions are not continuous.

In order to evaluate the performance of the MSDE-SPEA2-based method, the inverted generational distance (IGD) metric is introduced to test the method. IGD is a metric which reflects combined information about convergence and diversity of a solution set. IGD measures the average distance from the points in the Pareto front to their the closest solution in the obtained set. The results of the MSDE-SPEA2-based method are presented in Table. I on various task allocation settings. Comparing with the other four algorithms, PESA-II, SPEA-II, SPEA2 and SPEA2+SDE, the MSDE-SPEA2-based method achieves a better performance with the smallest values on both convergence and diversity.

The HV metric is introduced to calculate the volume of



(a) 100 agents VS 50 tasks (b) 200 agents VS 200 tasks (c) 250 agents VS 300 tasks

Fig. 12. The final results of the MSDE-SPEA2-based method on different agent sizes

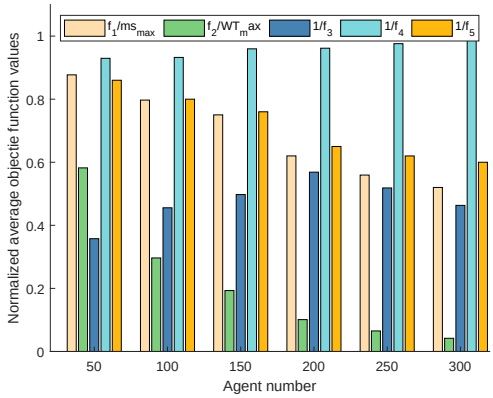


Fig. 10. The values of objective function through SPEA2 + MSDE with fixed task set

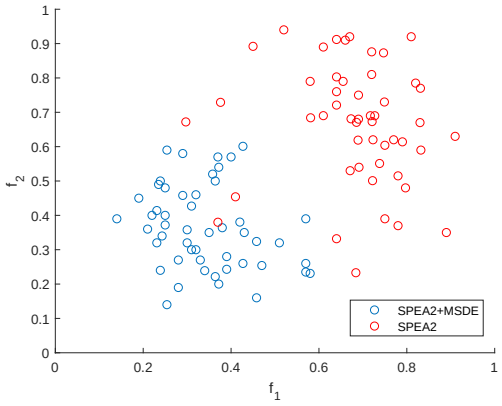


Fig. 11. The comparison between SPEA2+MSDE and SPEA2 on f_1 and f_2

the objective space between the obtained solution set and a reference point. The reference point is chosen by 1.2 times the biggest value of every objective. Table. II shows the results of HV among the five algorithms. The MSDE-SPEA2-based method achieves the best performance in different data settings with the largest HV values.

C. Comprehensive Evaluations on Layered MASs Task Allocation

The integrated task allocation on the basis of our layered model includes two parts. In the first layer, we focus on the step of task dividing and prioritization simplifying, which serves as the basis of the overall task allocation and also the preprocessing stage for the second layer. To demonstrate the significance and effectiveness of the task dividing in the first layer, the results are presented in Fig. 13. Fig. 13 includes the allocation evaluations with task dividing but without task dividing. It can be found that, with the prioritization simplifying in the first layer, the makespan saves much time and the scheduling becomes efficient.

VI. Conclusion

In this paper, the MTMRA issue has been addressed via many-objective optimization of MASs. In order to greatly improve the efficiency of the MTMRA algorithm, a novel layered MASs model has been proposed with respect to many-objective optimization. To be specific, in the first-layer MASs, the DQN-based deep Q-learning algorithm has been adopted to address the defined task dividing problem, which simplifies the complicated prioritization process of the original task set effectively. In the second-layer MASs, an MSDE approach has been applied in the step of SPEA2's population selection to handle multi-task allocation and task scheduling, simultaneously. It should be pointed out that the proposed MSDE-SPEA2-based algorithm has shown powerful capability in balancing many objectives, which can search the optimal allocation solutions in terms of five objectives including makespan, agent satisfaction, resource utilization, task completion and task waiting time under the different experiment settings. Finally, the perfect performance of the developed algorithms has been verified by extensive experiments for different parameters under different scenarios.

To summarize, the proposed DQN-based deep Q-learning and MSDE-SPEA2-based algorithms have presented excellent superiority in simplifying the complex task prioritization and deal with the many-objective optimization issue. Inspired by deep learning methods, it is worth exploring related algorithms on distributed MASs to achieve an efficient calculation method on many realistic problems included task allocation issue. Thus, further research topics include 1) the distributed MASs

TABLE I
PERFORMANCE COMPARISON (IGD) OF FIVE EAS ON DIFFERENT TASK ALLOCATION SETTINGS

Task allocation settings	PESA-II	SPEA-II	SPEA2	SPEA2+SDE	SPEA2+MSDE
50 *tasks* & 50 *tasks*	3.056E-2 (1.4E-2)	1.813E-1 (6.0E-1)	5.219E-2 (2.4E-2)	2.516E-2 (1.3E-3)	1.781E-2 (1.1E-3)
50 *tasks* & 75 *tasks*	5.794E-1 (6.4E-2)	7.293E-1 (1.0E-1)	6.201E-2 (1.98E-2)	4.612E-2 (1.9E-3)	2.772E-2 (1.4E-3)
50 *tasks* & 150 *tasks*	7.115E+0 (4.4E-2)	3.781E-1 (2.0E-3)	4.251E-1 (2.4E-3)	2.516E-2 (7.3E-4)	1.211E-2 (4.1E-4)
100 *tasks* & 100 *tasks*	4.827E-1 (4.1E-2)	2.813E+0 (5.1E-1)	3.293E-1 (3.8E-2)	7.284E-2 (6.2E-2)	5.862E-2 (3.0E-2)
200 *tasks* & 200 *tasks*	5.186E+1 (2.7E+0)	8.454E+1 (6.2E+0)	5.429E+0 (9.4E-2)	6.926E+0 (9.2E-2)	4.253E+0 (7.1E-2)
300 *tasks* & 300 *tasks*	8.235E+0 (5.4E-2)	7.327E-1 (3.4E-2)	7.824E-1 (8.2E-2)	2.917E-1 (7.1E-3)	1.362E-1 (3.6E-3)

TABLE II
PERFORMANCE COMPARISON (HV) OF FIVE EAS ON DIFFERENT TASK ALLOCATION SETTINGS

Task allocation settings	PESA-II	SPEA-II	SPEA2	SPEA2+SDE	SPEA2+MSDE
50 *tasks* & 50 *tasks*	3.382E+6 (6.4E+5)	1.923E+6 (5.0E+5)	5.829E+6 (7.3E+5)	2.073E+7 (2.1E+6)	2.517E+7 (3.2E+6)
50 *tasks* & 75 *tasks*	4.923E+6 (3.7E+5)	5.632E+6 (6.0E+5)	2.947E+6 (1.2E+5)	8.29E+6 (6.8E+5)	1.132E+7 (7.6E+5)
50 *tasks* & 150 *tasks*	4.713E+6 (1.6E+5)	3.631E+6 (7.1E+5)	4.928E+6 (3.1E+5)	7.274E+6 (8.3E+5)	7.632E+6 (9.1E+5)
100 *tasks* & 100 *tasks*	3.143E+6 (4.2E+5)	4.161E+6 (7.3E+5)	3.928E+6 (5.2E+5)	1.274E+7 (3.3E+5)	2.143E+7 (4.1E+5)
200 *tasks* & 200 *tasks*	6.612E+6 (5.1E+5)	2.782E+7 (4.12E+6)	4.018E+6 (4.7E+5)	1.274E+7 (3.2E+6)	3.362E+7 (7.1E+6)
300 *tasks* & 300 *tasks*	5.513E+6 (2.6E+5)	3.024E+6 (1.3E+5)	5.228E+6 (3.0E+5)	6.114E+6 (8.3E+5)	8.351E+6 (8.6E+5)

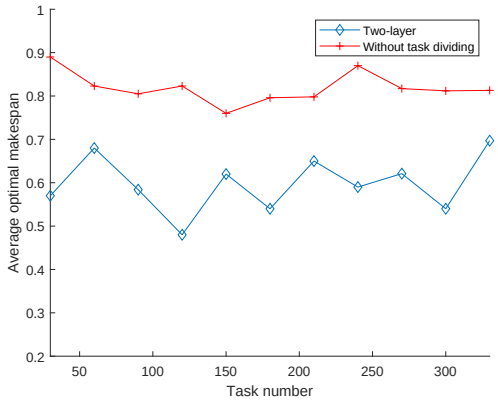


Fig. 13. The optimal makespan comparison between two kinds of methods

deep learning within different cooperation environment and various agent types for various targets [6], [25]; 2) solving the uncertain task allocation problem in a dynamic environment by using some novel optimization methods [2], [10], [35]–[38], [53]; and 3) the task allocation problem on MASs subject to engineering-oriented complexities [40], [46], [59], [62]–[64].

REFERENCES

[1] M. E. Akintunde, A. Kevorchian, A. Lomuscio and E. Pirovano, "Verification of rnn-based neural agent-environment systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6006–6013, 2019.

[2] S. Amador, S. Okamoto and R. Zivan, "Dynamic multi-agent task allocation with spatial and temporal constraints," in: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1384–1390, 2014.

[3] S. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[4] S. Bhardwaj, R. R. Ginanjar and D.-S. Kim, "Deep Q-learning based resource allocation in industrial wireless networks for URLLC," *IET Communications*, vol. 14, no. 6, pp. 1022–1027, 2020.

[5] M. J. Blondin and M. Hale, "An algorithm for multi-objective multi-agent optimization," *arXiv preprint arXiv: 2003.01745*, 2020.

[6] E. Borzello and L. D. Merkle, "Multi-agent cooperation using the ant algorithm with variable pheromone placement," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1232–1237, 2005.

[7] J. Cao, Z. Bu, Y. Wang, H. Yang, J. Jiang and H.-J. Li, "Detecting prosumer-community group in smart grids from the multiagent perspective", *IEEE Transactions on Systems Man Cybernetics: Systems*, vol. 49, no. 8, pp. 1652–1664, Aug. 2019.

[8] Z. Bu, G. Gao, H.-J. Li and J. Cao, CAMAS: "A cluster-aware multiagent system for attributed graph clustering", *Information Fusion*, vol. 37, pp. 10–21, Sept. 2017.

[9] C. Chen, K. Li, S. G. Teo, X. Zou, K. Li and Z, Zeng, "Citywide Traffic Flow Prediction Based on Multiple Gated Spatio-temporal Convolutional Neural Networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 14, pp. 1–23, 2020.

[10] H. Cheng, Z. Wang, Z. Wei, L. Ma and X. Liu, "On adaptive learning framework for deep weighted sparse autoencoder: A multiobjective evolutionary algorithm", *IEEE Transactions on Cybernetics*, in press, DOI: 10.1109/TCYB.2020.3009582.

[11] J. Contreras, M. Klusch and J. Yen, "Multi-agent coalition formation in power transmission planning: A bilateral shapley value approach," in: *Proc. International Conference on Artificial Intelligence Planning Systems*, pp. 19–26, 1998.

[12] R. Cui, J. Guo and B. Gao, "Game theory-based negotiation for multiple robots task allocation," *Robotica*, vol. 31, no. 6, pp. 923, 2013.

[13] Y. Cui, Y. Liu, W. Zhang and F. E. Alsaadi, "Sampled-based consensus for nonlinear multiagent systems with deception attacks: The decoupled method", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, in press, DOI: 10.1109/TSMC.2018.2876497.

[14] R. K. Dash, P. Vytelingum, A. Rogers, E. David and N. R. Jennings, "Market-based task allocation mechanisms for limited-capacity suppliers,"*IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 3, pp. 391–405, 2007.

[15] K. Deb, *Multi-objective optimization using evolutionary algorithms*, vol. 16, 2001.

[16] D. Ding, Z. Wang and Q.-L. Han, "Neural-network-based consensus control for multiagent systems with input constraints: The event-triggered case", *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3719–3730, 2020.

[17] F. Fellir, A. E. Attar, K. Nafil and L. Chung, "A multi-Agent based model for task scheduling in cloud-fog computing platform," in: *Proc. 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pp. 377–382, 2020.

[18] Y. Fu, H. Wang, G. Tian, Z. Li and H. Hu, "Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm," *Journal of Intelligent Manufacturing*, vol. 30, no. 5, pp. 2257–2272, 2019.

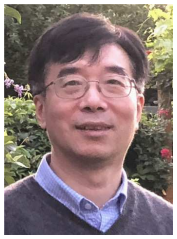
[19] C. Gao, Z. Wang, X. He and Q.-L. Han, "On consensus of second-order multiagent systems with actuator saturations: A generalized-Nyquist-criterion-based approach", *IEEE Transactions on Cybernetics*, in press, DOI: 10.1109/TCYB.2020.3025824.

[20] N. Gatti, F. Di Giunta and S. Marino, "Alternating-offers bargaining with one-sided uncertain deadlines: An efficient algorithm," *Artificial Intelligence*, vol. 172, no. 8-9, pp. 1119–1157, 2008.

[21] P. García-Sánchez, A. Tonda, A. J. Fernández-Leiva and C. Cotta, "Opti-

mizing hearthstone agents using an evolutionary algorithm," *Knowledge-Based Systems*, vol. 188, art. no. 105032, 2020.
[22] A. T. Goh, "Back-propagation neural networks for modeling complex systems," *Artificial Intelligence in Engineering*, vol. 9, no. 3, pp. 143–151, 1995.
[23] J. Han, Z. Zhang and X. Wu, "A real-world-oriented multi-task allocation approach based on multi-agent reinforcement learning in mobile crowd sensing," *Information*, vol. 11, no. 2, art. no. 101, 2020.
[24] M. Guo, B. Xin, J. Chen and Y. Wang, "Multi-agent coalition formation by an efficient genetic algorithm with heuristic initialization and repair strategy," *Swarm and Evolutionary Computation*, art. no. 100686, 2020.
[25] A. Gharbi, "A social multi-agent cooperation system based on planning and sistributed task allocation," *Information*, vol. 11, no. 1, art. no. 271, 2020.
[26] E. G. Jones, M. B. Dias and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints," *Autonomous Robots*, vol. 30, no. 1, pp. 41–56, 2011.
[27] K. Kesireddy, W. Shan and H. Xu, "Global optimal path planning for multi-agent flocking: A multi-objective optimization approach with NSGA-III," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 64–71, 2019.
[28] N. Kouka, R. Fdhila and A. M. Alimi, "Multi objective particle swarm optimization based cooperative agents with automated negotiation," in: *Proc. International Conference on Neural Information Processing*, pp. 269–278, 2017.
[29] Y. Kong, M. Zhang and D. Ye, "A negotiation-based method for task allocation with time constraints in open grid environments," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 3, pp. 735–761, 2015.
[30] I. Kovalenko, D. Ryashentseva, B. Vogel-Heuser, D. Tilbury and K. Barton, "Dynamic resource task negotiation to enable product agent exploration in multi-agent manufacturing systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2854–2861, 2019.
[31] B. Li, Z. Wang, Q.-L. Han and H. Liu, "Distributed quasiconsensus control for stochastic multiagent systems under Round-Robin protocol and uniform quantization", *IEEE Transactions on Cybernetics*, in press, DOI: 10.1109/TCYB.2020.3026001.
[32] T. Li, F. Ma and W. Liu, "Multi-agent oriented stable payoff with cooperative game," in: *Proc. International Conference in Swarm Intelligence*, pp. 74–81, 2013.
[33] Z. Li and C. Guo, "Multi-Agent Deep Reinforcement Learning Based Spectrum Allocation for D2D Underlay Communications," *IEEE Transactions on Vehicular Technology* vol. 69, no. 2, pp. 1828–1840, 2020.
[34] C. Liu, K. Li, Z. Tang and K. Li, "Bargaining game-based scheduling for performance guarantees in cloud computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 3, no. 1, pp.1–25, 2018.
[35] W. Liu, Z. Wang, Y. Yuan, N. Zeng, K. Hone and X. Liu, "A novel sigmoid-function-based adaptive weighted particle swarm optimizer", *IEEE Transactions on Cybernetics*, in press, DOI: 10.1109/TCYB.2019.2925015.
[36] W. Liu, Z. Wang, N. Zeng, Y. Yuan, F. E. Alsaadi and X. Liu, A novel randomised particle swarm optimizer, *International Journal of Machine Learning and Cybernetics*, in press, DOI: 10.1007/s13042-020-01186-4.
[37] Y. Liu, Q. Cheng, Y. Gan, Y. Wang, Z. Li and J. Zhao, "Multi-objective optimization of energy consumption in crude oil pipeline transportation system operation based on exergy loss analysis", *Neurocomputing*, vol. 332, pp. 100–110, Mar. 2019.
[38] Y. Liu, S. Chen, B. Guan and P. Xu, "Layout optimization of large-scale oil-gas gathering system based on combined optimization strategy", *Neurocomputing*, vol. 332, pp. 159–183, Mar. 2019.
[39] Y. Liu, N. Zhu and M. Li, "Solving many-objective optimization problems by a Pareto-based evolutionary algorithm with preprocessing and a penalty mechanism," *IEEE Transactions on Cybernetics*, DOI: 10.1109/TCYB.2020.2988896,2020.
[40] L. Ma, Z. Wang, J. Hu and Q.-L. Han, "Probability-guaranteed envelope-constrained filtering for nonlinear systems subject to measurement outliers", *IEEE Transactions on Automatic Control*, in press, DOI: 10.1109/TAC.2020.3016767.
[41] H. Mao, Z. Zhang, Z. Xiao, Z. Gong and Y. Ni, "Learning multi-agent communication with double attentional deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 34, art. no. 32, 2020.
[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv: 1312.5602*, 2013.
[43] A. Nath, A. Arun and R. Niyogi, "A distributed approach for road clearance with multi-robot in urban search and rescue environment," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 392–406, 2019. vol. 11, art. no. 4, 2020.
[44] H. Van Nguyen, H. Rezatofighi, B.-N. Vo and D. C. Ranasinghe, "Multi-objective multi-agent planning for jointly discovering and tracking mobile object," *arXiv preprint arXiv: 1911.09807*, 2019.
[45] M. Otte, M. J. Kuhlman and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Autonomous Robots*, vol. 44, no. 3, pp. 547–584, 2020.
[46] W. Qian, Y. Li, Y. Chen, and W. Liu, "L_2-L_∞ filtering for stochastic delayed systems with randomly occurring nonlinearities and sensor saturation", *International Journal of Systems Science*, vol. 51, no. 13, pp. 2360–2377, 2020.
[47] G. Qu, D. Brown and N. Li, "Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions," *Automatica*, vol. 105, pp. 206–215, 2019.
[48] F. Qureshi and D. Terzopoulos, "Distributed coalition formation in visual sensor networks: A virtual vision approach," in: *Proc. International Conference on Distributed Computing in Sensor Systems*, pp. 1–20, 2007.
[49] H. Ravichandar, K. Shaw and S. Chernova, "STRATA: unified framework for task assignments in large teams of heterogeneous agents," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 2, pp. 38, 2020.
[50] M. Roshanzamir, M. A. Balafar and S. N. Razavi, "A new hierarchical multi group particle swarm optimization with different task allocations inspired by holonic multi agent systems," *Expert Systems with Applications*, vol. 149, art. no. 113292, 2020.
[51] R. Rădulescu, P. Mannion, D. M. Roijers and A. Nowé, "Multi-objective multi-agent decision making: A utility-based analysis and survey," *Autonomous Agents and Multi-Agent Systems*, vol. 34, art. no. 10, 2020.
[52] T. Sandholm, K. Larson, M. Andersson, O. Shehory and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, no. 1, pp. 209–238, 1999.
[53] M. E. H. Souidi, A. Siam and Z. Pei, "Multi-agent pursuit coalition formation based on a limited overlapping of the dynamic groups," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 6, pp. 5617–1629, 2019.
[54] E. del Val, M. Rebollo and V. Botti, "Self-Organization in service discovery in presence of noncooperative agents," *Neurocomputing*, vol. 176, pp. 81–90, 2016.
[55] L. Wang, Z. Wang, G. Wei and F. E. Alsaadi, "Observer-based consensus control for discrete-time multi-agent systems with coding-decoding communication protocol", *IEEE Transactions on Cybernetics*, vol. 49, no. 12, pp. 4335–4345, 2019.
[56] K. Young and J. Wang, "Robot motion similarity analysis using an FNN learning mechanism," *Fuzzy sets and systems*, vol. 124, no. 2, pp. 155–170, 2001.
[57] Y. Yuan, Z. Wang, P. Zhang and H. Dong, "Nonfragile near-optimal control of stochastic time-varying multi-agent systems with control- and state-dependent noises", *IEEE Transactions on Cybernetics*, vol. 49, no. 7, pp. 2605–2617, 2019.
[58] C. Zhang, Q. Li, Y. Zhu and J. Zhang, "Dynamics of task allocation based on game theory in multi-agent systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 6, pp. 1068–1072, 2018.
[59] Z. Zhao, Z. Wang, L. Zou and J. Guo, "Set-Membership filtering for time-varying complex networks with uniform quantisations over randomly delayed redundant channels", *International Journal of Systems Science*, in press, DOI: 10.1080/00207721.2020.1814898.
[60] J. Zhu, Y. Song, D. Jiang and H. Song, "A new deep-q-learning-based transmission scheduling mechanism for the cognitive internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2375–2385, 2017.
[61] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
[62] L. Zou, Z. Wang, Q.-L. Han and D. H. Zhou, "Moving horizon estimation of networked nonlinear systems with random access protocol", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, in press, DOI: 10.1109/TSMC.2019.2918002.
[63] L. Zou, Z. Wang, J. Hu and D. H. Zhou, "Moving horizon estimation with unknown inputs under dynamic quantization effects", *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5368–5375, 2020.
[64] L. Zou, Z. Wang, H. Geng and X. Liu, "Set-membership filtering subject to impulsive measurement outliers: A recursive algorithm", *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 377–388, 2021.

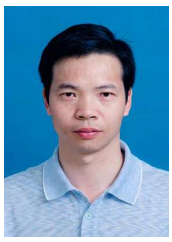
Mincan Li received the bachelor's degree in the Department of Information Science and Engineering, Hunan University, in 2014. She is currently a Ph.D. Candidate in the Department of Information Science and Engineering, Hunan University, Changsha, China. Since 2019, she has been a visiting Ph.D. student with the Department of Computer Science, Brunel University London, Uxbridge, U.K. Her research interests include multi-agent systems, many-objective optimization and machine learning.



Zidong Wang (SM'03-F'14) was born in Jiangsu, China, in 1966. He received the B.Sc. degree in mathematics in 1986 from Suzhou University, Suzhou, China, and the M.Sc. degree in applied mathematics in 1990 and the Ph.D. degree in electrical engineering in 1994, both from Nanjing University of Science and Technology, Nanjing, China.

He is currently Professor of Dynamical Systems and Computing in the Department of Computer Science, Brunel University London, U.K. From 1990 to 2002, he held teaching and research appointments in universities in China, Germany and the UK. Prof. Wang's research interests include dynamical systems, signal processing, bioinformatics, control theory and applications. He has published more than 600 papers in international journals. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, William Mong Visiting Research Fellowship of Hong Kong.

Prof. Wang serves (or has served) as the Editor-in-Chief for *International Journal of Systems Science*, the Editor-in-Chief for *Neurocomputing*, and an Associate Editor for 12 international journals including IEEE Transactions on Automatic Control, IEEE Transactions on Control Systems Technology, IEEE Transactions on Neural Networks, IEEE Transactions on Signal Processing, and IEEE Transactions on Systems, Man, and Cybernetics-Part C. He is a Member of the Academia Europaea, a Fellow of the IEEE, a Fellow of the Royal Statistical Society and a member of program committee for many international conferences.



Kenli Li Kenli Li received the Ph.D. degree in Computer Science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar at the University of Illinois at Urbana- Champaign from 2004 to 2005. He is currently a Cheung Kong Professor of Computer Science and Technology at Hunan University, the Dean of the College of Information Science and Engineering of Hunan University, and the Director in the National Supercomputing Center in Changsha. His major research interests include parallel and distributed processing, high-performance computing, and big data management. He has published more than 320 research papers in international conferences and journals such as IEEE-TC, IEEE-TPDS, IEEE-TCC, AAAI, DAC, ICPP, etc. He is an Distinguished Member of the CCF and a Senior Member of the IEEE. He is currently serving or has served as an Associate Editor for IEEE-TC, IEEE-TII, and IEEE-TSUSC.

Xiangke Liao received the B.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1985, and the M.S. degree from the National University of Defense Technology, Changsha, China, in 1988. He is currently a Full Professor and the Dean of the College of Computer Science, National University of Defense Technology. His research interests include parallel and distributed computing, high-performance computer systems, operating systems, cloud computing, and networked embedded systems.

Kate Hone received the B.A. degree in experimental psychology from the University of Oxford, Oxford, U.K., in 1990, and the M.Sc. degree in work design and ergonomics and the Ph.D. degree from the University of Birmingham, Birmingham, U.K., in 1992 and 1996, respectively.

Professor Hone is Head of the Department of Computer Science at Brunel University London, U.K. She previously held academic posts at the University of Nottingham, U.K. before joining Brunel in 2000. At Brunel she has held a number of posts including Director of the Graduate School between 2009 and 2018.

Professor Hone's research interests include spoken dialogue systems, affective computing, social signals processing, health informatics and intelligent data analysis.

Xiaohui Liu received the B.Eng. degree in computing from Hohai University, Nanjing, China, in 1982 and the Ph.D. degree in computer science from Heriot-Watt University, Edinburg, U.K., in 1988.

He is a Professor of Computing with Brunel University London, Uxbridge, U.K., where he directs the Centre for Intelligent Data Analysis. He has over 100 journal publications in computational intelligence and data science. Prof. Liu was a recipient of the Highly Cited Researchers Award by Thomson Reuters.