Liu Y, Zhang GF, Su ZP *et al.* Using computational intelligence algorithms to solve the coalition structure generation problem in coalitional skill games. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(6): 1136–1150 Nov. 2016. DOI 10.1007/s11390-016-1688-5

Using Computational Intelligence Algorithms to Solve the Coalition Structure Generation Problem in Coalitional Skill Games

Yang Liu [1], Guo-Fu Zhang [1,2,*], *Member, IEEE*, Zhao-Pin Su [1,2], *Member, IEEE*, Feng Yue [1,3], and Jian-Guo Jiang [1,2], *Senior Member, CCF*

[1] *School of Computer and Information, Hefei University of Technology, Hefei 230009, China*

[2] *Intelligent Manufacturing Institute, Hefei University of Technology, Hefei 230009, China*

[3] *Department of Science and Technology Management, Hefei University of Technology, Hefei 230009, China*

E-mail: liuyang@mail.hfut.edu.cn; {zgf, szp, yuefeng, jgjiang}@hfut.edu.cn

Received July 15, 2015; revised June 17, 2016.

Abstract Coalitional skill games (CSGs) are a simple model of cooperation in an uncertain environment where each agent has a set of skills that are required to accomplish a variety of tasks and each task requires a set of skills to be completed, but each skill is very hard to be quantified and can only be qualitatively expressed. Thus far, many computational questions surrounding CSGs have been studied. However, to the best of our knowledge, the coalition structure generation problem (CSGP), as a central issue of CSGs, is extremely challenging and has not been well solved. To this end, two different computational intelligence algorithms are herein evaluated: binary particle swarm optimization (BPSO) and binary differential evolution (BDE). In particular, we develop the two stochastic search algorithms with two-dimensional binary encoding and corresponding heuristic for individual repairs. After that, we discuss some fundamental properties of the proposed heuristic. Finally, we compare the improved BPSO and BDE with the state-of-the-art algorithms for solving CSGP in CSGs. The experimental results show that our algorithms can find the same near optimal solutions with the existing approaches but take extremely short time, especially under the large problem size.

Keywords coalitional skill game, coalitional structure generation, two-dimensional binary encoding, heuristic, individual repair

1 Introduction

In recent years, coalition formation is seen as a very important means of forming teams of autonomous agents that have to cooperate to perform certain tasks, and thus it has become a highly active area of multiagent systems (MAS) and artificial intelligence research[1-2]. Moreover, it has been successfully applied in many fields, from political sciences and economics[3-5], to operations research and computer science[6-8].

It is notable that much of the existing research on coalition formation, such as coalition structure games[9-10], overlapping coalition formation games[11-12], and coalitional resource games[13-14], assumes that the values of potential coalitions can be calculated with certainty[15-16], implying that the abilities or resources of each agent are quantifiable. However, in many important real-world scenarios under uncertainty[17-18], each agent has a set of skills that are very hard to be quantified and can only be qualitatively expressed, that is, each agent does not know or care about how strong each skill is[19-20].

With this in mind, Bachrach and Rosenschein[21] presented coalitional skill games (CSGs). In CSGs, each agent is endowed with a set of skills, each task requires a set of skills in order to be completed, a coalition can accomplish a task only if the coalition's members

Regular Paper

This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 61573125 and 61371155, and the Anhui Provincial Natural Science Foundation of China under Grant Nos. 1608085MF131, 1508085MF132, and 1508085QF129.

*Corresponding Author

cover the set of required skills for the task, and the gain for the coalition depends only on the subset of tasks it can fulfill. Besides, they investigated the computational complexity of several problems in CSGs, such as calculating the core, testing whether the core is empty, and computing the Shapley value[22].

Thereafter, Aziz *et al.*[23] showed the equivalence between simple CSGs and a subclass of multiple weighted voting games and analyzed the complexity of computing the cost of stability and the core of a simple coalitional skill game with a constant number of skills. Aziz and De Keijzer[24] proved that there exists a polynomial-time algorithm that computes an optimal partition for CSGs with a constant number of skills. Tran-Thanh *et al.*[25] relaxed the assumption that the dependence between skills and tasks is a binary relation and proposed a new coalitional skill vector model as a generalization of CSGs. In the vector model, each agent has a skill vector which consists of values that reflect its level in different skills, and each goal has a requirement expressed in terms of the minimum skill level necessary to achieve it.

There is no doubt that CSGs are a very simple model. Yet, they are quite expressive and can simulate many real-world scenarios. For example, in energy-constrained wireless sensor networks[26-27], each sensor node does not have other nodes' energy information, but forming coalitions for tasks to balance nodes' energy consumption and increase the lifetime of the whole network is often wise. Another example is agent-based web services and virtual organizations[28-32], and in these domains, intelligent agents form coalitions to respond to market opportunities as soon as possible, considering which service providers can provide relevant requested services, even if they do not know how strong the services of the providers are[33]. Additionally, such uncertainty is also natural and common in wireless communications[34-36], autonomous robotic systems[37], and electricity markets[38].

However, to date, as a central issue in CSGs, the coalition structure generation problem (CSGP) is extremely challenging and has not been well solved for the reason that CSGP is NP-complete[39]. To sovle CSGP, Bachrach *et al.*[39] proposed a fixed parameter tractable algorithm (which is called FPTA) on the basis of tree decomposition in hypergraph theory and showed that FPTA can solve the problem in time polynomial in the number of agents and skills, but exponential in the number of tasks and in the treewidth of skill-hypergraph. The main idea of FPTA can be shown as follows: first of all, considering the problem with a view to skills in CSGs and introducing hypergraph to describe the corresponding skill graph; second, implementing a tree decomposition of the given skill-hypergraph; and finally, coloring the agent in the tree and outputting the corresponding coalition structure, where all agents with the same color form a coalition. However, FPTA is extremely complicated and needs to consume a great deal of time, especially under large numbers of agents, skills, and tasks. In addition, FPTA is not often efficient in practice. The reason is that FPTA colors the tree nodes by backing-track coloring strategy which is a typical greedy strategy, while the greedy strategy makes a choice only according to the previous information, and once it has made a choice, this choice will not change no matter what the future outcome is. Therefore, FPTA cannot often get the global optimum solutions, but usually can obtain only the near optimal solutions. In particular, no any numerical result of FPTA has been provided in [39].

Recently, Nguyen[40] formulated the CSGP in CSGs as a complete set packing problem and used CPLEX to find the optimal coalition structure in the weighted CSGs. To realize the fast solution, Nguyen[40] proposed a fast approximation algorithm (henceforth called FAA) to reduce the dimensionality of CSGP as an integer programming problem. Beyond that, Nguyen[40] simplified the problem by the relaxation strategy. The empirical results show that FAA can find near optimal solutions. However, Nguyen[40] only performed numerical tests on CSGP with different numbers of agents, but did not evaluate the performance of FAA with different numbers of tasks and skills.

It is well known that the total number of coalitions is exponential with the number of agents[24]. Hence, as the search space complexity grows up, the cost of those algorithms (e.g., FPTA and FAA) can increase exponentially, making the search of a solution not feasible. Another way to tackle these problems is to find a suboptimal solution in a reasonable time, and moreover, in some cases, we may even find the optimal solution to the problem. In such techniques, computational intelligence (CI) algorithms, which are nature-inspired and population-based stochastic search techniques through the reproduction of generations, are becoming increasingly important and popular for solving computationally hard coalition formation problems[41-43]. Hence, in the present paper we intend to investigate the usefulness of binary particle swarm optimization (BPSO)[44] and binary differential evolution (BDE)[45] (see Section

3 for more details) in CI techniques and adopt BPSO and BDE to find suboptimal solutions in a reasonable time.

The remainder of this paper is structured as follows. Section 2 presents an overview of the CSGP in CSGs. In Section 3, we recall BPSO and BDE in CI technologies. After that, we show our algorithms to solving the CSGP in detail in Section 4, while, in Section 5, we measure the performance of the proposed algorithms in comparison with FPTA and FAA. Finally, Section 6 concludes the paper and presents future work.

2 Problem Formulations

Given the notations in Table 1, the model for CSGs in [21] can be recalled as follows. In a skill domain, there are a set of n agents, $A = \{a_1, \cdots, a_n\}$, a set of m tasks, $T = \{t_1, \cdots, t_m\}$, and a set of r skills, $S = \{s_1, \cdots, s_r\}$.

Table 1. Notations Used in the Model of CSGs

Variable	Definition
n	Number of agents
m	Number of tasks
r	Number of skills
A	Set of n agents
T	Set of m tasks
S	Set of r skills
a_i	The i-th agent
t_j	The j-th task
S_i	Set of skills of a_i
C	Set of member agents
$S(C)$	Set of skills of C
$T(C)$	Set of tasks that C can perform
$u(T(C))$	Value of $T(C)$

Each $t_j \in T$ ($j \in \{1, \cdots, m\}$) requires a set of skills $S(t_j) \subset S$, where $S(t_j) \neq \emptyset$. Each $a_i \in A$ ($i \in \{1, \cdots, n\}$) has a set of skills $S_i \subset S$.

A coalition C, $C \subseteq A$ and $C \neq \emptyset$, is a set of member agents. C also has a set of skills $S(C)$ which is a union of all the available skills that members can contribute to C, that is, $S(C) = \bigcup_{a_i \in C} S_i$. In addition, C can perform task t_j only if every skill required to accomplish t_j is owned by members in C, namely, $S(t_j) \subseteq S(C)$. It should be noted that C may fulfill many different tasks in T, and thus we denote the set of all the tasks that C can perform as $T(C) = \{t_j \in T | S(t_j) \subseteq S(C)\}$. Because the value of C, $v(C)$, is closely associated with the potential tasks that C will perform, in [39] a task value function maps a subset of the tasks that C can achieve to a real value, namely, $u : 2^T \to \mathbb{R}$, where u is monotone (if $T_1 \subset T_2 \subseteq T$, $u(T_1) \leqslant u(T_2)$) and satisfies $u(\emptyset) = 0$.

Formally, a coalitional skill game is a $(n + m + 4)$-tuple given by

$$\Gamma = (A, T, S, u, S_1, \cdots, S_n, S(t_1), \cdots, S(t_m)),$$

where the characteristic function of a coalition C is the value of the tasks that C can perform, namely, $v(C) = u(T(C))$.

In CSGs, CSGP is a central issue which aims to find the coalition structure (CS) with the maximal social welfare[39]. A CS is simply a partition of all the agents into several disjoint coalitions that work simultaneously. For example, $CS = (C_1, \cdots, C_e)$ is a coalition structure over A only if $\bigcup_{x=1}^{e} C_x = A$ and $C_x \cap C_y = \emptyset$ for all $x \neq y$ and $x, y \in \{1, \cdots, e\}$. Besides, the value of CS is given by $v(CS) = \sum_{C \in CS} v(C)$. Let $CS(A)$ denote a set of all the possible coalition structures on A, in general, the optimal coalition structure $CS^* \in CS(A)$ in a given coalitional skill game Γ is the partition of A which maximizes the social welfare, that is, for any other $CS \in CS(A)$, we have $v(CS) \leqslant v(CS^*)$.

Consider the following example.

Example 1. Consider the following coalitional skill game. There are four agents, $A = \{a_1, a_2, a_3, a_4\}$, with two possible tasks, $T = \{t_1, t_2\}$, and three skills $S = \{s_1, s_2, s_3\}$. The skill sets owned by each agent are $S_1 = \{s_1, s_2\}$, $S_2 = \{s_2, s_3\}$, $S_3 = \{s_1, s_3\}$, and $S_4 = \{s_3\}$. The skill sets required by each task are $S(t_1) = \{s_1, s_2\}$ and $S(t_2) = \{s_2, s_3\}$. To simplify the calculation of coalition value, Bachrach *et al.*[39] expressed the value of a coalition C as the number of tasks that C can accomplish, implying that the weight of each task is one. Then, CSGs can be called as task count skill games (TCSG). Considering the above case, we can easily find $CS^* = \{\{a_1, a_4\}, \{a_2, a_3\}\}$ is the optimal coalition structure and its value $v(CS^*) = 4$.

3 BPSO and BDE

CSGP is aimed at selecting a partition of the set of agents, and thus CSGP is a combinatorial optimization problem. There have been some techniques to solve simple CSGP, such as FPTA and FAA. As the search space complexity grows up, the cost of those algorithms can increase hugely (see Section 5 for more details), making the search of solutions infeasible. Particularly,

in a large number of coalition formation based practical applications[26-38], such as energy-constrained sensor networks, agent-based web services and virtual organizations, and wireless communications, finding the optimal solution is impossible under the real-time requirements, and an acceptable routine is to obtain a suboptimal solution in a reasonable time. For example, binary-encoding or integer-encoding based discrete computational intelligence (CI) algorithms, such as genetic algorithm[41-42] and simulated annealing[43] have been successfully applied in coalition formation.

In recent years, two new discrete computational intelligence algorithms BPSO and BDE have been becoming increasingly attractive. The reason is that, first, they have good exploration ability that has been experimentally verified in a wide variety of successful applications[46-50], and second, they have few parameters to be adjusted and are simple and quick.

In the domain of computational intelligence, BPSO and BDE are nature-inspired and population-based stochastic optimization techniques through the reproduction of generations. More specifically, the algorithms first initialize every individual to create the original population, use an evaluation function (or fitness function) to determine how good (or fit) each individual is, and then execute evolutionary operations to modify and update each individual to create the next generation. The algorithms will repeat the above steps to continue the evolution of every individual until a termination criterion is met (e.g., the maximal generation number reached or a solution with adequate objective function value found).

BPSO and BDE often perform well approximating solutions to many complex combinatorial optimization problems[46-50], especially when the search space is too large to search exhaustively, because they ideally do not make any assumption about the underlying fitness landscape in contrast with deterministic search approaches[41-43].

For further illustration, we recall BPSO and BDE in the following two subsections.

3.1 Binary Particle Swarm Optimization

The basic particle swarm optimization[51] often has a population (which is called a swarm) of candidate solutions (which are called particles). Let n_p be the number of particles in the swarm, each particle $s \in \{1, \cdots, n_p\}$ has a position vector $\boldsymbol{x}_s$ in the search-space and a velocity vector $\boldsymbol{v}_s$. These particles are moved and guided by their own best known position $\boldsymbol{p}_s$ as well as the entire swarm's best known position $\boldsymbol{g}$ according to the following simple formulae:

$$\boldsymbol{v}_s = \omega \boldsymbol{v}_s + c_1 r_1 (\boldsymbol{p}_s - \boldsymbol{x}_s) + c_2 r_2 (\boldsymbol{g} - \boldsymbol{x}_s),$$
$$\boldsymbol{x}_s = \boldsymbol{x}_s + \boldsymbol{v}_s,$$

where ω is a time-varying inertial weight and usually varies from 0.9 at the beginning to 0.4 toward the end of the optimization; c_1 and c_2 are two social parameters and usually both set to 2.0; r_1 and r_2 are two positive random numbers drawn from a uniform distribution between 0.0 and 1.0.

When improved positions are being discovered, $\boldsymbol{p}_s$ or $\boldsymbol{g}$ will be updated to guide the movements of the swarm. The process will be repeated until a satisfactory solution is discovered.

BPSO[44] is a discrete version of the original particle swarm optimization and uses the concept of velocity as a probability that a bit takes on "1" or "0". For more details, please check the following description of the procedure.

1) For each particle $s = 1, \cdots, n_p$ do:

• initialize the particle's position $\boldsymbol{x}_s$ in the search-space;

• evaluate the fitness of the particle;

• initialize the particle's best known position to its initial position: $\boldsymbol{p}_s \leftarrow \boldsymbol{x}_s$;

• if $\boldsymbol{p}_s$ is better than $\boldsymbol{g}$, update the swarm's best known position: $\boldsymbol{g} \leftarrow \boldsymbol{p}_s$;

• initialize the particle's velocity $\boldsymbol{v}_s$;

2) Until a termination criterion is met, repeat:

for each particle $s = 1, \cdots, n_p$ do:

• pick random numbers: $r_1, r_2, r_3 \sim U(0, 1)$;

• update the particle's velocity: $\boldsymbol{v}_s \leftarrow \omega \boldsymbol{v}_s + c_1 r_1 (\boldsymbol{p}_s - \boldsymbol{x}_s) + c_2 r_2 (\boldsymbol{g} - \boldsymbol{x}_s)$;

• update the particle's position (assume that x_s^d and v_s^d are a bit in $\boldsymbol{x}_s$ and $\boldsymbol{v}_s$, respectively):

$$x_s^d \leftarrow \begin{cases} 1, \text{ if } r_3 < \frac{1}{1+e^{-v_s^d}}, \\ 0, \text{ otherwise;} \end{cases}$$

• evaluate the fitness of the particle;

• if $\boldsymbol{x}_s$ is better than $\boldsymbol{p}_s$, update the swarm's best known position: $\boldsymbol{p}_s \leftarrow \boldsymbol{x}_s$;

• if $\boldsymbol{p}_s$ is better than $\boldsymbol{g}$, update the swarm's best known position: $\boldsymbol{g} \leftarrow \boldsymbol{p}_s$;

3) Output $\boldsymbol{g}$ which holds the best found solution.

3.2 Binary Differential Evolution

The standard differential evolution[52] explores the search space by three basic operations: mutation,

crossover, and selection. At each iteration, new candidate solutions (called donor vector) are created by the combination of individuals randomly chosen from the current population according to the simple mutation formulae. After that, the created new candidate solutions are mixed with a predetermined target vector, which is called crossover and produces the trial vector. Finally, the trial vector will be accepted for the next generation by selection operation if it has an advantage in the value of the objective function.

Differential evolution is simple and has few parameters, and it has good convergence in solving many continuous optimization problems. In order to apply differential evolution to solve problems defined for binary spaces, Pampará *et al.*[45] implemented the angle modulation approach within the standard differential evolution, as a mechanism to map a continuous-valued search space to a binary-valued search space.

The bit generating function in BDE is

$$g(x) = \sin(2\pi(x-a) \times b \times \cos(2\pi(x-a) \times c)) + d,$$

where x is a single element from a set of evenly separated intervals determined by the required number of bits that need to be generated (i.e., the dimension of the original, binary valued space), a represents the horizontal shift of the generating function, b represents the maximum frequency of the composed sin function, c represents the frequency of the composed cos function, d determines the vertical shift of the generating function.

Usually, the default coefficient values in BDE are set to $a = 0, b = 1, c = 1, d = 0$. More detailed descriptions of BDE are given below.

1) Set values of the control parameters:

• the mutation factor $F \in [0.4, 1]$,

• the crossover rate $Cr \in [0, 1]$,

• the population size n_p;

2) Initialize each individual with random positions in the search-space;

3) Until a stopping condition is met, repeat the following:

for each individual (with binary string $\boldsymbol{x}_s$) $s = 1, \cdots, n_p$, do:

• pick from the population at random three individuals $r_1, r_2, r_3 \sim U[1, n_p]$, which are distinct from each other as well as from s;

• generate a donor vector corresponding to the s-th target vector $\boldsymbol{x}_s$: $\boldsymbol{y}_s \leftarrow \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$;

• pick a random number $r_4 \sim U(0, 1)$;

• generate a trial vector for the s-th target vector:

$$\boldsymbol{z}_s \leftarrow \begin{cases} \boldsymbol{y}_s, \text{ if } r_4 \leqslant Cr, \\ \boldsymbol{x}_s, \text{ otherwise;} \end{cases}$$

• generate bit string by $g(\boldsymbol{z}_s)$ and pass it to the fitness function;

• if $\boldsymbol{z}_s$ is better than $\boldsymbol{x}_s$, update the target vector: $\boldsymbol{x}_s \leftarrow \boldsymbol{z}_s$;

4) Pick the individual from the population which has the best fitness.

4 Method of Solution

Although BPSO and BDE use easy one-dimensional binary encoding and are characterized by the fast working, they cannot be directly adopted to solve the CSGP in CSGs because one-dimensional binary encoding can produce only a coalition rather than a coalition structure. Hence, in the next three subsections, we will try to answer the following questions: How to represent a coalition structure? How to repair infeasible individuals? Is the proposed heuristic workable?

4.1 Encoding Scheme and Population Initialization

Fig.1 shows a 0-1 binary matrix of $(n+m) \times n$, representing a two-dimensional binary encoding. In the preceding n columns, each column denotes an agent, while in the residual m columns, each column denotes a task. On the other hand, each row represents a possible coalition and its satisfied task set. Note that there are at most n rows in the encoding, because there are at most n disjoint coalitions in a coalition structure with n agents. Let $x \in \{1, \cdots, n\}$ refer to a row and $y \in \{1, \cdots, (n+m)\}$ refer to a column, then $\delta_{x,y}$ denotes a bit in this encoding. For each $y \in \{1, \cdots, n\}$, if $\delta_{x,y} = 1$, a_y will join C_x. For each $y \in \{n+1, \cdots, n+m\}$, if $\delta_{x,y} = 1$, t_{y-n} is in the task set that will be handled by C_x. It is clear that, the last m columns can be used to represent the value of the evaluated coalition structure easily, because we just need to count the number of bit "1" in them.

In the beginning, as Fig.2 indicates, the initial population can be produced according to the constraint that each agent can join only a single coalition at any time. In Fig.2, $rand[1, n]$ represents an integer created randomly over the range $[1, n]$ and $rand[0, 1]$ denotes an integer generated randomly from $\{0, 1\}$. More specifically, in the y-th column, if the x^*-th row is selected

with $\delta_{x^*,y} \leftarrow 1$, all the other bits in column y must be set to "0".

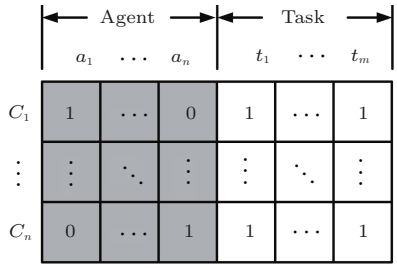


Fig.1. Two-dimensional binary encoding for an individual.

```
Each encoding is generated as follows:
for y := 1 to n do
    x* ← rand[1, n]
    δ_{x*,y} ← 1
    for x := 1 to n do
        if x ≠ x* then
            δ_{x,y} ← 0
        end if
    end for
end for
for y := n + 1 to n + m do
    for x := 1 to n do
        δ_{x,y} ← rand[0, 1]
    end for
end for
```

Fig.2. Pseudo-code for initializing individuals.

Consider the following example.

Example 2. Let us recall TCSG in example 1 and assume an individual is initialized as follows.

1	0	1	0	1	1
0	1	0	0	0	1
0	0	0	1	0	0
0	0	0	0	0	0

This individual denotes the coalition structure $CS = \{\{a_1, a_3\}, \{a_2\}, \{a_4\}\}$ and its value $v(CS) = 2+1+0 = 3$.

4.2 Heuristic for Individual Repairs

In CSGP, an agent can join only a single coalition at any time, while with the evolution of population, an individual is also in constant change and such situation may bring two problems that we summarize as follows.

• If $\exists y \in \{1, \cdots, n\}$ satisfies $\sum_{x=1}^{n} \delta_{x,y} > 1$, a_y has joined several different coalitions at the same time, which is contrary to the basic requirements of CSGP. Hence, whether an individual is feasible or not is closely related to the members in the coalition structure.

• If $\exists x \in \{1, \cdots, n\}$ and $\exists y \in \{n+1, \cdots, n+m\}$ satisfy $\delta_{x,y} = 1$ but $S(t_{y-n}) \nsubseteq S(C_x)$, t_{y-n} cannot be satisfied by C_x and thus C_x is infeasible. Hence, the feasibility of an individual is also inexorably linked to the task set of each coalition.

In fact, as long as either of the problems shown above occurs, an individual is just infeasible, which consumedly debases the availability of the population. Therefore, we develop a heuristic for individual repairs, which is illustrated in Fig.3, to ensure that each individual at every generation is feasible before they are passed to the fitness function.

```
for y ∈ {1, ··· , n} do
    Enumerate the bit "1" in column y
    if no bit "1" in column y then
        Select randomly a row x* and set δ_{x*,y} ← 1
    end if
    if more than a bit "1" in column y then
        Select randomly a bit "1" to be maintained and set
        other bits "1" to "0"
    end if
end for
for x ∈ {1, ··· , n} do
    for y ∈ {1, ··· , n} do
        if δ_{x,y} = 1 then
            C_x ← C_x + {a_y}
            S(C_x) ← S(C_x) ∪ S_y
        end if
    end for
    if C_x = ∅ then
        for y ∈ {n + 1, ··· , n + m} do
            if δ_{x,y} = 1 then
                δ_{x,y} ← 0
            end if
        end for
    else
        for y ∈ {n + 1, ··· , n + m} do
            if S(t_{y−n}) ⊆ S(C_x) and δ_{x,y} = 0 then
                δ_{x,y} ← 1
            end if
            if S(t_{y−n}) ⊈ S(C_x) and δ_{x,y} = 1 then
                δ_{x,y} ← 0
            end if
        end for
    end if
end for
```

Fig.3. Heuristic for individual repairs.

A brief description of the main stages of the heuristic is shown as follows.

• *Initializing.* Here, we give the original values of C_x and $S(C_x)$. For each $x \in \{1, \cdots, n\}$, $C_x \leftarrow \emptyset$ and $S(C_x) \leftarrow \emptyset$.

• *Checking Columns.* We should ensure that each agent does join only a single coalition. Thus, for each $y \in \{1, \ldots, n\}$, if there is no bit "1" in column y, namely, a_y does not join any coalition, we select randomly a row x^* and set $\delta_{x^*,y} \leftarrow 1$ to let a_y join C_{x^*},

because each agent has to join a coalition. On the other hand, if column y contains more than one bit "1", we select randomly a bit "1" to be maintained and set all the other bits "1" to "0".

• *Checking Rows.* When checking a row x, we first evaluate and record each member in C_x. For each $y \in \{1, \cdots, n\}$, if $\delta_{x,y} = 1$, $C_x \leftarrow C_x + \{a_y\}$ and $S(C_x) \leftarrow S(C_x) \cup S_y$. If $C_x = \emptyset$, namely, no agent joins C_x, C_x cannot satisfy any task, and thus we clear the corresponding task set, that is, for each $y \in \{n+1, \cdots, n+m\}$, if $\delta_{x,y} = 1$, $\delta_{x,y} \leftarrow 0$. If $C_x \neq \emptyset$, we should evaluate the possible target set that C_x can satisfy, and thus for each $y \in \{n+1, \cdots, n+m\}$, if $S(t_{y-n}) \subseteq S(C_x)$, namely, t_{y-n} can be satisfied by C_x, $\delta_{x,y} \leftarrow 1$, but if $S(t_{y-n}) \nsubseteq S(C_x)$, C_x cannot satisfy t_{y-n}, and thus we set $\delta_{x,y} \leftarrow 0$. Note that here each t_{y-n} that can be satisfied will be selected, even if the original $\delta_{x,y}$ is "0". The reason is that CSGP requests the evaluated coalition structure can satisfy as many tasks as possible.

It can be observed that the individual repair technique seems quite ad-hoc where the infeasible solution is repaired randomly. The reason is that CI algorithms, such as BPSO and BDE, are easy to fall into local optimum, especially dealing with huge amount of data. Repairing infeasible solutions randomly can create the diversity in the population and make the algorithm strive to escape from the local optima and mine the solutions as fully as possible. To further illustrate the heuristic, we consider the following example.

Example 3. Let us recall the TCSG in example 1. Table 2 shows an infeasible individual.

Table 2. Infeasible Individual

a_1	a_2	a_3	a_4	t_1	t_2
1	1	0	0	1	1
0	1	0	1	1	1
1	0	0	1	0	0
0	0	0	1	0	0

First, we check each column. In column 1, there are two bits "1", thus $\delta_{3,1}$ is selected to be maintained and $\delta_{1,1} \leftarrow 0$. Similarly, in column 2 $\delta_{2,2}$ is selected to be maintained and $\delta_{1,2} \leftarrow 0$. Because column 3 does not have bit "1", row 3 in column 3 is randomly selected and repaired to "1", namely, $\delta_{3,3} \leftarrow 1$. In column 4, there are three bits "1", and thus $\delta_{2,4}$ is randomly selected to be maintained but $\delta_{3,4} \leftarrow 0$ and $\delta_{4,4} \leftarrow 0$. Last, the individual is repaired as shown in Table 3.

Table 3. Repairs on Columns

a_1	a_2	a_3	a_4	t_1	t_2
[0]	[0]	0	0	1	1
0	1	0	1	1	1
1	0	[1]	[0]	0	0
0	0	0	[0]	0	0

Then, we check each row. In row 1, since $C_1 = \emptyset$, $\delta_{1,5} \leftarrow 0$ and $\delta_{1,6} \leftarrow 0$. In row 2, $C_2 = \{a_2, a_4\}$ and $S(C_2) = \{s_2, s_3\}$. Because $S(t_1) \nsubseteq S(C_2)$, $\delta_{2,5} \leftarrow 0$, while $S(t_2) \subseteq S(C_2)$, thus $\delta_{2,6} = 1$ is maintained. Similarly, in row 3, $C_3 = \{a_1, a_3\}$ and $S(C_3) = \{s_1, s_2, s_3\}$, because $S(t_1) \subseteq S(C_3)$ and $S(t_2) \subseteq S(C_3)$, $\delta_{3,5} \leftarrow 1$ and $\delta_{3,6} \leftarrow 1$. In addition, there is no bit "1" in row 4. Finally, the individual is repaired as shown in Table 4.

Table 4. Repairs on Rows

a_1	a_2	a_3	a_4	t_1	t_2
0	0	0	0	[0]	[0]
0	1	0	1	[0]	1
1	0	1	0	[1]	[1]
0	0	0	0	0	0

In the repaired individual, the coalition structure $CS = \{\{a_2, a_4\}, \{a_1, a_3\}\}$ and its value $v(CS) = 1+2 = 3$. As shown in the example, each $a_i \in A$ can only join a single coalition and each formed coalition can perform its tasks, which can ensure the feasibility of the individual.

It is natural to ask how hard it is to repair an individual and whether an individual can be repaired into a feasible one. We have the following results.

Proposition 1. *The worst case complexity of the heuristic for individual repairs is* $O(n \times (n+m) \times r)$.

Proof. In the heuristic, firstly, the previous n columns in an individual should be checked to let an agent only join a single coalition. Once column y is selected, at most $n-1$ bits "1" in column y should be revised to "0". Thus, the number of operations required to let each column contain a bit "1" is $O(n \times n) = O(n^2)$. Secondly, all the n rows in the individual should be checked to create the coalition structure. When checking a row x, there are at most n bits "1" in the previous n columns. For each member a_y in C_x, at most r skills should be memorized to calculate $S(C_x)$, and thus the number of operations required to evaluate each member in C_x is $O(n \times r)$. Then, if $C_x = \emptyset$, the target set should be cleared, that is, the residual m bits should be repaired to "0" and

the number of required operations is $O(m)$. However, if $C_x \neq \emptyset$, at most m tasks should be evaluated to determine the C_x's target set, while for each task t_{y-n}, at most r skills should be travelled to determine whether $S(t_{y-n})$ is a subset of $S(C_x)$, and thus the number of operations required to determine C_x's target set is $O(m \times r)$. Hence, the number of operations required to check rows is $O(n \times (n \times r + m \times r)) = O(n \times (n+m) \times r)$. To sum up, the worst-case complexity of the heuristic is $O(n^2 + n \times (n+m) \times r) = O(n \times (n+m) \times r)$. □

Proposition 2. *Each nonempty coalition C_x in a repaired individual can certainly satisfy its corresponding nonempty task target set $T(C_x)$.*

Proof. In the heuristic, if $C_x \neq \emptyset$, we evaluate each bit in the back m columns according to $S(C_x)$ and $S(t_{y-n})$. More specifically, if $S(t_{y-n}) \subseteq S(C_x)$, $\delta_{x,y} \leftarrow 1$ and t_{y-n} can be concluded into $T(C_x)$. On the other hand, if $S(t_{y-n}) \nsubseteq S(C_x)$, $\delta_{x,y} \leftarrow 0$ and t_{y-n} cannot be thrown into $T(C_x)$. Therefore, if $T(C_x) \neq \emptyset$, we have for each $t_{y-n} \in T(C_x)$, $S(t_{y-n}) \subseteq S(C_x)$, that is, each nonempty coalition in a repaired individual can certainly satisfy its nonempty target set. □

4.3 Fitness Function

We define the fitness function $f(\cdot)$ to determine how good a potential individual is and guide the search of population, satisfying

$$f(\cdot) = \max \sum_{x=1}^{n} \sum_{y=n+1}^{n+m} \delta_{x,y},$$

where $f(\cdot)$ is the number of tasks that the coalition structure CS in the individual can accomplish, that is, the value of CS. It is clear that, the bigger $f(\cdot)$ is, the more excellent an individual is, and thus the better the CS is.

5 Performance Evaluation

In order to demonstrate the performance of our heuristic-based BPSO and BDE, we have compared them with the existing state-of-the-art algorithms FPTA and FAA.

All the numerical tests that appear in this section are performed on a personal computer, Intel® Pentium® Dual 1.60 GHz with 2 GB of RAM and under the Windows 7 operating system. The code was written and tested on Visual Studio 2010. Following the existing work[39-40], all the datasets are randomly generated based on the given numbers of agents, tasks, and skills. The created $S(t_j)$ or S_i is a binary string in which "1" denotes that t_j or S_i requests or has the corresponding skill and "0" means that t_j or S_i does not need or does not have the corresponding skill. Each dataset is run for 50 independent trials with different random seeds. Additionally, based on the existing literature[46-50] and our testing on CSGP, we selected a group of appropriate experimental parameters for BPSO and BDE that are shown as follows.

- In BPSO, the population size is 20, the maximal generation number is 500, and c_1, c_2 are both set to 2.0.
- In BDE, the population size is 20, the maximal generation number is 500, the mutation factor is 0.8, and the crossover rate is 0.93.

5.1 Different *n*

In the first experiment, we tested the performance of BDE, BPSO, FPTA, and FAA with different numbers of agents (i.e., n, ranging from 5 to 30) with the fixed numbers of tasks and skills (i.e., $m = 2$ and $r = 5$).

Fig.4 shows the average coalition structure values obtained by the four algorithms. It can be observed that with the increase of n, the coalition structure values of all the algorithms are on the rise. The reason for this is that new agents have available skills and thus the given tasks can be satisfied more easily. Next, BPSO, BDE, and FPTA have obtained better coalition structure values than FAA on each test instance. When n is small, BPSO, BDE, and FPTA can get the same coalition structure values, but the situation changes when n is big. For example, when $n = 23$, the coalition structure value of FPTA is 11, but BPSO and BDE get

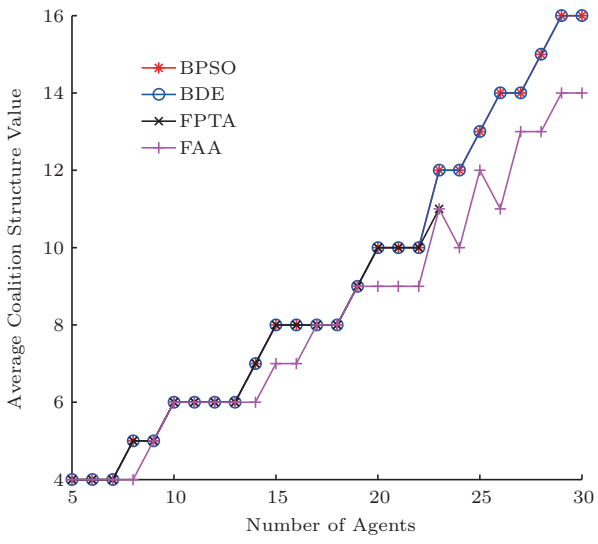


Fig.4. Average coalition structure value for different n.

the value 12. This indicates that FPTA is also easy to fall into local optimum and can only get the near optimal solutions in large domains because of the greedy backing-track coloring strategy in FPTA. In brief, when n is large, BPSO and BDE perform better than FPTA and FAA. Note that in Fig.4, we do not give the results of FPTA when n increases from 24 to 30. This is because when $n \geqslant 23$, FPTA is extremely slow and cannot output any solution within an acceptable time (see Table 5 for more details).

Table 5 shows the time required for the four algorithms with different numbers of agents. To make statistical inferences and powerful statements from the experimental data, here, we calculate the standard error of the mean, as well as the 95% confidence intervals[53] for the results of each algorithm on every test instance. With the increase of n, the average running time of the four algorithms has an increasing trend. When n is small, the difference among the four algorithms is not too big, but when n increases very much, BPSO and BDE are significantly faster than FPTA and FAA. For example, given $n = 23$, BPSO and BDE only took less than 0.02% of the time consumed by FPTA and 8.27% of the time consumed by FAA, and moreover, when $n \geqslant 23$, FPTA cannot obtain any solution in a reasonable time. The reason for this is that FPTA has to travel almost all the coalitions to maximize the coalition structure value, while there are 2^n coalitions at total for n agents. As for FAA, although the approximation method reduces the solution space and does not need to evaluate all the possible coalitions, it does calculate a large number of coalitions, especially when n is big. On the contrary, BPSO and BDE are population-based stochastic optimization techniques and thus they just need to evaluate a few coalitions by updating generations. The above results indicate that FPTA and FAA are more sensitive to the number of agents than BPSO and BDE.

Table 5. Total Execution Time Required (in Seconds) for BPSO, BDE, FPTA, and FAA on CSGP with Different n (Regarding the Standard Error of the Mean and the 95% Confidence Intervals)

n	BPSO	BDE	FPTA	FAA
5	0.19±3.52%	0.21±0.18%	0.07±0.76%	0.56±1.90%
6	0.29±0.81%	0.36±0.67%	0.24±0.87%	0.37±2.29%
7	0.33±0.92%	0.46±1.62%	0.28±1.32%	0.64±3.17%
8	0.36±0.83%	0.44±1.16%	0.29±0.94%	1.70±7.28%
9	0.39±0.39%	0.48±1.42%	0.34±1.15%	1.66±2.25%
10	0.43±1.13%	0.47±0.67%	0.36±1.05%	1.72±4.20%
11	0.65±4.27%	0.69±1.57%	7.05±31.77%	1.75±6.35%
12	0.66±1.10%	0.71±1.26%	7.84±14.30%	1.81±3.96%
13	0.68±0.73%	0.75±1.32%	9.71±73.81%	1.96±5.74%
14	0.74±0.89%	0.82±1.48%	10.70±29.09%	4.27±13.69%
15	0.85±3.78%	0.87±0.99%	13.39±86.40%	4.70±9.95%
16	1.01±1.34%	1.11±2.34%	264.93±555.58%	6.15±12.98%
17	1.09±0.94%	1.15±1.58%	312.30±281.21%	9.25±16.35%
18	1.13±1.26%	1.21±1.59%	320.59±1 175.03%	8.97±11.87%
19	1.26±5.52%	1.30±0.96%	380.43±2 336.05%	8.50±14.08%
20	1.31±4.42%	1.34±2.54%	396.46±2 972.13%	12.79±14.99%
21	1.69±6.84%	1.68±3.13%	6 905.55±5 751.22%	12.68±27.05%
22	1.70±2.27%	1.70±1.42%	8 278.21±20 627.85%	13.09±20.15%
23	1.74±1.32%	1.79±1.42%	9 944.05±21 283.06%	21.65±40.69%
24	2.13±5.10%	2.16±5.57%	–	16.54±38.44%
25	2.21±4.62%	2.23±2.45%	–	20.79±30.90%
26	2.28±5.51%	2.28±5.89%	–	20.79±30.35%
27	2.60±8.48%	2.69±2.86%	–	23.09±42.88%
28	2.88±5.64%	2.79±4.84%	–	23.60±18.14%
29	2.91±10.73%	2.90±6.32%	–	27.98±42.70%
30	3.35±7.57%	3.23±2.11%	–	27.05±28.34%

5.2 Different m

In this experiment, given different numbers of tasks (i.e., m, ranging from 2 to 22), we evaluate the four algorithms with the fixed numbers of agents and skills (i.e., $n = 5$ and $r = 5$). Note that here n is small, because FPTA and FAA are sensitive to n according to the results in the first experiment, and too big n is not beneficial for the analysis of the impact of parameter m.

Fig.5 shows the average coalition structure value for different m. As it depicts, with the increase of m, the coalition structure values of all the algorithms also increase, and moreover, BPSO, BDE, and FAA have obtained the same values. This is because the solution space with $n = 5$ is extremely small for the three algorithms, and meanwhile, the new added tasks with requested skills can be easily satisfied by the existing five agents. Note that in Fig.5, we do not give the results of FPTA when $m \geqslant 7$, because FPTA cannot output a solution in an acceptable time (see Table 6 for more details).

The total execution time of each algorithm for different m is shown in Table 6. As can be seen, the average running time of BPSO, BDE, and FAA increases extremely slowly, and FAA spends a bit more time than

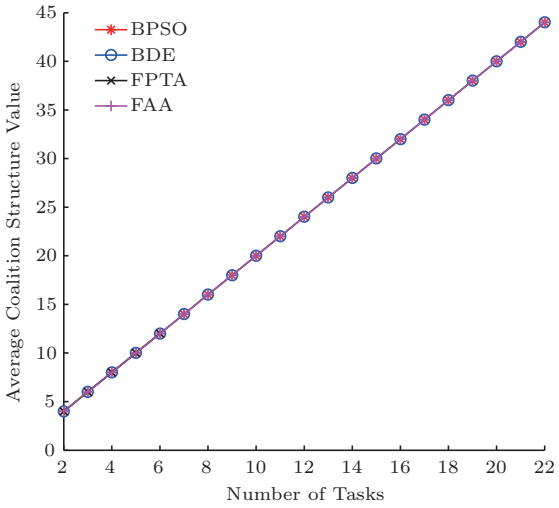


Fig.5. Average coalition structure value for different m.

BPSO and BDE on almost each test instance, but the difference among the three algorithms is not too big even if m is very large, which forms a sharp contrast with the results in the first experiment. In contrast, the average running time of FPTA increases greatly, which coincides with the previous results. Given $m = 6$, for example, BPSO and BDE only took less than 0.000 8% of the time consumed by FPTA and 55.38% of the time

Table 6. Total Execution Time Required (in Seconds) for BPSO, BDE, FPTA, and FAA on CSGP with Different m (Regarding the Standard Error of the Mean and the 95% Confidence Intervals)

m	BPSO	BDE	FPTA	FAA
2	0.19±2.16%	0.22±0.30%	0.07±0.87%	0.56±1.65%
3	0.22±0.85%	0.26±0.26%	1.16±3.45%	0.49±1.88%
4	0.26±1.32%	0.32±1.88%	35.85±39.52%	0.48±2.03%
5	0.29±0.69%	0.33±0.37%	1 390.53±2 466.88%	0.48±1.36%
6	0.33±0.78%	0.36±0.35%	48 973.83±199 526.10%	0.65±2.64%
7	0.35±0.67%	0.38±0.94%	–	0.65±2.99%
8	0.38±0.88%	0.43±1.22%	–	0.60±2.70%
9	0.42±0.67%	0.45±0.38%	–	0.64±2.60%
10	0.40±0.53%	0.48±0.29%	–	0.67±4.68%
11	0.46±1.76%	0.53±1.29%	–	0.64±3.47%
12	0.46±0.75%	0.54±0.24%	–	0.83±4.13%
13	0.49±0.39%	0.60±1.57%	–	0.80±2.77%
14	0.52±0.61%	0.61±1.02%	–	0.80±2.14%
15	0.54±0.83%	0.64±0.75%	–	0.80±1.76%
16	0.57±0.71%	0.70±1.22%	–	0.80±2.14%
17	0.59±0.67%	0.72±1.95%	–	0.83±1.88%
18	0.62±0.83%	0.73±0.37%	–	0.80±1.91%
19	0.66±2.31%	0.81±1.17%	–	0.80±2.26%
20	0.69±0.52%	0.81±0.55%	–	0.81±1.83%
21	0.74±2.29%	0.83±1.53%	–	0.84±2.10%
22	0.73±0.63%	0.92±3.33%	–	0.79±6.93%

consumed by FAA, and moreover, when $m \geqslant 7$, FPTA cannot get any solution in a reasonable time. The reason is that the number of possible coalitions has nothing to do with the number of tasks and is extremely small under small n, and thus BPSO, BDE, and FAA just need to evaluate a few coalitions to calculate the coalition structure value. However, although FPTA also needs to evaluate a few coalitions, it has to travel and evaluate all the possible subsets of T to maximize the value of achievable candidate task solutions for a given coalition structure, and thus with the increase of m, FPTA has to face and explore a huge search space, which results in a large amount of time consumption. The above results indicate that FPTA is extremely sensitive to the number of tasks, while BPSO, BDE, and FAA are not sensitive to m.

5.3 Different r

In this experiment, we measure the four algorithms based on different numbers of skills (i.e., r, ranging from 5 to 30) and the fixed numbers of agents and tasks (i.e., $n = 5$ and $m = 2$). Note that based on the previous experiments, we know FAA is sensitive to n and FPTA is sensitive to both n and m, thereby here we select small n and m to make the comparisons as fair as possible.

Fig.6 shows the average coalition structure value for different r. It can be observed that with the increase of r, the four algorithms can get the same values, but the coalition structure value of each algorithm does not increase. The reason is that the solution space under $n = 5$ and $m = 2$ is extremely small, and meanwhile, the new added skills cannot change the numbers of possible coalitions and target sets, and thus the whole solution space for the four algorithms is fixed and extremely limited.

Table 7 illustrates the total execution time of each algorithm for different r. It can be seen that with the growing r, the average running time of the four algorithms increases extremely slowly, and FAA takes slightly more time than BPSO, BDE, and FPTA on each test instance, but the difference among the four algorithms is not too big, which is significantly different from the empirical results in the previous experiments. Given $r = 30$, for example, BPSO and BDE are a little slower than FPTA but only take less than 62.97% of the time consumed by FAA. The reason is that in this experiment n and m are extremely small for fair comparison and the possible coalitions and target sets are extremely limited. In this case, BPSO, BDE, and FPTA just need to explore an extremely small solution

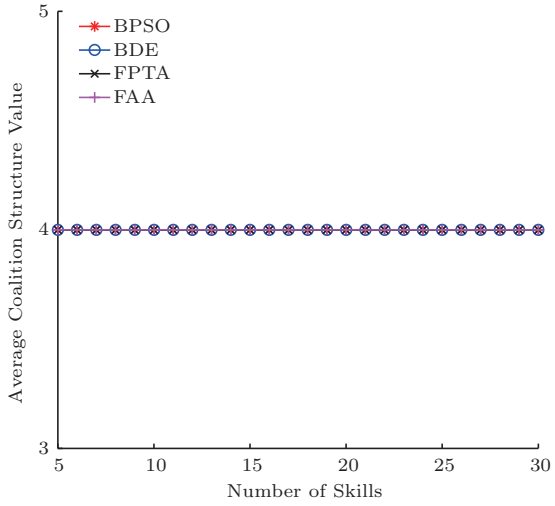


Fig.6. Average coalition structure value for different r.

Table 7. Total Execution Time Required (in Seconds) for BPSO, BDE, FPTA, and FAA on CSGP with Different r (Regarding the Standard Error of the Mean and the 95% Confidence Intervals)

r	BPSO	BDE	FPTA	FAA
5	0.17±3.51%	0.22±0.22%	0.07±0.81%	0.56±1.29%
6	0.19±1.00%	0.24±0.26%	0.10±0.99%	0.53±3.25%
7	0.19±1.11%	0.25±0.65%	0.10±0.86%	0.57±1.87%
8	0.19±0.78%	0.25±0.27%	0.11±0.77%	0.54±2.41%
9	0.20±1.68%	0.26±0.64%	0.12±0.70%	0.56±2.08%
10	0.20±3.09%	0.27±0.64%	0.12±1.08%	0.57±2.59%
11	0.20±0.48%	0.27±0.96%	0.13±0.75%	0.59±3.21%
12	0.20±1.94%	0.28±2.10%	0.13±0.78%	0.56±2.70%
13	0.20±2.47%	0.29±1.59%	0.15±0.99%	0.58±5.47%
14	0.21±2.30%	0.28±0.56%	0.14±0.88%	0.53±1.76%
15	0.20±0.59%	0.30±1.70%	0.15±1.41%	0.56±2.22%
16	0.20±0.64%	0.31±1.33%	0.15±0.73%	0.57±2.00%
17	0.21±1.08%	0.29±0.18%	0.16±0.90%	0.55±1.72%
18	0.21±0.34%	0.31±2.05%	0.16±0.86%	0.56±2.35%
19	0.20±0.54%	0.31±1.02%	0.17±0.95%	0.56±1.71%
20	0.21±0.75%	0.33±4.17%	0.17±0.90%	0.57±2.42%
21	0.21±0.67%	0.31±0.59%	0.18±1.01%	0.56±2.38%
22	0.21±1.20%	0.32±0.74%	0.19±1.18%	0.57±1.64%
23	0.22±1.43%	0.31±0.51%	0.18±1.01%	0.52±2.55%
24	0.22±0.74%	0.32±0.67%	0.18±0.99%	0.56±2.08%
25	0.21±0.65%	0.33±0.83%	0.19±0.99%	0.57±2.34%
26	0.23±0.74%	0.33±0.56%	0.20±1.68%	0.54±1.80%
27	0.21±0.47%	0.33±0.86%	0.23±1.24%	0.56±2.14%
28	0.21±0.78%	0.34±0.84%	0.21±1.18%	0.58±2.65%
29	0.23±1.37%	0.34±0.89%	0.21±1.10%	0.54±3.14%
30	0.22±0.35%	0.34±0.53%	0.21±1.02%	0.54±2.85%

space. On the contrary, executing the dimensionality reduction and relaxation in such a small solution space is complete waste of effort and time for FAA. The above results indicate that all the four algorithms are not sensitive to the number of skills.

5.4 Discussion

The reliability and the effectiveness of the proposed computational intelligence algorithms BPSO and BDE are validated from the results of statistical analysis based on 50 independent runs of each algorithm and different numbers of agents, tasks, and skills. We find that CSGP in CSGs is closely related to the numbers of agents and tasks, but has little relation with the number of skills.

Our heuristic-based BPSO and BDE are efficient no matter how parameters n, m, and r change, which justifies the utility of computational intelligence algorithms in practice, especially in large domains. Specifically, when the number of agents n is big, BPSO and BDE are far more effective than FPTA and FAA in terms of both the coalition structure value and the time consumption, which may indicate that the proposed heuristic for individual repairs can support strong heuristic information to guide the evolution of population. Additionally, BPSO is a little faster than BDE because unlike BDE, BPSO has no evolution operators such as crossover and mutation.

FAA is sensitive to parameter n. When there are too many agents, the performance of FAA will decrease much. Yet, one bright spot for FAA is that it is not affected by the growing number of tasks. Hence, FAA is feasible in scenarios where there are a large number of tasks but few agents.

As for FPTA, its performance is greatly restricted by parameters n and m. When the number of agents or tasks is big, FPTA performs extremely poorly and cannot give any solution in a reasonable time. Accordingly, FPTA is serviceable only in scenarios where there are a large number of skills but few agents and tasks.

The above observations indicate that BPSO and BDE are more feasible and robust for CSGP in large domains. Hence, computational intelligence algorithms are powerful and promising tools for solving complicated and hard computing problems related to coalition formation. In particular, the two-dimensional binary encoding not only is easy to comprehend, but also fits well with the combinational characteristic of CSGP. Moreover, there are at most n rows in an individual for a coalition structure contains at most n coalitions; thus the evaluation space in computational intelligence algorithms is extremely limited, which further supports their potential utility in other cooperative games, such as weighted voting games[54], graph games[55-57], and network flow games[58-59].

Weighted voting games[54] are games where each agent has a weight, and a coalition of agents wins if the total weight of its members meets or exceeds a certain threshold and loses otherwise. The CSGP in these games is making the winning coalitions in a coalition structure as many as possible. Graph games are games played over a graph in which agents are the vertices and each edge has a weight. The value of a coalition is the total weights of all the valid edges connected by its members. The CSGP in graph games is just finding a partition of agents to maximize the total weights of all the valid edges in a coalition structure. It is clear that a coalition structure in the above games can also be represented by the two-dimensional binary encoding, but different games may need different heuristics for individual repairs.

Network flow games are also games played over a directed network flow graph in which each edge has a capacity and is controlled by an agent. A coalition of agents wins if it can send a flow that is above the threshold value from a source vertex to a target vertex, and loses otherwise. Unlike CSGs, weighted voting games, and graph games, the solution to a network flow problem influences the value of a coalition, thereby one of the challenges is finding a coalition rather than a coalition structure to achieve the maximal flow between the given source vertex and the target vertex. For this problem, we can also use computational intelligence algorithms, such as one-dimensional binary encoding based BPSO and BDE, or integer encoding based ant colony optimization that has been successfully applied in searching for an optimal coalition for the given task[60].

6 Conclusions

This paper's methodology provides computational intelligence algorithms for solving CSGP in CSGs. The proposed framework includes a two-dimensional binary encoding scheme and a heuristic for individual repairs that corrects the infeasibility in the matrices created in the process of population evolution. To benchmark the effectiveness of our algorithms, we tested the proposed algorithms based on two popular discrete computational intelligence algorithms BPSO and BDE, and

compared their performance with that of the existing state-of-the-art algorithms FPTA and FAA. The empirical results in terms of the coalition structure value and running time showed that our heuristic-based BPSO and BDE are more efficient and effective than FPTA and FAA, especially when the problem size is large. For example, for the case of 23 agents, the solutions determined by BPSO and BDE are more precise than those by FPTA and FAA, and meanwhile, BPSO and BDE took less than 0.02% of the time consumed by FPTA and 8.27% of the time consumed by FAA, respectively.

However, we did not imply in this paper that our algorithms are always superior to FPTA and FAA. From a more practical viewpoint, this work can be seen as an initial step toward a reasonable guide for solving CSGP from the perspective of CI techniques, which may be helpful for solving practical coalition formation problems in large domains. This work still has some limitations to be improved: these algorithms (i.e., BPSO, BDE, FPTA, and FAA) are not evaluated on real-world test instances; the strengths and the weaknesses of these algorithms are not studied on CSGP in comparison with alternative CI algorithms. Additionally, it would be interesting to investigate the serviceability of CI algorithms in other cooperative games.

Acknowledgements The authors would like to thank the editors and anonymous referees for their insightful comments and suggestions.

References

[1] Khan M A, Turgut D, Bölöni L. Optimizing coalition formation for tasks with dynamically evolving rewards and nondeterministic action effects. *Autonomous Agents and Multi-Agent Systems*, 2011, 22(3): 415-438.

[2] Ye D, Zhang M, Sutanto D. Self-adaptation-based dynamic coalition formation in a distributed agent network: A mechanism and a brief survey. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(5): 1042-1051.

[3] Li C, Sycara K, Scheller-Wolf A. Combinatorial coalition formation for multi-item group-buying with heterogeneous customers. *Decision Support Systems*, 2010, 49(1): 1-13.

[4] Liao S S, Zhang J D, Lau R *et al*. Coalition formation based on marginal contributions and the Markov process. *Decision Support Systems*, 2014, 57(1): 355-363.

[5] Ke G Y, Bookbinder J H, Kilgour D M. Coordination of transportation and quantity discount decisions, with coalition formation. *International Journal of Production Research*, 2014, 52(17): 5115-5130.

[6] Saad W, Han Z, Başar T *et al*. Hedonic coalition formation for distributed task allocation among wireless agents. *IEEE Transactions on Mobile Computing*, 2011, 10(9): 1327-1344.

[7] Guerrero J, Oliver G. Multi-robot coalition formation in real-time scenarios. *Robotics and Autonomous Systems*, 2012, 60(10): 1295-1307.

[8] Liu A, Li Q, Huang L *et al*. Coalitional game for community-based autonomous web services cooperation. *IEEE Transactions on Services Computing*, 2013, 6(3): 387-399.

[9] Service T C, Adams J A. Constant factor approximation algorithms for coalition structure generation. *Autonomous Agents and Multi-Agent Systems*, 2011, 23(1): 1-17.

[10] Rahwan T, Michalak T, Wooldridge M *et al*. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence*, 2012, 186: 95-122.

[11] Zick Y, Chalkiadakis G, Elkind E. Overlapping coalition formation games: Charting the tractability frontier. In *Proc. the 11th International Conference on Autonomous Agents and Multiagent Systems*, June 2012, pp.787-794.

[12] Zick Y, Markakis E, Elkind E. Arbitration and stability in cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, 2014, 50: 847-884.

[13] Dunne P E, Kraus S, Manisterski E *et al*. Solving coalitional resource games. *Artificial Intelligence*, 2010, 174(1): 20-50.

[14] Chitnis R, Hajiaghayi M, Liaghat V. Parameterized complexity of problems in coalitional resource games. In *Proc. the 25th AAAI Conference on Artificial Intelligence*, August 2011, pp.620-625.

[15] Rahwan T, Jennings N R. An algorithm for distributing coalitional value calculations among cooperating agents. *Artificial Intelligence*, 2007, 171(8/9): 535-567.

[16] Airiau S, Sen S. A fair payoff distribution for myopic rational agents. In *Proc. the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2009, pp.1305-1306.

[17] Chalkiadakis G, Markakis E, Boutilier C. Coalition formation under uncertainty: Bargaining equilibria and the Bayesian core stability concept. In *Proc. the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007, pp.412-419.

[18] Chalkiadakis G, Boutilier C. Sequentially optimal repeated coalition formation under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 2012, 24(3): 441-484.

[19] Kenari S M S, Jahan M V, Jalali M. Multi-skill agents coalition formation under skill uncertainty. In *Proc. International Symposium on Artificial Intelligence and Signal Processing*, June 2011, pp.89-96.

[20] Sridhar U, Mandyam S. Capability-weighted group utility maximizer for network coalitional games under uncertainty. In *Proc. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, August 2012, pp.613-617.

[21] Bachrach Y, Rosenschein J S. Coalitional skill games. In *Proc. the 7th International Conference on Autonomous Agents and Multiagent Systems*, May 2008, pp.1023-1030.

[22] Bachrach Y, Parkes D C, Rosenschein J S. Computing cooperative solution concepts in coalitional skill games. *Artificial Intelligence*, 2013, 204: 1-21.

[23] Aziz H, Brandt F, Harrenstein P. Monotone cooperative games and their threshold versions. In *Proc. the 9th International Conference on Autonomous Agents and Multiagent Systems*, May 2010, pp.1107-1114.

[24] Aziz H, De Keijzer B. Complexity of coalition structure generation. In *Proc. the 10th International Conference on Autonomous Agents and Multiagent Systems*, May 2011, pp.191-198.

[25] Tran-Thanh L, Nguyen T D, Rahwan T *et al*. An efficient vector-based representation for coalitional games. In *Proc. the 23rd International Joint Conference on Artificial Intelligence*, August 2013, pp.383-389.

[26] Wu D, Cai Y, Wang J. A coalition formation framework for transmission scheme selection in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 2011, 60(6): 2620-2630.

[27] Akkarajitsakul K, Hossain E, Niyato D. Coalition-based cooperative packet delivery under uncertainty: A dynamic bayesian coalitional game. *IEEE Transactions on Mobile Computing*, 2013, 12(2): 371-385.

[28] Blankenburg B, He M, Klusch M *et al*. Risk-bounded formation of fuzzy coalitions among service agents. In *Proc. the 10th International Workshop on Cooperative Information Agents*, September 2006, pp.332-346.

[29] Wei G, Vasilakos A V, Zheng Y *et al*. A game-theoretic method of fair resource allocation for cloud computing services. *The Journal of Supercomputing*, 2010, 54(2): 252-269.

[30] Niyato D, Vasilakos A V, Zhu K. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. In *Proc. the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2011, pp.215-224.

[31] Huang B, Gao C, Chen L. Partner selection in a virtual enterprise under uncertain information about candidates. *Expert Systems with Applications*, 2011, 38(9): 11305-11310.

[32] Rahimi M R, Venkatasubramanian N, Vasilakos A V. MuSIC: Mobility-aware optimal service allocation in mobile cloud computing. In *Proc. the 6th IEEE International Conference on Cloud Computing*, June 28-July 3, 2013, pp.75-82.

[33] Sheng Q Z, Qiao X, Vasilakos A V *et al*. Web services composition: A decade's overview. *Information Sciences*, 2014, 280: 218-238.

[34] Khan M A, Tembine H, Vasilakos A V. Game dynamics and cost of learning in heterogeneous 4G networks. *IEEE Journal on Selected Areas in Communications*, 2012, 30(1): 198-213.

[35] Duarte P B F, Fadlullah Z M, Vasilakos A V *et al*. On the partially overlapped channel assignment on wireless mesh network backbone: A game theoretic approach. *IEEE Journal on Selected Areas in Communications*, 2012, 30(1): 119-127.

[36] Wang C Y, Ko C H, Wei H Y *et al*. A boting-based Femtocell downlink cell-breathing control mechanism. *IEEE/ACM Transactions on Networking*, 2016, 24(1): 85-98.

[37] Dutta A, Dasgupta P, Baca J *et al*. SearchUCSG: A fast coalition structure search algorithm for modular robot reconfiguration under uncertainty. *Robotica*, 2014, 32(2): 225-244.

[38] Roh J H, Shahidehpour M, Wu L. Market-based generation and transmission planning with uncertainties. *IEEE Transactions on Power Systems*, 2009, 24(3): 1587-1598.

[39] Bachracht Y, Meir R, Jung K *et al*. Coalitional structure generation in skill games. In *Proc. the 24th AAAI Conference on Artificial Intelligence*, July 2010, pp.703-708.

[40] Nguyen T D. A fast approximation algorithm for solving the complete set packing problem. *European Journal of Operational Research*, 2014, 237(1): 62-70.

[41] Sen S, Dutta P. Search for optimal coalition structures. In *Proc. the 4th International Conference on Multi-Agent Systems*, July 2000, pp.287-292.

[42] Yang J, Luo Z. Coalition formation mechanism in multi-agent systems based on genetic algorithms. *Applied Soft Computing*, 2007, 7(2): 561-568.

[43] Keinänen H. Simulated annealing for multi-agent coalition formation. In *Proc. the 3rd KES International Symposium on Agent and Multi-Agent Systems-Technologies and Applications*, June 2009, pp.30-39.

[44] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm. In *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, October 1997, pp.4104-4108.

[45] Pampará G, Engelbrecht A P, Franken N. Binary differential evolution. In *Proc. IEEE International Conference on Evolutionary Computation*, July 2006, pp.1873-1879.

[46] Pedrasa M A A, Spooner T D, MacGill I F. Scheduling of demand side resources using binary particle swarm optimization. *IEEE Transactions on Power Systems*, 2009, 24(3): 1173-1181.

[47] Cao R F, Wang X C, Wu Z K *et al*. A parallel Markov cerebrovascular segmentation algorithm based on statistical model. *Journal of Computer Science and Technology*, 2016, 31(2): 400-416.

[48] Du H Z, Xia N, Jiang J G *et al*. A Monte Carlo enhanced PSO algorithm for optimal QoM in multi-channel wireless networks. *Journal of Computer Science and Technology*, 2013, 28(3): 553-563.

[49] He R J, Yang Z Y. Differential evolution with adaptive mutation and parameter control using Lévy probability distribution. *Journal of Computer Science and Technology*, 2012, 27(5): 1035-1055.

[50] Lu X F, Tang K. Classification- and regression-assisted differential evolution for computationally expensive problems. *Journal of Computer Science and Technology*, 2012, 27(5): 1024-1034.

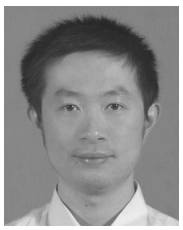
[51] Kennedy J, Eberhart R C. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, November 1995, pp.1942-1948.

[52] Storn R, Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997, 11(4): 341-359.

[53] Altman D G, Machin D, Bryant T N *et al*. Statistics with Confidence: Confidence Intervals and Statistical Guidelines. London, England: BMJ Books, 2000.

[54] Elkind E, Chalkiadakis G, Jennings N R. Coalition structures in weighted voting games. In *Proc. the 18th European Conference on Artificial Intelligence*, July 2008, pp.393-397.

[55] Deng X, Papadimitriou C H. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 1994, 19(2): 257-266.
[56] Voice T, Polukarov M, Jennings N R. Coalition structure generation over graphs. *Journal of Artificial Intelligence Research*, 2012, 45(1): 165-196.
[57] Bachrach Y, Kohli P, Kolmogorov V *et al*. Optimal coalition structure generation in cooperative graph games. In *Proc. the 27th AAAI Conference on Artificial Intelligence*, July 2013, pp.81-87.
[58] Kalai E, Zemel E. Totally balanced games and games of flow. *Mathematics of Operations Research*, 1982, 7(3): 476-478.
[59] Bachrach Y, Rosenschein J S. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems*, 2009, 18(1): 106-132.
[60] Xia N, Jiang J, Hu C. Solution to agent coalition problem using improved ant colony optimization algorithm. In *Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, September 2004, pp.475-478.



Yang Liu received his B.S. degree in communication engineering and his M.S. degree in electronic information engineering from Hefei University of Technology, Hefei, in 2005 and 2008, respectively. He is currently pursuing his Ph.D. degree in information and communication engineering at the School of Computer and Information, Hefei University of Technology, Hefei. His research interests include modeling and simulation of complex systems, image processing, and distributed artificial intelligence.

Guo-Fu Zhang received his B.S. and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, in 2002 and 2008, respectively. He is currently an associate professor of the Agents, Coalition, Decision Group, the School of Computer and Information, Hefei University of Technology, Hefei. His research interests include evolutionary computation, intelligent agents, and search-based software engineering.

Zhao-Pin Su received her B.S. and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, in 2004 and 2008, respectively. She is currently an associate professor of the Agents, Coalition, Decision Group, the School of Computer and Information, Hefei University of Technology, Hefei. She is interested in agent-based simulation, coalition formation for disaster response, and collaborative decision support systems for disaster response management.

Feng Yue received his B.S., M.S., and Ph.D. degrees in computer science from Hefei University of Technology, Hefei, in 2004, 2009, and 2015, respectively. He is currently an associate researcher at the Department of Science and Technology Management, Hefei University of Technology, Hefei. His research interests include automatic control, image processing, and software engineering.

Jian-Guo Jiang received his M.S. degree in signals, circuits, and systems from Hefei University of Technology, Hefei, in 1989. He is currently a professor of the School of Computer and Information, Hefei University of Technology, Hefei. He is the head of the Texas Instruments — Hefei University of Technology DSPS Laboratory. His research interests include automatic control, image processing, and multi-agent systems.