Applied Soft Computing Journal 96 (2020) 106603

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Multi-task allocation with an optimized quantum particle swarm method

Mincan Li [a,b], Chubo Liu [a,b], Kenli Li [a,b,*], Xiangke Liao [c], Keqin Li [a,d]

[a] College of Computer Science and Electronic Engineering, Hunan University, Hunan, China
[b] College of Computer Science and Electronic Engineering, National Supercomputing Center in Changsha, Hunan University, China
[c] College of Computer Science National University of Defense Technology, Changsha, China
[d] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 13 February 2020
Received in revised form 12 June 2020
Accepted 28 July 2020
Available online 7 August 2020

Keywords:
Coalition scheduling
EQPSO
MAS
Multi-task allocation
SQPSO

ABSTRACT

Multi-task allocation in multi-agent systems aims to accomplish tasks efficiently and successfully, while obtaining more rewards to enhance the entire system operation at the same time. Most existing assignment methods are based on agent coalitions, which cannot balance the profit distribution and task execution success rate or ignore the coalition stability, leading to a low execution level and assignment failures. Few coalition scheduling methods exist for multi-task allocation based on a fixed agent population. In this paper, we propose an effective stability quantum particle swarm optimization (SQPSO) algorithm which includes high rewards obtaining, benefit dividing, coalition stability insuring, and a historical task mechanism for search acceleration. Secondly, we design an efficient establishment quantum particle swarm optimization (EQPSO) algorithm for coalition scheduling, which is equipped with coalition similarity judgment to reduce the coalition formation time cost. The experiment results show that SQPSO guarantees a superior coalition for every task and earlier convergence in the whole task set allocation, and EQPSO gives the optimal scheduling order which reduces the total execution time.

1. Introduction

Task allocation is a significant component in multi-agent systems for the achievement of comprehensive and complex goals. When a complex task arrives, it will be allocated to a coalition directly such that agents cooperate to finish it or be divided into subtasks and then allocate to agents to accomplish. The process of finding the essential solution is known as task allocation. Therefore, the challenges of task allocation include three factors. First, for the way that allocates one task to an agent coalition, the essential step is to find the best coalition so that agents can cooperate to finish a task. Second, for the approach that divides a task into subtasks then allocates them to agents, it requires a number of agents that coordinate to finish the tasks. Third, to measure the quality of allocations, standards are complex and various, which can be execution time, task achievement, success rate of allocation and total utility, or a combination of above. Based on these three kinds of challenges, three branches of solving method emerge.

Different task allocation methods are proposed to adapt various of constraints and circumstances to get lower execution time, higher task achievement and more total utilities. As for the first challenge, Dutta et al. [1] explored coalition formation for heterogeneous agent, which made different type agents cooperate for finishing instantaneous allocations. Besides, Jiang et al. [2] based on game-theory proposed autonomous decision-making method for agent cooperation to address task allocation in a large-scale population. While for the approach dealing with second challege, Abdallah et al. [3] accomplished the task decomposing and found agents suitable for subtasks. Based on this idea Rahman et al. [4] proposed optimum subtask allocation to achieve a two-level feed-forward optimization for better performance. What is more, in the subtask allocation method, with resource or time constraints there is no doubt that coordination is an effective concept for agents to solve conflicts of resource use or time constraint problems in a cooperative multi-agent systems (MAS) environment. In the coordination mechanism based on protocols, such as contextual resource negotiation-based (CRN) [5], contract net protocol (CNP) [6,7] or other negotiation methods, agents are supplied with communication skills to aid coordination when limitations or conflicts occur. For the third challenge, one significant point of the influence of measuring allocation is the cost. It is obvious that the main cost is the communication cost during these coordinations. Thus, the pattern of allocation is extremely important

[*] Corresponding author at: College of Computer Science and Electronic Engineering, Hunan University, Hunan, China.
E-mail addresses: limc@hnu.edu.cn (M. Li), liuchubo@hnu.edu.cn (C. Liu), lkl@hnu.edu.cn (K. Li), xkliao@nudt.edu.cn (X. Liao), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.asoc.2020.106603
1568-4946/© 2020 Published by Elsevier B.V.

for cutting down time from the aspect of interaction frequency among agents. Based on this perspective, Faezeh [8] proposed an efficient task assignment approach that not only improved the allocation success rate via resource reliability calculation but also saved the execution time by accurate task matching. Hence, the negotiation protocols and advanced reliable methods mentioned above can contribute to decreasing the execution time to a certain extent. However, unlike above non-coalition measures, the mechanism for coalitions usually attaches to coalition advantages in cooperating to finish the task at a higher efficiency level.

Using coalitions to replace discrete agents has an obvious advantage, which is reducing the time needed for information passing, such that maximum utility increases in task allocation. Compared with the method in which several agents execute subtasks, assuming that plenty of agents are available for tasks, a cooperative coalition that solve one task might save greater time and cost because of advantages on resource sharing, conflict solution and cooperation optimization. Therefore, for the allocation based on coalitions, the most significant portion of task achievement is formulating a suitable coalition to meet the corresponding task requirements. Coalition formation has been shown to be an NP-hard problem [9] and the correlative algorithms aim to find the optimal coalition structure to satisfy one or several objectives in many research studies. And to improve the search efficiency, Rahwan et al. [10] decreased the search space and applied a cycle technology that can generate a valid coalition only. Besides, Yu et al. [11] considered resource constraints and proposed a new heuristic to give an optimal coalition combination for single-task robots, multi-robot tasks and the instantaneous assignment (ST-MR-IA) problem. Both of these two studies paid attention to a problem, how to find an optimal coalition in the search space. And they also gave the corresponding better solutions. Furthermore, with the increasing complexity of real applications, MAS turns to multi-task for multi-agent. With respect to the resource, Doucette [12] proposed a distributed algorithm for newly arriving tasks that included two types of agents-task agents and resource proxy agents. In his algorithm, the target was focused on leverage preemption: if there are resources more suitable for newly arriving tasks, the preemption occurs. However, when agents need cooperation to implement the tasks, it is a challenge for effective scheduling to search in exponential agent group combinations. Faced the exponential search space of agent combinations, an intelligent optimization algorithm is also a feasible strategy. Particle swarm optimization (PSO) has a big advantage, i.e., it is easy to modify PSO to weighted PSO, hybrid PSO, when it performs the shortcomings of premature convergence or traps in local optimality. And for the problem of multi-agent system task allocation, using PSO is easy for agent coalition encoding. Brief coalition expression makes optimal solution finding easier and more efficient. PSO method not only performs better results in agent control, solution search and social simulation, but also is good at the assignment, such as customer order assignment, etc. Win-Chin [13] reduced the order completion time of one agent and limited all agents' total completion time using the weighted PSO, which achieved matching of the suitable order and right agent. According to the classification PSO algorithm, this method has the limitation of falling into local optima. However, the quantum particle swarm optimization (QPSO) [14] produces a better performance than the PSO with respect to getting stuck in a local optimal solution, and its two-situation calculation makes the entire process more efficient. Because the quantum evolution algorithm (QEA) displays information by quantum state vectors and expresses chromosome coding by the probability distribution of qubits, a chromosome can be described as a superposition of multiple quantum states. These features make the quantum evolution algorithm having more parallelism than traditional evolutionary computing, which leads to high efficiency, and is applied in many works, such as [15–17]. Then, as a novel algorithm, QPSO is combined with improved QEA theory. In the QPSO algorithm, qubits are applied on particle current position encoding, quantum revolving gates are used to searching the optimal location of particles, and quantum NOT gates help the mutation of particle locations to avoid premature convergence. Therefore, the optimal solution searching capacity and optimal efficiency of QPSO are better than basic PSO. And QPSO has been used for many fields. For instance, Alokananda et al. [18] proposed a novel method based on quantum inspired which identified the optimal number of clusters automatically from an image data set.

Our objective is to accomplish the allocation and scheduling of multi-task to multi-agent through a novel QPSO method, which is aimed at finding the optimal coalition for each task based on the aspects of execution time, agent satisfaction and total profit. Compared with traditional allocation, we comprehensively consider the satisfaction of agents, which is related to the coalition stability. Only when agent cooperation could deliver more benefits in the corresponding group, i.e., agents prefer to stay in this coalition rather than maintaining other coalitions or becoming independent because their satisfaction is at a superior level, can we believe that this coalition is relatively stable and has a low possibility to occur execution failures caused by alliance collapse. Additionally, we also consider the task similarity in the process of coalition searching for the entire task set to accelerate the total running time. On the one hand, an accurate and efficient coalition set can make more utilities and save more time, on the other hand, the global scheduling is also a significant contributor to improve execution time for tasks and coalitions. The establishment and disbanding of one coalition occupy time, and several disjoint coalitions could execute the corresponding task simultaneously. Thus, formulating an appropriate coalition scheduling for tasks is expected to speed up the achievement. By combining the above two important aspects of multi-task allocation in MAS, our contributions are described as follows. Firstly, we design a novel stability quantum particle swarm optimization (SQPSO) method to find the best coalition for each task in the task set, which simultaneously includes agent satisfaction, task similarity and completion reward. Secondly, we finish the coalition scheduling for the tasks in the establishment of quantum particle swarm optimization (EQPSO) algorithm to improve the speed of the entire execution time. Section 2 presents the related work on MAS task allocation measures and applications. Then, Section 3 describes our allocation problem and presents the details of the two main algorithms, SQPSO and EQPSO. And in Section 4 we describe the experiments on a whole allocation process and give the related analysis. Finally, Section 5 presents the conclusions and notes the directions of our future work.

2. Related work

Task allocation based on no coalition, such as control mechanisms, usually combines two types of approaches, centralized and distributed, to make a system more flexible, which presents a highly self-organized distributed method and a comprehensive quality of centralized method [5]. The exact control mechanism combination includes several significant aspects of task allocation for improving the execution time in the total MAS, namely, resource access, reliability and constraint. First, the resource access method might cause resource accumulation disparities, which means that agents in certain areas could become quite "rich" in the process of "borrowing" other agents' resources through links to finish tasks, and subsequently they have more opportunities to be assigned tasks. Thus, in this circumstance, the "richer" the agents are, the more chances they will get, which makes task allocation become stuck with the partial agents. This is the largest disadvantage in that tasks are not assigned to other agents that might be more appropriate to them. The studies of Quentin et al. [19] based on two types of negotiations could decrease the assignment prejudice through a "reducer" agent, which helps to avoid unfair allocation within the data resource. Lucija et al. [20] proposed a method called STAPSO to solve the task assignment in the field of software development which shown promising performances on scrum sprint planning. Secondly, the resource supplied for tasks might not be reliable, and certain cheating behaviors might occur, i.e, agents accept tasks but do not have sufficient resources to accomplish them. To avoid this phenomenon, Jiang [21] used reconnaissance in their mechanism to test such agents. Guessoum [22] used agent replacement rules to ensure "cheater" agents could be replaced by positive agents when monitors observed them. Although these methods found better solutions for tasks, certain problems still exist for different constraints, such as time, resource or agent conflicts. Thus, the third point is offering accurate measures for the constraints. Kong [23] supplied time price calculation equations for agents and signed a contract to accomplish the allocation in the open grid circumstance according to the price computing results. Xing [24] proposed an approach for information collection to decrease the contacting links, and avoid the communication constraint and subsequently assigned the tasks for agents with sufficient and suitable space capability to avoid the space constraint. These three branches of task allocation without coalitions could find better solutions, but they also suffer from disadvantages in communication or increased system cost.

According to the agents' capabilities and task requests, the coalition formation algorithm is a feasible method for reducing communication cost and improving allocation effectiveness, especially when it is equipped with the appropriate cooperation mechanism and scheduling designs. Methods based on coalitions have been widely studied for optimal groups to achieve better cooperation on tasks. Yin [25] proposed the "quit-to-choose" mechanism for agents to obtain an undetermined state and maintained this state until the most suitable task arrived, at which point the agents join in the corresponding coalition. For dynamic tasks, Jin [26] suggested a new coordination control for agent group cooperation on tracking tasks, which was proved extraordinarily effective and was extended to uncertain allocation. Morisawa [27] presented the comparative advantage theory that allowed agents to choose necessary partners to maximize the economic benefits in the uncertain task environment. It is obvious that the coalition facilitates much greater progress on task cooperation and this technology has been applied to many realistic scenes. Jiang [28] designed a two-layer decision model for the agent group formation cycle to address the cyber–physical production systems (CPPS) targets and constraints. Besides, Ramchurn [29] applied agent autonomy to unmanned aerial vehicles (UAV) cooperation to finish tasks related to scheduling, assignment and reaction of dropping and they designed a flexible coordination model for UAV to ensure that they establish a suitable and correct coalition, similar to agents.

Optimization and improvement are significant components in MAS for task allocation based on a coalition because of the coalition advantages and wide application fields, and its formation method plays a central and crucial role in improving assignment. In fact, a number of optimizations in MAS simulation rely on intelligent optimization algorithms, including coalition optimization. For instance, for the aspect of MAS simulation, Janecek [30] combined PSO algorithm for agent social simulation, which produced the decision making process of customer consumption. As an example for agent task allocation, Younas et al. [31] examined a modified genetic algorithm (GA) to fix the assignment problem (AP) by focusing on one class problem in which every task is assigned to a group of agents to finish, which means that they used coalitions to address the large tasks problem. And it has been proved that their novel GA could can find the near-optimal solution using crossover operator building. Similarly, Kouka [32] presented a novel multi-objective particle swarm optimization (MOPSO) for MAS address the local optimum and stagnation problems. Obviously, the researchers obtained improvement among cooperative agents using this measure and the Pareto order queue to dynamically adjust agent population and subpopulation. It is worth to noting that associating MAS with the quantum property increases the correction and also accelerates the algorithm running time. Fougères et al. [33] introduced quantum cognition for agent strategy determination and applied it to the Takuzu game and presented a discussion of complex agent system modeling with quantum cognition. Furthermore, Sriarunothai [34] speeded up the agent learning process using an ion trap quantum, and Lon [35] used quantum annealing for the best coalition on D-wave. Consequently, it is necessary and significant to introduce quantum cognition into MAS for shorter running time and higher accuracy.

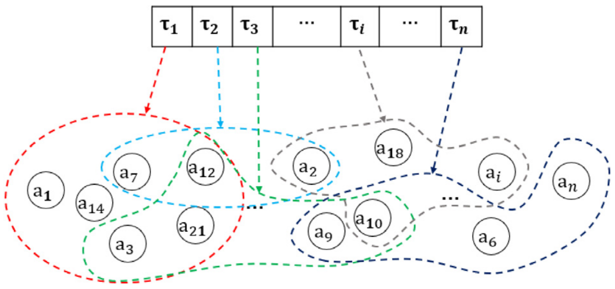


Fig. 1. The relationship of tasks, agents and coalitions.

3. The proposed algorithms

3.1. Problem description

The problem is finding the optimal coalition for each task that can meet the conditions of more profits, high agent satisfaction and rapid execution time, giving the most suitable scheduling for coalitions and the optimal scheduling plan for the entire task set.

(1) Task, agent and coalition definition.
A task, denoted as τ_i, is 2-tuple (t_d, B_{τ_i}), where t_d is the duration time of task τ_i and B_r is the set of r kinds of capability values of τ_i, $B_{\tau_i} = \{b_1, b_2, \ldots, b_r\}$.
An agent, denoted as a_i is 2-tuple (ID, B_{a_i}), where $ID = i$ is the identification number of a_i and B_{a_i} is the set of r kinds of capability values of a_i, $B_{a_i} = \{b_1, b_2, \ldots, b_r\}$.
A coalition, denotes as C_i is a set of agents that are allocated to task τ_i. The size of every coalition is different, which is decided by the allocation targets.
In order to make the relationship of tasks, agents and coalitions, we give the following illustration. In Fig. 1, we can see that different tasks have various agent combinations and unequal coalition sizes. When executing the current task τ_i, agents make the coalition C_i, and after finishing this task coalition C_i dismisses automatically.
(2) Assumptions.

(1) One agent can join only one coalition working on one task at a time.

(2) In the entire execution process of one coalition, the agent can only join one coalition from the beginning to the end or ignore it completely with no participation.
(3) The reward for finishing any task by one agent itself is less than the coalition cooperation reward, and the total environment is non-superadditive.
(4) Different coalitions can obtain varied rewards from the same task, and different tasks feedback varied profits for the same coalition.
(5) The condition that one task can be executed successfully is that at least one agent can reach all the capability value in the B_r of the task. (Therefore, according to assumptions (3) and (5) we ensure that no more than one agent cannot meet the corresponding task's requirement set.)

To solve the multi-task allocation in MAS, we take task similarity into consideration, which is defined as:

$$\rho_{\tau_i,\tau_j} = \frac{\sum_{k=1}^{r'}(v_i^k - \overline{v}_i)(v_j^k - \overline{v}_j)}{\sqrt{\sum_{k=1}^{r'}(v_i^k - \overline{v}_i)}\sqrt{\sum_{k=1}^{r'}(v_j^k - \overline{v}_j)}}. \quad (1)$$

Since we consider the optimal Shapley value [36] of the optimal solution, we calculate the similarity index based on the task rewards of each combination agent group. The similarity index between the two tasks τ_i and τ_j can be indicated by the task rewards v_i^k, which means the average reward for the kth combination of the coalition working for task τ_i and there are r' ($r' = \sum_{i=1}^{n} C_n^i$, n is the agent population) kinds of combination size total. $\overline{v}_i$ means the average reward value of all kinds of coalitions of task τ_i. The similarity application accelerates the process of assignment.

3.2. Coalition evaluation

To evaluate the coalition assigned, we consider the coalition stability, the total functions it obtains and the average coalition member satisfaction. In the cooperative game theory environment, selfish agents aim to earn more benefits, and thus agents prefer to join the task coalition that could assign them more from the total rewards. Since that one of our judgment rules is agent satisfaction calculation, we write the following equation:

$$S(a_i) = \frac{v_{\tau_j}(C_j, a_i) - v_{\tau_j}(a_i)}{v_{\tau_j}(a_i)}. \quad (2)$$

In the above equation, $S(a_i)$ denotes the satisfaction of agent a_i, $v_{\tau_j}(a_i)$ denotes the profit agent a_i obtained by finishing task τ_j itself, and $v_{\tau_j}(C_j, a_i)$ denotes the profit that agent a_i is assigned by cooperating in coalition C_j for task τ_j. Through this equation, we count every agent's satisfaction, therefore the average satisfaction index, which represents coalition stability can be calculated by:

$$S_{C_i} = \frac{1}{|C_i|}\sum_{i=1}^{|C_i|} S(a_i), \quad (3)$$

where S_{C_i} denotes coalition C_i's stability index and $|C_i|$ is the size of coalition C_i.

As shown, the coalition stability is strongly related to what every member obtains, and thus, the obtained assignment has a decisive effect for each coalition. From the perspective of justice, we divide the total profit according to the Shapley rules [36]:

$$v(C, a_i) = \sum_{C' \in C} \omega(|C'|)[v(C') - v(C' \backslash \{a_i\})], \quad (4)$$

$$\omega(|C'|) = \frac{(|C'| - 1)!(|C| - |C'|)!}{|C|!},$$

Table 1
Coalition cooperation cost value.

Coalition size	1	2	[3,4]	[5,7]	[8,10]	[11,15]
$v_{cost}(C')$	0	0.4	0.5	0.8	1.1	1.5

here, $v(C, a_i)$ denotes the value that the ith agent a_i gets from the coalition C's total gain. According to the Shapley uniqueness principle [37] the $v(C')$ is the total profits of coalition C', where $C' \in C$, and its value is calculated by $v(C') = v(\tau, C') - v_{cost}(C')$, which means finishing one task τ will bring $v(\tau, C')$ reward and the cooperating cost inside the coalition $v_{cost}(C')$ should be deducted from it. We set the communication cooperation cost in Table 1 according to the coalition size.

The other judgment rule is the single task execution time. The time calculation of coalition C working for task τ is shown in the following equation:

$$T_{(\tau,C)} = \frac{1}{r}\sum_{i=1}^{r}\left(\frac{b_{i,\tau}}{\frac{1}{|C|}\sum_{j=1}^{|C|} b_{i,j}}\right) t_d. \quad (5)$$

As shown in Section 3.1, there are r kinds of capabilities for every agent and task, thus we count every kind of capability average value among members to evaluate the total efficiency of the coalition. In this work, we present the execution time $T_{(\tau,C)}$ that coalition C work for task τ. $b_{i,\tau}$ denotes the ith capability value of task τ, and $b_{i,j}$ represents the ith capability value of the jth agent in coalition C.

3.3. SQPSO for task allocation

Inspired by the QPSO algorithm proposed by Tang et al. [38], which ensured that the calculation did not easily get stuck into the local optimization, we propose stability quantum particle swarm optimization (SQPSO) algorithm for task optimal allocation and corresponding scheduling. In the first step, we use our algorithm to find the best suitable coalition for every task in the multi-task circumstance.

By determining the two judgment rules in Section 3.2, we design the fitness function of one particle P_i, which can be transformed into the corresponding task coalition C_i:

$$f(P_i) = \alpha v(C_i)S(C_i) + \beta\frac{1}{T_{(\tau,C_i)}}, \quad (6)$$

where $v(C_i)$ denotes coalition C_i's reward, $S(C_i)$ denotes the coalition satisfaction and we can set $\alpha's$ and $\beta's$ value by the rule: $\alpha + \beta = 1$.

After giving the fitness function, the main steps of the SQPSO algorithm are detailed as follows. Compared with the regular PSO algorithm, SQPSO has several obvious differences which we can see in the following three steps. First, in the first step, SQPSO adopts the probability amplitude of qubits to the current location encoding expression and every quantum particle has two positions which correspond probability amplitude of two quantum states $|0\rangle$ and $|1\rangle$ respectively. Next, SQPSO uses quantum revolving gates to accomplish particle moving, which makes the particle's speed update by revolving gate angle changing. Finally, unlike regular PSO, in order to avoid loss of population diversity, SQPSO introduces a mutation operator through the quantum NOT gate which also avoids premature convergence.

(1) Produce the initial quantum particle swarm. One particle represents one coalition formation and it is encoded as:

$$P_i = \left[\left|\begin{array}{c|c|c|c|c|c} cos(\theta_{i1}) & cos(\theta_{i2}) & \cdots & cos(\theta_{ij}) & \cdots & cos(\theta_{in}) \\ sin(\theta_{i1}) & sin(\theta_{i2}) & \cdots & sin(\theta_{ij}) & \cdots & sin(\theta_{in}) \end{array}\right|\right]. \quad (7)$$

Here, $i = 1, 2, \ldots, m$, m is the size of the quantum particle swarm, $j = 1, 2, \ldots, n$, n represents the agent population size, $\theta_{ij} = 2\pi \times rnd$, and rnd is a random value in (0, 1). Therefore, every particle includes two states, cosine state P_{ic} and sine sites P_{is} for two types of coalition combination:

$$\begin{array}{l} P_{ic} = ((cos(\theta_{i1})), (cos(\theta_{i2})), \ldots, (cos(\theta_{in}))) \\ P_{is} = ((sin(\theta_{i1})), (sin(\theta_{i2})), \ldots, (sin(\theta_{in}))) \end{array}. \tag{8}$$

Taking P_{ic} as an example, each qubit $cos(\theta_{ij})$ represents the ith agent's probability of joining the coalition that executes the corresponding task. The value of the join probability is converted into 1 or 0 for joining the coalition or not joining the coalition, and thus the particle is converted into a coalition vector CV_{ic} as follows:

$$CV_{ic} = (cv_{i1}, cv_{i2}, \ldots, cv_{in}), \quad cv_{ij} = \begin{cases} 0, & if \quad cos(\theta_{ij}) > 0 \\ 1, & otherwise \end{cases} \tag{9}$$

The calculation methods of P_{is} and CV_{is} are the same as above.

(2) Update the quantum particle. The updating of every particle P_{ic} and P_{is} follows the increment of the qubit argument θ:

$$\begin{array}{l} \tilde{P}_{ic} = (cos(\theta_{i1}(t) + \Delta\theta_{i1}(t+1)), \ldots, cos(\theta_{in}(t) + \Delta\theta_{in}(t+1))) \\ \tilde{P}_{is} = (sin(\theta_{i1}(t) + \Delta\theta_{i1}(t+1)), \ldots, sin(\theta_{in}(t) + \Delta\theta_{in}(t+1))) \end{array}, \tag{10}$$

$$where \quad \Delta\theta_{ij}(t+1) = w\Delta\theta_{ij}(t) + c_1 r_1(\Delta\theta_l) + c_2 r_2(\Delta\theta_g)$$
$$\Delta\theta_l = \begin{cases} 2\pi + \theta_{ilj} - \theta_{ij}, & (\theta_{ilj} - \theta_{ij} < -\pi) \\ \theta_{ilj} - \theta_{ij}, & (-\pi \le \theta_{ilj} - \theta_{ij} \le \pi) \\ \theta_{ilj} - \theta_{ij} - 2\pi, & (\theta_{ilj} - \theta_{ij} > \pi) \end{cases},$$
$$\Delta\theta_g = \begin{cases} 2\pi + \theta_{gj} - \theta_{ij}, & (\theta_{gj} - \theta_{ij} < -\pi) \\ \theta_{gj} - \theta_{ij}, & (-\pi \le \theta_{gj} - \theta_{ij} \le \pi) \\ \theta_{gj} - \theta_{ij} - 2\pi, & (\theta_{gj} - \theta_{ij} > \pi) \end{cases}$$

In this manner, P_{ic} and P_{is} are transformed to $\tilde{P}_{ic}$ and $\tilde{P}_{is}$.

(3) Execute the mutation operation. Via the quantum NOT gate, the mutation can be achieved by:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} cos(\theta_{ij}) \\ sin(\theta_{ij}) \end{bmatrix} = \begin{bmatrix} sin(\theta_{ij}) \\ cos(\theta_{ij}) \end{bmatrix} = \begin{bmatrix} cos(\frac{\pi}{2} - \theta_{ij}) \\ sin(\frac{\pi}{2} - \theta_{ij}) \end{bmatrix}, \tag{11}$$

where $i \in \{1, 2, \ldots, m\}$, and $j \in \{1, 2, \ldots, n\}$. The mutation probability is p_m, and every particle has one random value rnd_i in (0, 1). If $rnd_i < p_m$, then $\lceil n/2 \rceil$ qubits are chosen randomly from particle i, and the NOT gate operation is executed on them.

According to the fitness function of coalition for one task, the SQPSO algorithm is designed as in Algorithm 1.

Let q denote the number of tasks. Before the optimal coalition finding of every task, we apply the similarity calculation in lines 3–4. If the similarity is greater than 0.6, we apply the historic optimal position to this iteration for the current task, where P_{il} denotes particle i's self-optimal position, P_{ic} and P_{is} denote particle i's two position states, and P_g represents the global optimal position.

Algorithm 1 SQPSO Algorithm

```
Input: Agent set $A$ and task set $\tau$
1:  Find the optimal coalition for each task in set $\tau(|\tau|= q)$ and use the history task set is $\tau'$ for task similarity computation by Equation (1).
2:  for $t = 1$ to $q$ do
3:      if max($\rho_{(\tau_t, \tau'_j)}$) > 0.6 then
4:          $P_g = P_{i\tau'_{max}}$
5:      end if
6:      Produce $m$ quantum state particles by Equation (7).
7:      for $g = 1$ to $g_{max}$ do
8:          for $i = 1$ to $m$ do
9:              Produce coalition vectors $CV_{ic}$ and $CV_{is}$ from $P_{ic}$ and $P_{is}$ by Equations (7) and (8), then calculate the corresponding fitness.
10:             if $fitness(CV_{ic}) > fitness(CV_{il})$ then
11:                 $P_{il} = P_{ic}$
12:             end if
13:             if $fitness(CV_{is}) > fitness(CV'_{il})$ then
14:                 $P'_{il} = P_{is}$
15:             end if
16:             if $fitness(CV_{il}) > fitness(CV_g)$ then
17:                 $P_g = P_{il}$
18:             end if
19:             Update quantum particle by Equation (10), calculate the updating vectors $\tilde{P}_{ic}$ and $\tilde{P}_{is}$.
20:             Execute the mutation operation by Equation (11).
21:         end for
22:     end for
23:     Transform $P_g$ to the coalition vector by Equation (9), then out put task $\tau_i$'s optimal coalition combination into the allocation set $C$ and add task $\tau_i$ into history task set $\tau'$.
24: end for
25: Output the allocation set $C(|C|= q)$ for task set $\tau$.
```

3.4. Coalition scheduling

In addition to finding the optimal coalition for every task, to accomplish integral allocation, we design the coalition scheduling to save addition time in the entire execution. First, because the different tasks have diverse coalition members, disjoint coalition sets can execute the corresponding task at the same time. Second, adjacent groups which contain a portion of the same agent members can save the coalition formation time by keeping the intersection members. Through Eq. (5), we can obtain the execution time computation of each task, but for precise and effective scheduling, we take the coalition establishment time into consideration for the whole multi-task execution time. The coalition establishment cost that we consider is the time of dismissing and rebuilding between two coalitions in the scheduling. We define the total coalition formation time δ_{FT} of one scheduling for q tasks as the following equation:

$$\delta_{FT} = FT_{C_1} + \sum_{i=2}^{q} \begin{cases} FT_{C_i} \frac{|C_f \backslash C'| + |C_i \backslash C'|}{|C_i|}, & if \quad \frac{|C_f \backslash C'| + |C_i \backslash C'|}{|C_i|} < 1 \\ FT_{C_i}, & otherwise \end{cases}, \tag{12}$$

where q is the number of tasks, C represents the coalition in the scheduling, $C' = C_i \cap C_f$ and C_f denotes the one coalition before C_i. Considering that two adjacent coalitions might have the same subset of agents, the former coalition C_f can save a certain amount of time by keeping the same members and only dismissing and adding the different agents. Thus, the latter coalition C_i formation time FT_{C_i} can be decreased to $(|C_f \backslash C'| + |C_i \backslash C'|)/|C_i|$ times by above equation. The value of the formation time is set as $FT_{C_i} = v_{cost(C_i)}|C_i|(|C_i| - 2)/2$, where the value of $v_{cost(C_i)}$ is set according to Table 1.

By determining the execution time computation method, we also use a similar algorithm for scheduling. Thus, the main steps details of EQSPSO are as follows. The quantum particle is set as:

$$P_i = \left[\left| \begin{matrix} cos(\theta_{i1}) \\ sin(\theta_{i1}) \end{matrix} \right| \begin{matrix} cos(\theta_{i2}) \\ sin(\theta_{i2}) \end{matrix} \left| \begin{matrix} \cdots \\ \cdots \end{matrix} \right| \begin{matrix} cos(\theta_{ij}) \\ sin(\theta_{ij}) \end{matrix} \left| \begin{matrix} \cdots \\ \cdots \end{matrix} \right| \begin{matrix} cos(\theta_{iq}) \\ sin(\theta_{iq}) \end{matrix} \right| \right], \tag{13}$$

where, every particle has q qubits that represent q tasks in the whole task set. Therefore, every qubit j denotes the jth task's start time. According to the two states of each particle P_i, we can obtain the scheduling time order vectors TS_{ic} and TS_{is} by the following equation:

$$\begin{matrix} TS_{ic} = (ts_{i1}, ts_{i2}, \dots, ts_{iq}), & ts_{ij} = \frac{1}{2}[b_i(1+\alpha_i^j) + a_i(1-\alpha_i^j)] \\ TS_{is} = (ts_{i1}, ts_{i2}, \dots, ts_{iq}), & ts_{ij} = \frac{1}{2}[b_i(1+\beta_i^j) + a_i(1-\beta_i^j)] \end{matrix}. \tag{14}$$

Using Eq. (14), we implement a solution space transformation in which a_i and b_i represent the upper and lower limitations of the time variable, respectively. According to the actual task set, the upper value is the sum execution time of every task, the lower value is zero, and α_i^j and β_i^j represent the sine value and cosine value of the jth qubit in P_i, respectively.

Therefore, our scheduling fitness function is designed as:

$$f(P_i) = Tcost(q) - \delta_{FT}. \tag{15}$$

where $Tcost(q)$ represents the total running time of q tasks, which is calculated by the corresponding scheduling order of TS. Based on the above, the establishment quantum particle swarm optimization (EQPSO) scheduling algorithm is designed as in Algorithm 2.

Algorithm 2 Scheduling Algorithm EQPSO

```
Input: Coalition set with corresponding time
1:  Produce $m$ quantum state particles by Equation (13).
2:  for $g = 1$ to $g_{max}$ do
3:      for $i = 1$ to $m$ do
4:          Produce time order vectors $TS_{ic}$ and $TS_{is}$ from $P_{ic}$ and $P_{is}$ by Equations (14), then calculate the corresponding fitness, that is the whole execution time by Equation (15).
5:          if $fitness(TS_{ic}) < fitness(TS_{il})$ then
6:              $P_{il} = P_{ic}$
7:          end if
8:          if $fitness(TS_{is}) < fitness(TS'_{il})$ then
9:              $P'_{il} = P_{is}$
10:         end if
11:         if $fitness(TS_{il}) < fitness(TS_g)$ then
12:             $P_g = P_{il}$
13:         end if
14:         Update quantum particle by Equation (10), calculate $\tilde{P}_{ic}$ and $\tilde{P}_{is}$.
15:         Execute the mutation operation by Equation (11).
16:     end for
17: end for
18: Output the allocation time order queue set $TO(|TO| = q)$.
```

4. Experiment

The objective of this section is to evaluate and analyze the performance of our SQPSO method in MAS task allocation and that of our new scheduling method EQPSO in coalition scheduling. Our testbed is based on MATLAB R2018b. In this part, we need the agent data set and the task data set. For the design of data set, we use a synthetic data generation code to produce the virtual task data and agent data which meets the problem definitions. According to the task's and agent's 2-tuple, our data generation can ensure the tuple's vector. Besides, it can set similar tasks data, different rewards to carious agents and different profits for various "coalition-task" pairs. Thus, the reason why we use a synthetic data generation code to do the data test of two algorithms is that the normal data set cannot meet the above requirements. By the way, our problem definitions are based on real allocations. Although the synthetic data generation code produce the data set for our experiments, the produce processing obeys the realistic applications, such as disaster evacuation, factory and seller matching, machine scheduling and medical resource assignment. As long as our data set crossposting to the specific model of realistic problem, our method can be applied on it. The rest of this section is organized as follows. In Section 4.1, we evaluate the performance of SQPSO, includes fitness value and convergency, in varied parameter and task settings. Secondly, we test the EQPSO algorithm for the coalition scheduling in Section 4.2, which includes evaluation and analysis under different settings of parameters such as similarity index, task size, etc.

4.1. SQPSO performance

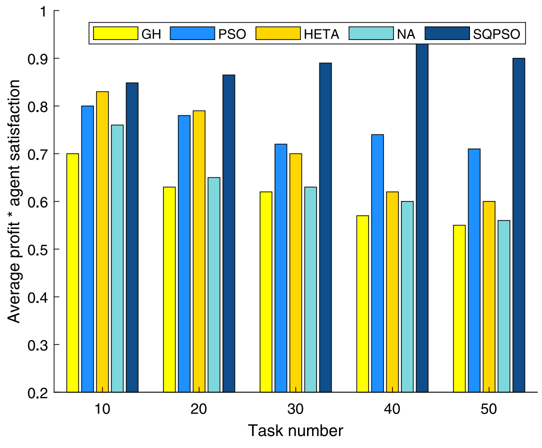
As described in Section 3.3, we design the SQPSO algorithm based on better assignment in coalition reward for the corresponding tasks, coalition stability and execution time. Thus in this subsection, we evaluate the SQPSO performance on the allocation results in Section 4.1.1 and perform algorithm analysis on the different conditions in Section 4.1.2.

4.1.1. Assignment effect

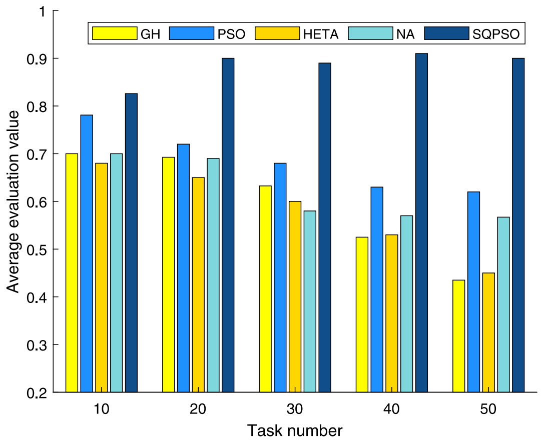
In order to evaluate allocation effect of our algorithm, three methods are shown under the same experiment circumstance, Greedy heuristic, normal PSO and High efficient task allocation. For Greedy heuristic approach, it takes agent rewards order into task allocation to obtain a higher pay back in every greedy step. And High efficient task allocation calculates agent capability enrichment factor according to each task before every assignment. Then we can set the parameters as follows: the agent population is 10, the task set size is set in [10,50] and increases by 10, and the single iteration maximum is $g_{max} = 500$. The related attribute values of the task requirements and agent capabilities are found in our data sets, which ensures that different combinations for the same task have different reward values and that the profits between tasks are varied. Therefore, every type of reward can be assigned by the Shapley rules to evaluate the corresponding coalition qualities undertaken by four methods. According to the fitness function in Eq. (6), in SQPSO we can set the values of α and β according to our targets. As a result, we set $\alpha = 1, \beta = 0$ and $\alpha = 0.5, \beta = 0.5$ separately. Therefore with the former setting, we can focus on the total profits that the coalition earns and the coalition stability, while for the latter setting we can focus on the integrated qualities, which include reward and execution time.

After running SQPSO algorithm 500 times and obtaining the average profit of the same task set, we get the normalized results for convenience of comparison in Fig. 2 under the settings ($\alpha = 1, \beta = 0$ and $\alpha = 0.5, \beta = 0.5$) respectively. To demonstrate the SQPSO advantages, we also run the other three algorithms under the same experimental conditions for comparison. After 500 times running we calculate the statistics of the average rewards and the agent satisfaction of the optimal coalitions under the Greedy heuristic method (GH), normal PSO method High efficient task allocation method (HETA) and Negotiation allocation algorithm (NA), as shown in Fig. 2.

From Fig. 2, it is clear that the SQPSO method obtains a better performance than other methods under the two kinds of weight



(a) $\alpha = 1, \beta = 0$



(b) $\alpha = 0.5, \beta = 0.5$

Fig. 2. Average assignment outcome comparison.

coefficient settings of α and β. By increasingly growing number of tasks, SQPSO always get the best results of the optimal coalition, which is more than 0.8 for every task set.

In Fig. 2(a), the other four methods do not take coalition stability into the calculations, hence, the statistics for their results are lower than those of our method, which means that the coalition stabilities cannot be guaranteed using these four methods. Especially, with the growing number of tasks, the calculation results of agent profits and capability enrichment factor have more approximate values which makes Greedy heuristic and High efficient method have a worse performance. Besides, NA method has more conflicts that cannot be avoided and lead to worse results. And the reason why SQPSO has higher values than PSO method is that SQPSO algorithm has stronger ability to avoid getting stuck in local optimum than PSO. Although we set the PSO parameters as: $V_{max} = 5$, $V_{min} = -5$ (particle speed range), $X_{max} = 1$, $X_{min} = 1$ (optimum solution range), the optimal inertia factor $\omega > 0$, and PSO gives its best results, it is not better than QPSO. While in Fig. 2(b), we can observe that SQPSO has the best performance in comprehensive solution searching. With the setting of $\alpha = \beta = 0.5$, the execution time is taken into the fitness counting. While GH and HETA method cannot integrate more objectives for better coalitions because of the complexity of themselves, they get a lower result. And NA method cannot handle the complex goals except avoiding the conflicts between agent coalition formation, so it performs unstably. Through improved calculation complexity, SQPSO gives the best result on average optimal solutions within varied task sizes. At the setting of $\alpha = \beta = 0.5$, we also give the best performance on the corresponding setting of these five algorithms in Table 2. We can observe that SQPSO has an optimum result under every condition. Greedy heuristic method and negotiation method have weaker solutions. High efficient method has unstable evaluations under different conditions. In this time, we set the adaptive weight ω' for PSO, it can be seen that it shows better than in Fig. 2(a), but QPSO also exceeds PSO a lot. To summarize, SQPSO has outstanding validity in finding the optimal coalition with both higher coalition stability and lower execution time under the condition of a growing searching space.

Excepting task number, agent population size is another factor in the assignment effect. Agent size has a strong and direct influence on the search space of one task allocation. Therefore, we test a 30-task set with different agent populations on SQPSO, which means that each task-coalition matching has $\sum_{i=2}^{|A|} C_{|A|}^{i}$ ($|A|$ is the agent size) kinds of possible combinations. Here, we choose the two parameters setting as: $\alpha = 0.5$ and $\beta = 0.5$, and

Table 2
Optimum coalition evaluation value of five algorithms.

Task number	Optimum value				
	GH	PSO	HETA	NA	SQPSO
10	0.752	0.804	0.730	0.780	0.877
20	0.710	0.734	0.685	0.727	0.933
30	0.662	0.707	0.623	0.596	0.904
40	0.551	0.654	0.551	0.588	0.926
50	0.462	0.633	0.502	0.570	0.941

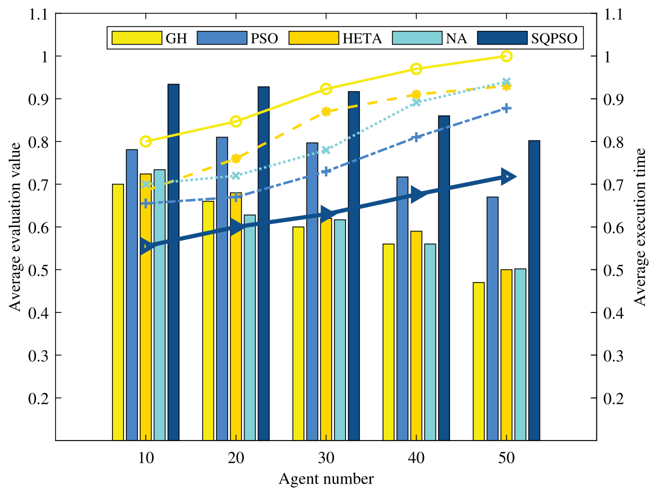


Fig. 3. Comparison on different agent populations.

it will be fixed in the subsequent experiments. By running the SQPSO, Greedy heuristic method, PSO High efficient algorithm and Negotiation allocation method 500 times, we get the average of the optimal coalition evaluation values, and the results are shown in Fig. 3.

With a growing number of agent populations, the search space increases substantially such that the calculation capability of the general greedy method, negotiation allocation method and high efficient method cannot follow the speed of growth and obtain the worse results. In the notably large combination space, the differences between superior coalitions are more delicate, and thus PSO is more easily trapped in local optima. Therefore, PSO presents a decreasing trend in the above figure. We note that SQPSO maintains a steady performance exceeding 0.8, although the line shows a small declining trend because of the growing

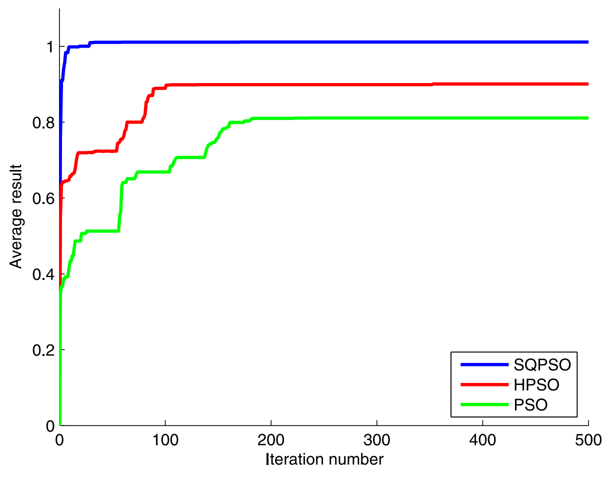


Fig. 4. Average convergence of three methods.

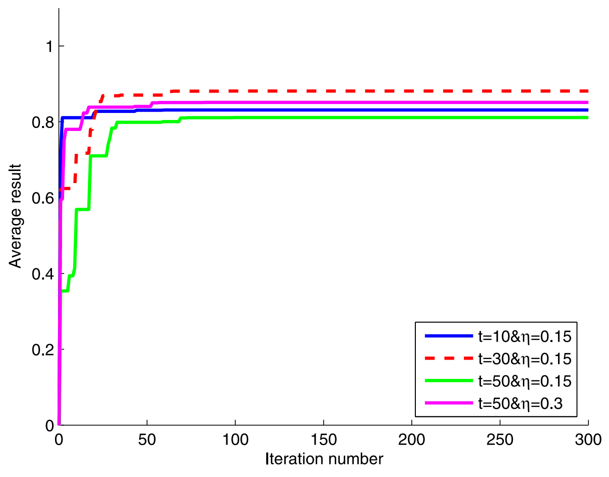


Fig. 5. SQPSO performance on different task settings.

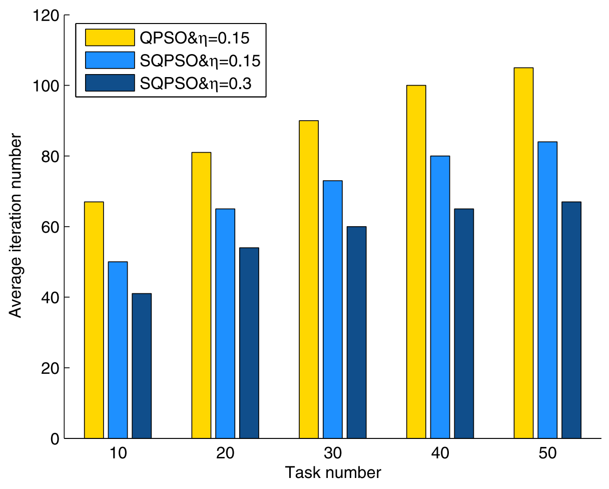


Fig. 6. SQPSO performance on different task settings.

searching space. Besides, from the five lines, we can compare the execution time of the five algorithms on different agent population sizes. It is obvious that all algorithms' running time grows up with the increasing agent population size. But it should be noticed that Greedy algorithm always performs worst, and SQPSO obtains the shortest time. The reason is that SQPSO can find the optimal solution more efficiently by the quantum revolving gate and the historical mechanism saving the time between the solutions of similar tasks.

4.1.2. Algorithm convergence

The convergence is one of the standards for estimating the SQPSO's speed of finding the optimum, thus we set the task population as 30 and with a 10-agent size for 10 runs. For each running, we record the current global best fitness value change of every task and obtain the average. We subsequently calculate the average of 10 times running. We set the iteration number as $g_{max} = 500$ and run the PSO, HPSO (PSO method with added historical task mechanism) and SQPSO for the average change process of the fitness value shown in Fig. 3.

The results show that SQPSO obtains the best coalition solution at approximately the 50th iteration at the average level. The PSO's convergence point is about 200 and that of HPSO is near 120. It is obvious that the similarity rules aid the speed aspect through comparison of PSO and HPSO. And SQPSO has a notable advantage in finding the optimal coalition for the tasks fast which can reflect by the blue line. Therefore, the first main reason for SQPSO's early convergence point is our historical task set for similar task judgment, which speeds up the entire algorithm. The second main reason is that SQPSO method can calculate two fitness values of one particle which accelerates the convergence. With these two aspects, SQPSO shows the better results. In addition, the blued line height in Fig. 4 also shows the excellent SQPSO searching ability.

In order to test the further effect of our similarity rules on the convergence, we run the SQPSO algorithm on the varied task set conditions. As shown in Fig. 5, we set the similar task percentage as 15% and 30% for different task numbers of 10, 30 and 50. (Under the searching order of the task set, the similar task percentage of 15% means that 15% of tasks can find a similar task in the history task set.)

First, when the similarity index η is 15%, the 10-task line converges near the 49th iteration, while the 30-task line and 50-task line converge at the 60th and 63th iterations respectively. The 10-task line converges earlier because the similar task number is $\eta * 10 = 1.5$, therefore no more than two tasks are similar tasks. Thus, the result is similar to the result of SQPSO in Fig. 4. As long as η is invariant, the convergence point displays little fluctuation within [50, 100] when the task set size changes. Second, when the size of task is fixed at 50, we can see that the convergence point of $\eta = 0.3$ occurs earlier than that of $\eta = 0.15$, which proves that the similar tasks rule improves the algorithm.

Besides, we compare QPSO with SQPSO in different task sizes, and Fig. 6 displays the results. From Fig. 6, SQPSO has a smaller average iteration number than QPSO in the same task size conditions, and the average iteration number decreases when η grows in SQPSO performance because of the historical mechanism.

4.2. Scheduling and execution time

In this section we test the coalition scheduling of EQPSO and compare it with other three methods. In the first section we examine the scheduling time of EQPSO and perform the comparison. And in the last part, we analyze the EQPSO under different parameter settings.

4.2.1. Scheduling performance

By accepting the consequences of SQPSO algorithm in Section 4.1.1 Fig. 2(b), we can obtain optimal coalition solutions for 10 to 50 task sets'. And based on these coalition sets, we can do the coalition scheduling by running EQPSO, and other four

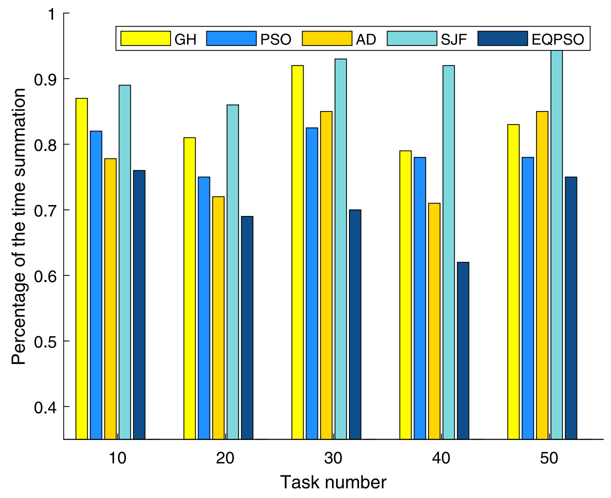


Fig. 7. Scheduling time comparison in different task sizes.

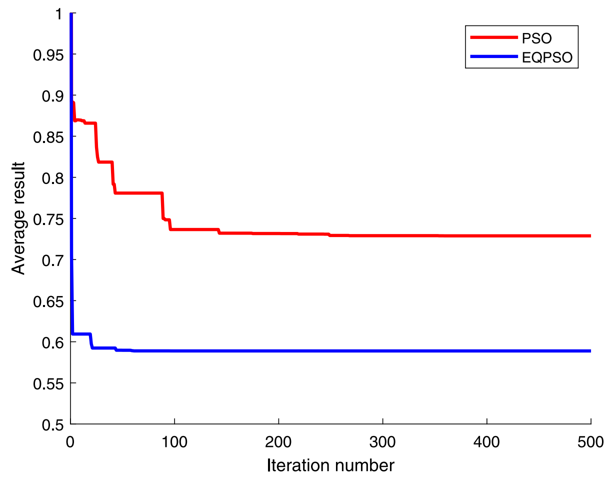


Fig. 8. Average convergence of two scheduling methods.

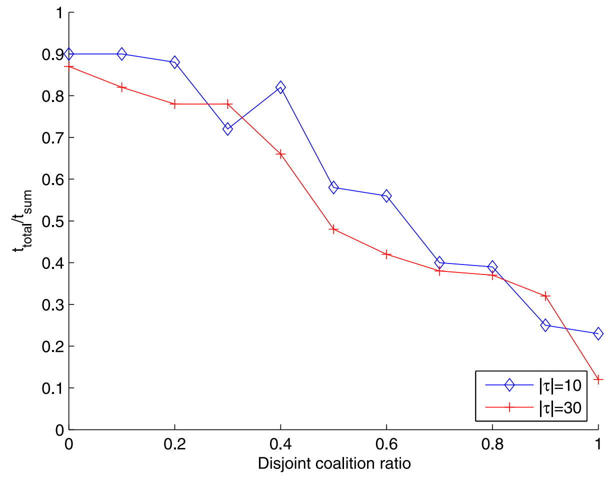


Fig. 9. Scheduling time under different disjoint ratios.

approaches, PSO, Agent dependency (AD), Greedy heuristic (GH) method and shortest Job First (SJF) algorithm, respectively. The EQPSO algorithm obeys the counting rules of execution time and the similar coalition mechanism to save time, like we described in previously parts. Whereas the PSO algorithm only obeys the normal execution time, Greedy heuristic obeys greedy rules avoiding agent conflicts, Agent dependency method calculates constraints set to decrease the time and SJF obeys the shortest sequence of tasks. By running these methods, we can obtain five optimal scheduling time queues, and then we subsequently calculate the total execution time of each task set and calculate the statistics, which are shown in Fig. 7.

At the same conditions, regardless of the task population, EQPSO obtains a shorter time than others for the entire execution, which proves that saving formation time between similar coalitions is necessary and effective. Both Agent dependency method and Greedy heuristic method have unstable performance because of the various size of agent confliction under different task populations. And SJF shows the worst performance because the simple sequence has little help in saving more time. Whereas, using similar coalition computation can save approximately 4%–18% of time, which proves our time calculation rules are necessary. For the further analysis, this percentage is decided by the similarity index and disjoint ratio. The former parameter, the similarity index, reflects the possibility and percentage of formation time that EQPSO can save between neighboring similar coalitions in the scheduling order. The latter parameter, the disjoint ratio, exposes the possibility and ratio of the coalitions that can be executed at the same time in the corresponding agent population. These two parameters will be discussed in the following subsection.

We also present the statistics of the EQPSO's convergence in the above conditions, and the results are shown in Fig. 8. Compared with the PSO algorithm, EQPSO reaches the convergence point much earlier, at approximately the 60th iteration, whereas the PSO reaches this point near 240th iterations. The EQPSO's convergence is similar to that of SQPSO, and thus we do not present the repeated illustrations.

4.2.2. Scheduling adaptability

As mentioned in Section 4.2.1, the similarity index and disjoint ratio are the most significant aspects that can influence the execution time in EQPSO. Therefore, we test how these two parameters influence the execution time in the 10-task and 30-task sets and show the results in the following figures.

It is obvious that the larger the disjoint coalition percentage, the higher the possibility of simultaneously executing the coalitions. At the same time, the total execution time is much shorter. If there are no disjoint coalitions (disjoint ratio equals to 0), it means that no coalitions can be executed at the same time in the current agent population, thus the execution time minimum is the value of saving time between the similar neighboring coalitions to the extent possible.

From Fig. 9, when the ratio is zero (which means only saving formation time), the minimum time when $|\tau| = 10$ is 90%, and the maximum when $|\tau| = 30$ is 87%. If the ratio is equal to 1, it means that all of the coalitions are disjoint, thus all of the coalitions can be executed at the same time. Therefore the total execution time equals to the longest single task execution time. In Fig. 10, the two longest time values are 23% and 12%. Except for the two special extreme values of the disjoint percentage, we can observe that the entire trend is declining, thus the high ratio helps to decrease the entire execution time significantly.

Compared with the disjoint ratio, the similarity index also has an influence on the scheduling effect. From Fig. 9, the two lines both show a "V" trend, and the analysis of this phenomenon is described as follows: when the index is equal to 0, it means that all coalitions are not similar, therefore EQPSO can only assign disjoint coalitions executed simultaneously for a better scheduling. As a result, the time can be decreased to 80.2% and 75.1%. When

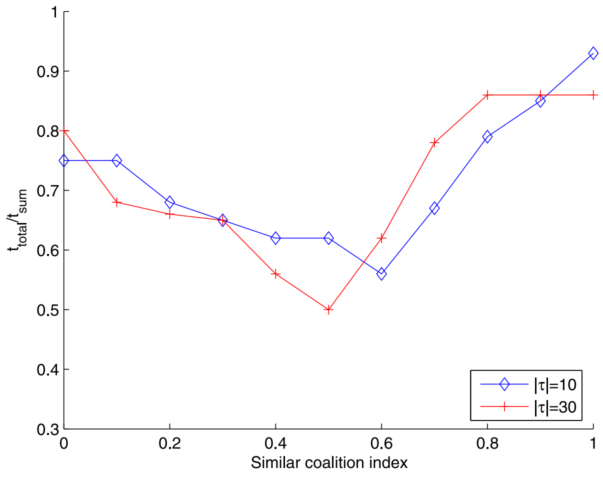


Fig. 10. Scheduling time under different similarity index.

the index is equal to 1, all coalitions are similar, thus no coalitions can be executed at one time and it leads to two high points of 93% and 86%. However, when the index values reach a suitable value with the disjoint ratio in the range of (0, 1), the time can be decreased to the smallest value in the corresponding scheduling, which is the lowest point of the "V" trend.

5. Conclusion

In this paper, we propose the SQPSO algorithm for multi-task allocation in MAS, which focuses on coalition stability, reward assignment and execution time, and then we propose the EQPSO method for coalition scheduling to decrease the entire execution time for a multi-task set, which creatively takes coalition similarity into the consideration. In the SQPSO algorithm, the first contribution is that combining the similar task judgment and quantum particle two-state feature improves the speed of complex task allocation in MAS, and the second contribution is that agent satisfaction calculation ensures the coalition stability, which increases the quality of assignment. The third contribution is that EQPSO gives a guarantee for the total scheduling because it saves time from similar coalitions in the aspect of formation costs. Besides, in the experimental section, we show the performance of the two algorithms under varied parameters and conditions, such as agent population and task set, then we give illustrations and explanations. To summarize, the SQPSO and EQPSO methods accelerate the allocation speed and are better able to avoid the local optimum dilemma, thus reaching a better global result especially in the complex and notably large search space of MAS. Additionally, the historical task mechanism and similarity rules also reinforce the optimal consequences. In the future, we will study automatic coalition formation for MAS task allocation on the Qbsolv platform. The research on the quantum-agent in MAS under complex topologies is also a significant target for future studies.

CRediT authorship contribution statement

Mincan Li: Conceptualization, Methodology, Acquisition of data, Analysis and interpretation of data, Writing - original draft, Visualization, Investigation, Coding, Validation, Writing - review & editing. **Chubo Liu:** Conceptualization, Methodology, Analysis and interpretation of data, Writing - original draft, Visualization, Investigation. **Kenli Li:** Supervision. **Xiangke Liao:** Supervision. **Keqin Li:** Visualization, Investigation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments. The research is supported by the National Key Research and Development Program of China, China (Grant no. 2016YFB0201800) and the Department of Education of Hunan Province, China (Grant no. CX20190307).

References

[1] A. Dutta, A. Asaithambi, V. Ufimtsev, et al., Distributed coalition formation with heterogeneous agents for task allocation, in: The Thirty-Second International Flairs Conference, 2019.

[2] I. Jang, H.-S. Shin, A. Tsourdos, Anonymous hedonic game for task allocation in a large-scale multiple agent system, IEEE Trans. Robot. 34 (6) (2018) 1534–1548.

[3] S. Abdallah, V. Lesser, Modeling task allocation using a decision theoretic model, in: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, 2005, pp. 719–726.

[4] S.M. Rahman, Y. Wang, Mutual trust-based subtask allocation for human–robot collaboration in flexible lightweight assembly in manufacturing, Mechatronics 54 (2018) 94–109.

[5] Y. Jiang, J. Jiang, Contextual resource negotiation-based task allocation and load balancing in complex software systems, IEEE Trans. Parallel Distrib. Syst. 20 (5) (2009) 641–653.

[6] S.J. Alam, F. Hillebrandt, M. Schillo, Sociological implications of gift exchange in multiagent systems, J. Artif. Soc. Soc. Simul. 8 (3) (2005).

[7] M. Owliya, M. Saadat, G.G. Jules, M. Goharian, R. Anane, Agent-based interaction protocols and topologies for manufacturing task allocation, IEEE Trans. Syst. Man Cybern. A 43 (1) (2013) 38–52.

[8] F. Rahimzadeh, L.M. Khanli, F. Mahan, High reliable and efficient task allocation in networked multi-agent systems, Auton. Agents Multi-Agent Syst. 29 (6) (2015) 1023–1040.

[9] S.-X. Su, S.-L. Hu, C.-Y. Shi, Coalition structure generation with worst case guarantees based on cardinality structure, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2007, p. 197.

[10] T. Rahwan, S.D. Ramchurn, N.R. Jennings, A. Giovannucci, An anytime algorithm for optimal coalition structure generation, J. Artif. Intell. Res. 34 (2009) 521–567.

[11] Y. Zhang, L.E. Parker, Considering inter-task resource constraints in task allocation, Auton. Agents Multi-Agent Syst. 26 (3) (2013) 389–419.

[12] J.A. Doucette, G. Pinhey, R. Cohen, Multiagent resource allocation for dynamic task arrivals with preemption, ACM Trans. Intell. Syst. Technol. (TIST) 8 (1) (2016) 3.

[13] W.-C. Lin, Y. Yin, S.-R. Cheng, T.E. Cheng, C.-H. Wu, C.-C. Wu, Particle swarm optimization and opposite-based particle swarm optimization for two-agent multi-facility customer order scheduling with ready times, Appl. Soft Comput. 52 (2017) 877–884.

[14] S. Yang, M. Wang, et al., A quantum particle swarm optimization, in: Proceedings of the 2004 Congress on Evolutionary Computation, Vol. 1, IEEE Cat. No. 04TH8753, IEEE, 2004, pp. 320–324.

[15] Y. Meraihi, A. Ramdane-Cherif, M. Mahseur, D. Acheli, A hybrid quantum evolutionary algorithm with cuckoo search algorithm for QoS multicast routing problem, Int. J. Commun. Netw. Distrib. Syst. 22 (3) (2019) 329–361.

[16] J. Wright, I. Jordanov, Convergence properties of quantum evolutionary algorithms on high dimension problems, Neurocomputing 326 (2019) 82–99.

[17] Y. Zhang, X. Qian, J. Wang, M. Gendeel, Fuzzy rule-based classification system using multi-population quantum evolutionary algorithm with contradictory rule reconstruction, Appl. Intell. 49 (11) (2019) 4007–4021.

[18] A. Dey, S. Dey, S. Bhattacharyya, J. Platos, V. Snasel, Novel quantum inspired approaches for automatic clustering of gray level images using Particle Swarm Optimization, Spider Monkey Optimization and Ageist Spider Monkey Optimization algorithms, Appl. Soft Comput. 88 (2020) 106040.

[19] Q. Baert, A.C. Caron, M. Morge, J.-C. Routier, Fair multi-agent task allocation for large data sets analysis, in: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, 2016, pp. 24–35.

[20] L. Brezočnik, I. Fister, V. Podgorelec, Scrum task allocation based on particle swarm optimization, in: International Conference on Bioinspired Methods and their Applications, Springer, 2018, pp. 38–49.
[21] Y. Jiang, Y. Zhou, W. Wang, Task allocation for undependable multiagent systems in social networks, IEEE Trans. Parallel Distrib. Syst. 24 (8) (2013) 1671–1681.
[22] Z. Guessoum, J.-P. Briot, N. Faci, O. Marin, Towards reliable multi-agent systems: An adaptive replication mechanism, Multiagent Grid Syst. 6 (1) (2010) 1–24.
[23] Y. Kong, M. Zhang, D. Ye, X. Luo, A negotiation method for task allocation with time constraints in open grid environments, in: Next Frontier in Agent-Based Complex Automated Negotiation, Springer, 2015, pp. 19–36.
[24] X. Su, M. Zhang, Q. Bai, Dynamic task allocation for heterogeneous agents in disaster environments under time, space and communication constraints, Comput. J. 58 (8) (2015) 1776–1791.
[25] X. Yin, B. Li, D. Li, Y. Zhang, J. Zhu, Task allocation via coalition formation in agent networks, J. Intell. Fuzzy Systems 30 (1) (2016) 197–210.
[26] L. Jin, S. Li, H.M. La, X. Zhang, B. Hu, Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach, Automatica 100 (2019) 75–81.
[27] T. Morisawa, K. Hayashi, I. Mizuuchi, Allocating multiple types of tasks to heterogeneous agents based on the theory of comparative advantage, J. Robot. 2018 (2018).
[28] Z. Jiang, Y. Jin, E. Mingcheng, Q. Li, Distributed dynamic scheduling for cyber-physical production systems based on a multi-agent system, IEEE Access 6 (2018) 1855–1869.
[29] S.D. Ramchurn, J.E. Fischer, Y. Ikuno, F. Wu, J. Flann, A. Waldock, A study of human-agent collaboration for multi-UAV task allocation in dynamic environments, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015, pp. 1184–1192.
[30] A. Janecek, T. Jordan, F.B. de Lima-Neto, Agent-based social simulation and PSO, in: International Conference in Swarm Intelligence, Springer, 2013, pp. 63–73.
[31] I. Younas, F. Kamrani, M. Bashir, J. Schubert, Efficient genetic algorithms for optimal assignment of tasks to teams of agents, Neurocomputing 314 (2018) 409–428.
[32] N. Kouka, R. Fdhila, A.M. Alimi, Multi objective particle swarm optimization based cooperative agents with automated negotiation, in: International Conference on Neural Information Processing, Springer, 2017, pp. 269–278.
[33] A.-J. Fougères, Agent having quantum properties: The superposition states and the entanglement, in: International Conference on Computational Collective Intelligence, Springer, 2017, pp. 389–398.
[34] T. Sriarunothai, S. Wölk, G.S. Giri, N. Friis, V. Dunjko, H.J. Briegel, C. Wunderlich, Speeding-up the decision making of a learning agent using an ion trap quantum processor, Quantum Sci. Technol. 4 (1) (2018) 015014.
[35] F. Leon, A.-Ş. Lupu, C. Bădică, Multiagent coalition structure optimization by quantum annealing, in: International Conference on Computational Collective Intelligence, Springer, 2017, pp. 331–341.
[36] F. Gul, Bargaining foundations of shapley value, Econometrica (1989) 81–95.
[37] P. Dubey, On the uniqueness of the Shapley value, Internat. J. Game Theory 4 (3) (1975) 131–139.
[38] D. Tang, Y. Cai, J. Zhao, Y. Xue, A quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems, Inform. Sci. 289 (2014) 162–189.

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. Her research interests include multi-agent system and machine learning.

Chubo Liu received the B.S. degree and Ph.D. degree in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. He is currently an associate professor of computer science and technology at Hunan University. His research interests are mainly in game theory, approximation and randomized algorithms, cloud and edge computing. He has published over 10 papers in journals and conferences such as the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Cloud Computing*, the *IEEE Transactions on Reliability*, the *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, the *Theoretical Computer Science*, and ICPADS.

Kenli Li (corresponding author) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He was a Visiting Scholar with the University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently a Full Professor of computer science and technology with Hunan University and an Associate Director with the National Supercomputing Center, Changsha, China. He has authored over 160 papers in international conferences and journals, such as the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Journal of Parallel and Distributed Computing, International Conference on Parallel Processing, and CCGrid. His major research interests include parallel computing, grid and cloud computing, and DNA computing. He is an Outstanding Member of CCF, and is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS.

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 1985. He is currently a Full Professor and the Dean of the School of Computer, National University of Defense Technology. His current research interests include parallel and distributed computing, high-performance computer systems, operating systems, cloud computing, and networked embedded systems. Mr. Liao is a member of the Association for Computing Machinery.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 720 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.