Applied Soft Computing 11 (2011) 2164–2172

Contents lists available at ScienceDirect

Applied Soft Computing

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

A revision algorithm for invalid encodings in concurrent formation of overlapping coalitions

Guofu Zhang [a,b,c,d,,*], Jianguo Jiang [a,c,d], Changhua Lu [a,b,c,d], Zhaopin Su [a,e], Hua Fang [a,d], Yang Liu [a,d]

[a] School of Computer and Information, Hefei University of Technology, Hefei 230009, China
[b] Information and Communication Engineering Postdoctoral Research Station, Hefei University of Technology, Hefei 230009, China
[c] National Engineering Research Center of Special Display Technology, Hefei 230009, China
[d] Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei 230009, China
[e] Management Science and Engineering Postdoctoral Research Station, Hefei University of Technology, Hefei 230009, China

ARTICLE INFO

Article history:
Received 25 May 2008
Received in revised form 22 July 2010
Accepted 25 July 2010
Available online 4 August 2010

Keywords:
Concurrent tasks
Overlapping coalitions
Two-dimensional binary encoding
Invalid encodings
Revision algorithm

ABSTRACT

In multi-agent systems, agents are inclined to form coalitions to improve individual performance and perform tasks more efficiently. However, the most existing researches assume that the desired outcome is a coalition structure that consists of disjoint coalitions where every agent that has joined a coalition is no longer available to join other coalitions, which leads to waste of resources. In a number of practical scenarios an agent may have to be involved in executing more than one task simultaneously, and distributing its resources to several completely different coalitions. To tackle such problems, we propose the concurrent formation of overlapping coalitions and introduce a two-dimensional binary encoding to search the coalition space. We mainly focus on the revision algorithm for invalid encodings. Specifically, by using the proposed revision algorithm, an agent may join in several different coalitions at the same time without any resource conflict. Moreover, we prove by mathematical induction that the proposed algorithm will not discard any invalid encoding and can revise any invalid encoding into a valid one. Finally, a contrast experiment is illustrated to demonstrate the proposed algorithm.
© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Distributed intelligent control based on multi-agent systems (MAS) is springing up vigorously for cooperative tasks in many open distributed computing applications. In most cases, these applications are composed of many software agents with certain functional capabilities and usually delegate their tasks to those agents to perform.

When an agent encounters tasks that are difficult to accomplish with its finite resources, it has to interact and cooperate with other agents by forming teams where each team is assigned a task and is called a coalition.

Coalition formation is a fundamental and important form of interaction in MAS. Such coalitions can improve the performance of individual agents and perform tasks more efficiently. Therefore, forming effective coalitions is a major research challenge in the field of MAS and has received a considerable amount of attention. For example, coalition formation has been successfully and widely used in electric transmission system [6], e-business [2], sensor network [3], multi-robot cooperation [18], resource coallocation [14], and some other combinatorial optimization problems [9].

Generally speaking, however, much of the research within this area has assumed non-overlapping coalitions where agents are members of at most one coalition, that is, after a coalition is formed, every agent that has joined that coalition is no longer available to join other coalitions, even if it has great capabilities. Therefore, it may be inevitable that quite a few resources of agents will be wasted in MAS.

In fact, in a number of practical scenarios an agent may have to be involved in executing more than one task simultaneously, and distributing its resources and capabilities to several completely different coalitions. Against this background, this paper is absorbed in concurrent formation of overlapping coalitions. We allow that a coalition may undertake several different tasks and each agent may be a member of more than one coalition at the same time. Obviously, this overlapping property can improve the utilization of agents' resources and increase the efficiency of task execution. To achieve the goal, we mainly focus on the revision algorithm for invalid encodings based on a two-dimensional binary encoding and advance the state of the art in the following ways:

- We address the problem of coalition formation in multi-task environments where task execution is concurrent.

* Corresponding author at: School of Computer and Information, Hefei University of Technology, Tunxi Road No. 193, Hefei 230009, Anhui, China.
E-mail address: zgf@hfut.edu.cn (G. Zhang).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.07.015

- We develop a novel revision algorithm to solve resource conflicts when several different coalitions compete against each other for the same agents with finite resources at the same time.
- Compared to previous work, the proposed revision algorithm will not discard any invalid encoding and can revise any invalid encoding into a valid one.
- The proposed revision algorithm can be applied in any evolutionary computing algorithm with binary encoding, such as genetic algorithm [7], binary differential evolution [8], discrete particle swarm optimization [5,11], and so on.

The remainder of this paper is organized as follows. Section 2 gives a brief description of overlapping coalitions model. In Section 3, we discuss related work. Section 4 introduces a two-dimensional binary encoding for concurrent formation of overlapping coalitions. In Section 5, we present an overview of the only other algorithm available in the literature for revising invalid encodings due to Lin and Hu [10]. Section 6 shows how the improved algorithm can be used to revise invalid encodings. In Section 7, we analyze the revision algorithm by mathematical induction, and in Section 8, we evaluate its performance by contrast experiments. Finally, Section 9 concludes and presents future work.

2. Overlapping coalitions model

Consider a set of n bounded-capabilities agents, $A = \{a_1, a_2, \ldots, a_n\}$, which have to cooperate to execute m tasks, $T = \{t_1, t_2, \ldots, t_m\}$, in environments where task execution is concurrent.

Definition 1. Task: Each $t_k \in T$ has a vector of required r-dimensional capabilities, $\mathbf{D}_k = [d_1^k, d_2^k, \ldots, d_r^k]$, $d_j^k \geq 0$, $k = 1, 2, \ldots, m$, $j = 1, 2, \ldots, r$, $r \in \mathbb{N}$.

Definition 2. Agent: Each $a_i \in A$ has an original vector of real non-negative r-dimensional capabilities, $\mathbf{B}_i = [b_1^i, b_2^i, \ldots, b_r^i]$, $b_j^i \geq 0$, $i = 1, 2, \ldots, n$, where each capability is a property of an agent that quantifies its ability to perform a specific action. Moreover, each $a_i \in A$ has a vector of real workload for every $t_k \in T$, $\mathbf{W}_{ki} = [w_1^{ki}, w_2^{ki}, \ldots, w_r^{ki}]$, $0 \leq w_j^{ki} \leq b_j^i$, which is a real contribution of a_i for executing t_k.

Definition 3. Overlapping coalitions: A coalition C_k, $C_k \subseteq A$ and $C_k \neq \varnothing$, with responsibility for task t_k, is a set of member agents. C_k has a vector of r-dimensional capabilities, $\mathbf{B}_{C_k} = [b_1^{C_k}, b_2^{C_k}, \ldots, b_r^{C_k}]$, which is the sum of the capabilities that the coalition members contribute to this specific coalition. Note that in the case of overlapping coalitions this sum is not the sum of all of the original capabilities of the members, because the agents may be members of more than one coalition, and can contribute part of their capabilities to one coalition and part of them to another. Thus, here $\mathbf{B}_{C_k}$ satisfies that for each $j = 1, 2, \ldots, r$, $b_j^{C_k} = \sum_{a_i \in C_k} w_j^{ki}$. Since $\mathbf{B}_{C_k}$ is the sum of real contribution of every agent in C_k, it is easily known that in the case of overlapping coalitions, $\mathbf{B}_{C_k} = \sum_{a_i \in C_k} \mathbf{W}_k = \mathbf{D}_k$.

Given definitions above, a coalition C_k can perform its task t_k only if the vector of capabilities necessary for its fulfillment $\mathbf{D}_k$ satisfies

$$b_j^{C_k} \geq d_j^k, j = 1, 2, \ldots, r. \tag{1}$$

In addition, the value of coalition C_k with responsibility for t_k is assigned by a characteristic function $V(C_k) \geq 0$ which is just the gain distributed among agents in coalition C_k [21].

$$V(C_k) = \Phi(t_k) - \Theta(C_k) - \Pi(C_k). \tag{2}$$

where $\Phi(t_k)$ is the guerdon paid for finishing t_k and usually is a given constant number; $\Theta(C_k)$ is the total cost of all members' contribution, namely, $\Theta(C_k) = \sum_{a_i \in C_k} \sum_j w_j^{ki}$; $\Pi(C_k)$ is the total cost of communication between members. Note that the communication cost between a_{i_1} and a_{i_2} is $\pi_{i_1 i_2}$, which is a given constant number, satisfying $\pi_{i_1 i_1} = 0, \pi_{i_1 i_2} = \pi_{i_2 i_1}$, if $C_k = \{a_{i_1}, a_{i_2}, \ldots, a_{i_{n-1}}, a_{i_n}\}$, $\Pi(C_k) = (\pi_{i_1 i_2} + \pi_{i_1 i_3} + \cdots + \pi_{i_1 i_{n-1}} + \pi_{i_1 i_n}) + (\pi_{i_2 i_3} + \pi_{i_2 i_4} + \cdots + \pi_{i_2 i_{n-1}} + \pi_{i_2 i_n}) + \cdots + (\pi_{i_{n-1} i_n})$.

Concurrent formation of overlapping coalitions is just generating m coalitions, $C_1, C_2, \ldots, C_m$, simultaneously according to task $t_1, t_2, \ldots, t_m$ under the condition

$$\sum_{a_i \in A} b_j^i \geq \sum_{t_k \in T} d_j^k, j = 1, 2, \ldots, r. \tag{3}$$

The object is to maximize the payoff of the whole system

$$V_{MAS} = \sum_{k=1}^{m} V(C_k). \tag{4}$$

3. Related work

To date, much of the existing research on coalition formation has focused on disjoint coalitions, where it is usually assumed that every agent that has joined a coalition is no longer available to join other coalitions. In this context, many solutions [1,4,12,17,20] have been proposed to find the optimal coalitions in the whole set of possible coalition structures, which is computationally complex due to the size of the set which is exponential in the number of agents. So Sen and Dutta [13] adopt the genetic algorithm and use one-dimensional integral encoding to identify the optimal coalition structure. In addition, Yang and Luo [21] improve on Sen and Dutta's algorithm and design a two-dimensional binary chromosome encoding and corresponding crossover and mutation operators to search the coalition structure space. However, all of them suffer from an important drawback that each agent can exactly take part in only a coalition, producing a big waste of capabilities and limiting the scope of their applications in real-world scenarios. In fact, an agent may be involved in executing more than one task, and distributing its resources among several (not necessarily disjoint) coalitions.

To model this domain, Shehory and Kraus [15,16] firstly introduce the notion of overlapping coalitions in their seminal work on coalition formation for task allocation and develop greedy algorithms for finding a solution to the overlapping coalitions problem that exhibits logarithmic bound. But no bound can be guaranteed from the optimal solution that could have been found by searching all possible coalitions, and their algorithm usually bears with high communication cost, and moreover, they only consider a specific block-world scenario where task execution is serial.

Then, Dang et al. [3] develop a polynomial algorithm for overlapping coalitions to track targets of interest in multi-sensor networks, which can typically generates solutions much closer to the optimal than the theoretical bound. But their deterministic search algorithm depends on a complete information environment and thus is at its wit's end when there is a mass of targets and sensors.

Since the size of the space is exponential in the number of agents, Zhang et al. [22] adopt discrete particle swarm optimization and design a two-dimensional binary encoding to search overlapping coalitions. But their algorithm doesn't do well in solving resource conflicts over the usage of joint resources and produces many invalid encodings, especially when the sum of all agents' capabilities is much closer to the sum of all tasks' required capabilities.

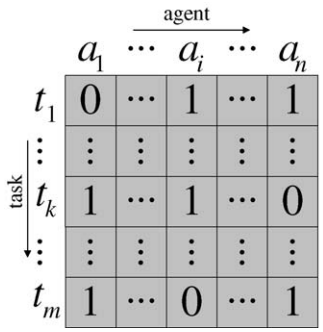

Fig. 1. Two-dimensional binary encoding.

Thus, Lin and Hu [10] propose a revision algorithm for invalid encodings to improve on Zhang et al.'s work. In their algorithm, the residual capabilities of all valid coalitions (which can perform their corresponding tasks) are transferred to the first agent, a_1, and delegate a_1 to join and help other invalid coalitions. But their algorithm also suffers from a number of drawbacks which are directly relevant to the work described in this paper, and an overview of their algorithm is present in Section 5.

To address these shortcomings, our research will do a thorough literature review of existing algorithms, and evaluate them both theoretically and empirically. Based on our findings, we will develop a more efficient algorithm for concurrent formation of overlapping coalitions. Like Lin and Hu's algorithm, ours mainly focuses on how to revise an invalid encoding into a valid one without any resource conflict. Therefore, when it comes to evaluating performance, we compare the proposed algorithm with Lin and Hu's.

4. Two-dimensional binary encoding

Intuitively, parallel formation of overlapping coalitions is just selecting a portion of members from n agents to form m coalitions to execute m tasks, which is similar to a two-dimensional combinatorial optimization problem. While two-dimensional binary encoding not only is simple, easy to comprehend, but also fundamentally fits well with the two-dimensional characteristic of coalition formation problem, which establishes a good foundation for problem solving and provides an extremely broad space for designing algorithms with excellent performances [21]. Therefore, in this paper we devote ourselves to this encoding mode.

Fig. 1 shows a 0–1 binary matrix of $m \times n$, representing a two-dimensional binary encoding. Each row represents a task and each column represents an agent. The element at the intersection of the k th row and the i th column is defined as Δ_{ki}. If $\Delta_{ki} = 1$, agent a_i will take part in coalition C_k with responsibility for task t_k. If $\Delta_{ki} = 0$, agent a_i will not join and cooperate to perform t_k. Since overlapping coalitions allow that an agent may join several different coalitions at the same time, a column may contain more than a bit "1" in an encoding.

5. Lin and Hu's algorithm

After an encoding is generated randomly, we may have to be faced with two tough problems. On the one hand, see Fig. 1, if in the k th row, $\exists j \in \{1, 2, \ldots, r\}1$, $\sum_{i \wedge \Delta_{ki}=1} b_j^i < d_j^k$, that is $b_j^{C_k} < d_j^k$, coalition C_k cannot perform task t_k, here C_k is an infeasible coalition, and thus this encoding is invalid. On the other hand, column i may contain more than a bit "1", that is several coalitions compete against each other for a_i simultaneously, but if a_i does not have enough capabilities and can not satisfy these coalitions' need at the same time, resource conflicts may take place, and thus this encoding is also invalid.

In fact, as long as either of problems shown above takes place, this encoding is just invalid, which consumedly debases the availability of encodings. Therefore, Lin and Hu [10] propose a revision algorithm for invalid encodings. An algorithmic description of their work is listed in Fig. 2.

Their main idea is transferring residual capabilities of all valid coalitions to a_1 and delegating a_1 to help and solve other invalid coalitions. But from a more technical point of view, their algorithm suffers from a number of drawbacks.

In Step1, discarding an encoding by mere situation that row 1 is invalid, is extremely arbitrary and blind, because it is absolutely possible that residual capabilities of other unchecked rows can satisfy C_1's need.

In Step3, a column i contains at most a bit "1", thus a_i can only join at most a coalition, which is too rigorously to make agents join coalitions freely and vigorously. More figuratively speaking, this situation practises egalitarianism, with every agent "eating from the same big pot", and consumedly decrease the efficiency of task execution and the income of the whole system.

In Step5, if $\mathbf{B}_{C_k} \geq \mathbf{D}_k$, C_k is valid already and it is not necessary for a_1 to join C_k, thus "$\Delta_{k1} \leftarrow 1$" should be eliminated. Otherwise, a_1 will bring extra cost to C_k and thus decrease C_k's income. Moreover, discarding an encoding by mere situation that none of invalid coalitions can perform its task after a_1 joins it, is also very blind, because other unchecked valid coalitions with great residual capabilities may be competent for solving those invalid coalitions.

Particularly, a_1 represents a feasible coalition in deed, so it is clear that a valid coalition may have participated in several tasks, but it is not clear that whether every member in this coalition has really taken on tasks or not, and how much workload every member should perform at least for each task. Therefore, their algorithm can not tell the system which agent join which task in deed, and thus can not provide a specific and significant reference for practical control tasks in MAS.

The underlying cause for those drawbacks cited above is that Lin and Hu solve the problem only with a view to agents and coalitions, but do not consider the problem from the encoding itself, and thus their algorithm can not primely carry forward the merits of two-dimensional binary encoding. Therefore, in the improved algorithm, we devote ourselves to research on this encoding model, and if an encoding is invalid, we will make great efforts to revise it into a valid one without any resource conflict, instead of discarding it for any reason.

6. The improved algorithm

Definition 4. Two-value function $check(i)$ denotes whether every column i has been checked or not. If column i has been checked previously, $check(i) = 1$. Otherwise, $check(i) = 0$.

Definition 5. If $\Delta_{ki} = 1$ and $check(i) = 0$, we need to calculate the workload, $\mathbf{L}_{ki} = [l_1^{ki}, l_2^{ki}, \ldots, l_r^{ki}]$, that a_i should perform at least for t_k according to

$$l_j^{ki} \leftarrow d_j^k - \sum_{a_{i^*} \in C_k \wedge i^* \neq i} [(1 - check(i^*)) \times b_j^{i^*} + check(i^*) \times w_j^{ki^*}] \quad (5)$$

$$l_j^{ki} \leftarrow \begin{cases} l_j^{ki}, l_j^{ki} > 0 \\ 0, l_j^{ki} \leq 0 \end{cases} \quad (6)$$

where unchecked columns should contribute $\mathbf{B}_i$ and checked columns should contribute $\mathbf{W}_{ki}$.

- Step1. If $\mathbf{B}_{C_1} < \mathbf{D}_1$, then discard this encoding and end the algorithm; else $\mathbf{B}_1 \leftarrow \mathbf{B}_1 + (\sum_{a_i \in C_1 \wedge i \neq 1} \mathbf{B}_i - \mathbf{D}_1)$.
- Step2. For every $k = 2, 3, \cdots, m$, $\Delta_{k1} \leftarrow 0$.
- Step3. For every $k = 2, 3, \cdots, m$ and every $i = 2, 3, \cdots, n$, if $\Delta_{ki} = 1$ and for every $k' = 1, 2, \cdots, k-1, \Delta_{k'i} = 0$, then $\Delta_{ki} \leftarrow 1$; else $\Delta_{ki} \leftarrow 0$.
- Step4. $k \leftarrow 2$.
- Step5. If $\mathbf{B}_{C_k} \geq \mathbf{D}_k$, $\Delta_{k1} \leftarrow 1$, $\mathbf{B}_1 \leftarrow \mathbf{B}_1 + (\mathbf{B}_{C_k} - \mathbf{D}_k)$; else travel all invalid coalitions from row 2 to row m, find out every invalid coalition which is as important as t_k, if none of invalid coalitions found out above can perform its task after a_1 joins it, then discard this encoding and end the algorithm, else select a row k^* whose coalition has the most value after a_1 joins it, $\Delta_{k^*1} \leftarrow 1$, $\mathbf{B}_1 \leftarrow \mathbf{B}_1 + (\mathbf{B}_{C_{k^*}} - \mathbf{D}_{k^*})$.
- Step6. $k \leftarrow k + 1$. If $k \leq m$, goto Step5; else end the algorithm.

Fig. 2. Lin and Hu's revision algorithm.

If $\mathbf{L}_{ki} = 0$, a_i is dispensable for C_k, namely, $C_k - \{a_i\}$ can still perform t_k, set $\Delta_{ki} \leftarrow 0$ and thus let a_i drop out of C_k to decrease C_k's cost and increase its income. Otherwise, $\mathbf{L}_{ki} > 0$, and here a_i is absolutely necessary for C_k.

Given this, we can now express the improved algorithm as follows:

- Step1. (*Row Checking*) For every $k = 1, 2, \ldots, m$, if $\exists j \in \{1, 2, \ldots, r\}$, $b_j^{C_k} < d_j^k$, select randomly a column i^* in row k, satisfying $\Delta_{ki^*} = 0$ and $b_j^{i^*} > 0$, set $\Delta_{ki^*} \leftarrow 1$, $C_k \leftarrow C_k + \{a_{i^*}\}$, repeat this step until C_k is feasible.
- Step2. (*Initialization*) For every $i = 1, 2, \ldots, n$, set $check(i) \leftarrow 0$. For every $k = 1, 2, \ldots, m$ and for every $i = 1, 2, \ldots, n$, set $\mathbf{W}_{ki} \leftarrow 0$ and $\mathbf{L}_{ki} \leftarrow 0$.
- Step3. (*Column Checking*) If for each $i = 1, 2, \ldots, n$, $check(i) = 1$, end the algorithm. Otherwise, select randomly an unchecked column i.
- Step4. For every $k = 1, 2, \ldots, m$, if $\Delta_{ki} = 1$, calculate each $\mathbf{L}_{ki}$ according to Eqs. (5) and (6), where if $\mathbf{L}_{ki} = 0$, a_i is needless for C_k, set $\Delta_{ki} \leftarrow 0$, $C_k \leftarrow C_k - \{a_i\}$, otherwise, a_i is necessary for C_k.
- Step5. If column i has no any bit "1", set $check(i) \leftarrow 1$, goto Step3.
- Step6. If $\sum_{k \wedge \Delta_{ki}=1} \mathbf{L}_{ki} \leq \mathbf{B}_i$, a_i has great capabilities and can satisfy several coalitions at the same time. Hence for every $k = 1, 2, \ldots, m$, if $\Delta_{ki} = 1$, set $\mathbf{W}_{ki} \leftarrow \mathbf{L}_{ki}$.
- Step7. If $\sum_{k \wedge \Delta_{ki}=1} \mathbf{L}_{ki} > \mathbf{B}_i$, a_i has no enough capabilities and possible resource conflicts take place, then continue according to the following aspects:
  - * Step7.1. Select randomly a row k^* with $\Delta_{k^*i} = 1$ at column i, set $\Delta_{k^*i} \leftarrow 0$, $C_{k^*} \leftarrow C_{k^*} - \{a_i\}$, $\mathbf{L}_{k^*i} \leftarrow 0$, repeat this step until $\sum_{k \wedge \Delta_{ki}=1} \mathbf{L}_{ki} \leq \mathbf{B}_i$.
  - * Step7.2. For the rest each bit "1" with $\Delta_{ki} = 1$ at column i, set $\mathbf{W}_{ki} \leftarrow \mathbf{L}_{ki}$. Update a_i's residual capabilities, set $\mathbf{B}_i \leftarrow \mathbf{B}_i - \sum_{k \wedge \Delta_{ki}=1} \mathbf{W}_{ki}$ and $check(i) \leftarrow 1$.
  - * Step7.3. However, note that C_{k^*} is infeasible after $C_{k^*} \leftarrow C_{k^*} - \{a_i\}$. Therefore, all invalid rows (which are feasible before column checking) should be adjusted again to be valid as follows:
    - * Step7.3.1. Select randomly an invalid row k^*.
    - * Step7.3.2. Firstly, consider whether there are some checked members in C_{k^*} have needed residual capabilities. If members in C_{k^*} has needed capabilities, let them contribute their residual resources to C_{k^*}. Here we first consider agents that have been checked previously in C_{k^*} to decrease the times that bit "0" in row k^* is reset to "1", because the more bits "1" is in row k^*, the more communication cost C_{k^*} will have and the lower income C_{k^*} will get. Moreover, if we neglect those checked agents' residual capabilities, it is possible that even if all bits "0" in row k^* have been reset to "1", but C_{k^*} still can not perform t_{k^*}, especially when the sum of all agents' capabilities is much closer to the sum of all tasks' required capabilities.
    - * Step7.3.3. If C_{k^*} is still invalid, select randomly a column i^* in row k^*, satisfying $\Delta_{k^*i^*} = 0$ and having needed capabilities, set $\Delta_{k^*i^*} \leftarrow 1$, $C_{k^*} \leftarrow C_{k^*} + \{a_{i^*}\}$, repeat this step until C_{k^*} can perform t_{k^*}.
    - * Step7.3.4. If there is no any infeasible row, goto Step3; Otherwise, goto Step7.2.1.

To illustrate how the proposed algorithm works, Fig. 3 shows an example for revising invalid encodings. As shown in the figure, every agent is randomly selected to join coalitions freely. Note that at a column which has resource conflicts, a row chosen stochastically is sacrificed to ensure that other corresponding tasks can be performed, and at the same time, the sacrificed row (or coalition) is compensated for the loss of capabilities by other available agents and can still perform its given task. Moreover, every agent's real contribution to its tasks is always lower or at most equal to its original capabilities, and this can insure any encoding against any resource conflict.

7. Computational complexity

The proposed revision algorithm for resolving possible resource conflicts has the following property:

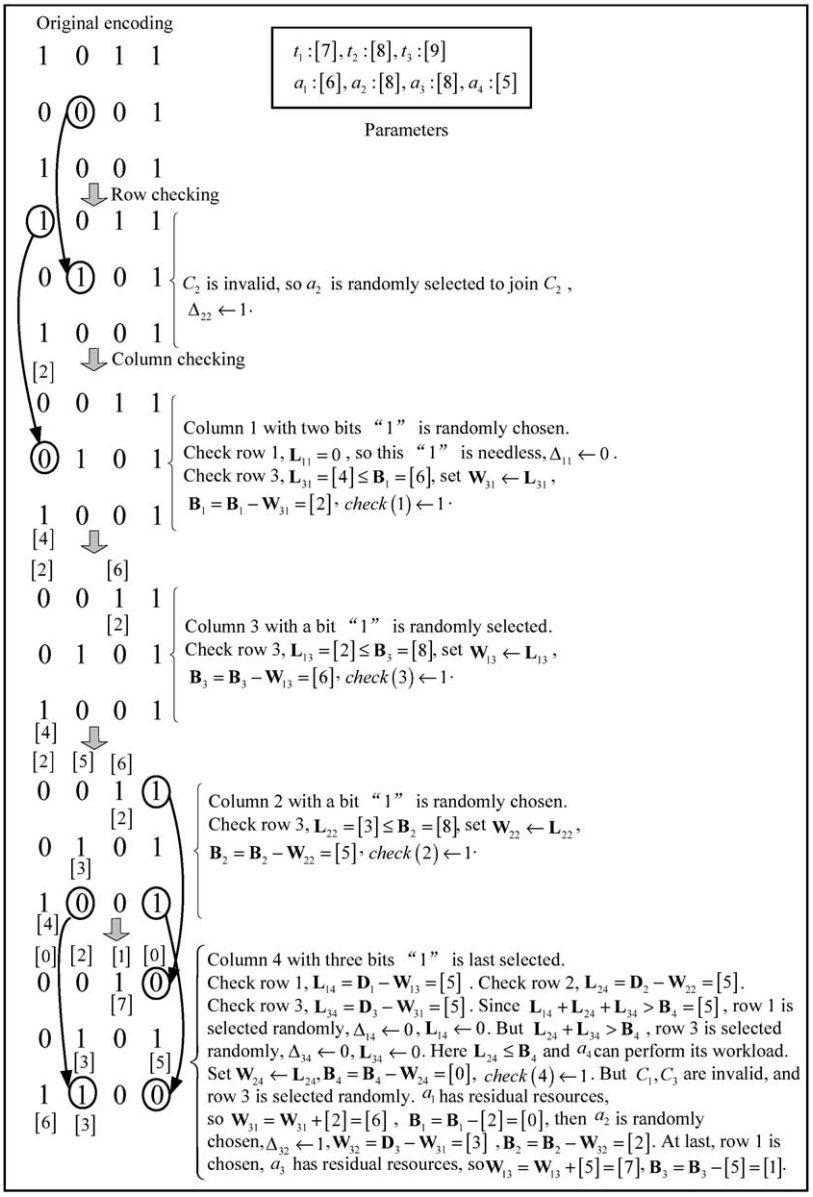

Fig. 3. The revision process of an invalid encoding for the case of 4 agents and 3 tasks.

Theorem 1. *Any given invalid encoding can be revised into a valid one based on the proposed algorithm.*

Proof. It is easily known that all infeasible rows can be adjusted into valid rows on the basis of Eq. (3), so here we just need to prove all columns can be revised to be feasible, namely, there is no any resource conflict at any column based on the proposed algorithm.

We will argue by induction. First, this result is trivial for $n = 1$, it just says that if for every $j = 1, 2, \ldots, r$, $b_j^1 \ge \sum_{t_k \in T} d_j^k$, then any invalid encoding can be revised into a valid one. Since there is only one column in an encoding and $\mathbf{B}_1 \ge \sum_k \mathbf{D}_k$, it is hard to argue with that.

Inductive step. Suppose for some integer $n = q \ge 1$, all q columns can be revised to be feasible. When $n = q + 1$, we know that q columns can be checked and adjusted without any resource conflict from the induction hypothesis. Thus there is only one column left to be checked. Let's say the column is i^*. If column i^* has no any bit "1" or $\sum_{k \wedge \Delta_{ki^*}=1} \mathbf{L}_{ki^*} \le \mathbf{B}_{i^*}$, a_{i^*} has great capabilities and can satisfy several coalitions at the same time, therefore column i^* is feasible. If $\sum_{k \wedge \Delta_{ki^*}=1} \mathbf{L}_{ki^*} > \mathbf{B}_{i^*}$, a_{i^*} has no enough capabilities and possible resource conflicts take place, then some rows will be sacrificed to ensure other tasks can be performed. At the same time, according to constraint (3) each sacrificed coalition can be compensated for the loss of capabilities by other agents (members in the coalition or new available agents), which has been shown at Step7 in Section 6. Finally, column i^* can be adjusted to be valid and all $q + 1$ columns can be revised without any resource conflict. Hence the result follows by induction. □

Now, the efficiency of the improved algorithm can be judged from computations as follows:

- In Step1, we need to check each dimension in each row, and when checking a row, we must travel all agents to calculate its coalition

Table 1
The baseline settings of all experiments.

Number of agents	30
Number of tasks	10
Number of capabilities	2
Communication costs	*RAND*(1, 5)
Population size	25
Number of iterations	500

capability, so the complexity of Step1 is $\mathcal{O}(m \times n \times r)$.
- Similarly, it can be easily seen that the complexity of Step2 is $\mathcal{O}(m \times n)$.
- Specifically, from Step3 to Step7, we need to check each column to solve possible resource conflicts. At worst, a column may have m bits "1" and only hold at most a bit "1" after revising. First, we need to travel $(n-1)$ agents to calculate $\mathbf{L}_{ki}$ for each bit "1". Secondly, when a bit "1" is reset to "0", we may need to travel at most $(n-1)$ agents to ensure the sacrificed coalition is still valid. So the complexity of checking on column is $\mathcal{O}(n \times m \times n \times r) = \mathcal{O}(n^2 \times m \times r)$.

Therefore, the complexity of the improved algorithm for revising invalid encodings is $\mathcal{O}(n^2 \times m \times r)$, which is of polynomial complexity.

8. Performance evaluation

Having calculated the computational complexity, we now present empirical results against Lin and Hu's algorithm. Given different numbers of agents and tasks, we test the performance of two algorithms based on the same simulation platform, namely, discrete particle swarm optimization [11] with different parameter settings. The baseline settings of all experiments are shown in Table 1, where $\sum_{i=1}^{30}\mathbf{B}_i$ is far more than $\sum_{k=1}^{10}\mathbf{D}_k$, but $\sum_{i=1}^{15}\mathbf{B}_i$ is very close to $\sum_{k=1}^{10}\mathbf{D}_k$.

Each algorithm is run for 50 independent trials and empirical results are estimated based on the following metrics:

- Variable number of agents.
- Variable number of tasks.
- Variable number of capabilities.
- Different settings of communication costs.
- Different parameter settings.

8.1. Variable number of agents

Definition 6. Average total income: the average result of the best solutions over 50 runs.

Given different number of agents, Table 2 shows the average total income and Table 3 describes the average number of discarded encodings.

As can be seen, the proposed algorithm obtains significantly more payoff, especially when the number of agents becomes bigger. The reason for this is that the proposed algorithm allows agents to compete against each other freely for given tasks, and thus we can select better teams with lower communication costs. In contrast, when the number of agents increases, the total income of Lin's and Hu's algorithm decreases significantly, despite fewer discarded encodings. This is because the size of every coalition searched by Lin and Hu's algorithm becomes much large (e.g. each coalition has a mass of members, especially when a_1 represents a huge coalition), while a mass of members will result in a larger amount of communication costs and thus decrease a coalition's total income significantly.

Table 2
The average total income for different number of agents.

Number of agents	The proposed algorithm	Lin and Hu's
15	6546	5463
16	6544	5286
17	6546	5149
18	6550	4988
19	6553	4782
20	6549	4585
21	6546	4351
22	6547	4147
23	6555	3992
24	6554	3732
25	6552	3455
26	6549	3239
27	6552	2937
28	6556	2715
29	6554	2453
30	6556	2242

From Table 3 we know that the proposed algorithm does not discard any encoding, namely, it repairs every invalid encoding into a valid one. In contrast, Lin and Hu's algorithm does discard more and more invalid encodings when the number of agents becomes smaller. This is because the total resource of all agents is more and more close to the total required resource of all tasks, and the possible resource conflicts take place more and more often. Therefore, a large amount of invalid encodings, which cannot be repaired by Lin and Hu's algorithm, have to be discarded.

8.2. Variable number of tasks

Given different number of tasks, Table 4 shows the average total income and Table 5 describes the average number of discarded encodings.

As can be seen, at first, the total income of the proposed algorithm and Lin and Hu's algorithm both become more and more when the number of tasks becomes bigger. Moreover, the difference between the proposed algorithm and Lin and Hu's algorithm is not too big, when the number of tasks is no more than 8. However, the total income of Lin and Hu's algorithm starts to decrease significantly when there are 9 or more tasks. This is because the size of the solving coalition for t_9, searched by Lin and Hu's algorithm, becomes much large, resulting in many communication costs and decreasing its total income. Especially, when there are 14 and more tasks, the income of Lin and Hu's algorithm is always negative.

Table 3
The average number of discarded encodings for different number of agents.

Number of agents	The proposed algorithm	Lin and Hu's
15	0	2333
16	0	2073
17	0	1762
18	0	1443
19	0	1406
20	0	1306
21	0	1156
22	0	988
23	0	557
24	0	501
25	0	499
26	0	385
27	0	228
28	0	158
29	0	137
30	0	132

Table 4
The average total income for different number of tasks.

Number of tasks	The proposed algorithm	Lin and Hu's
5	4171	3194
6	4650	3408
7	5132	3289
8	5608	3174
9	6089	2714
10	6555	2227
11	7048	1816
12	7527	1099
13	8017	510
14	8506	−95
15	8977	−991
16	9442	−1845
17	9919	−2580
18	10,403	−3321
19	10,875	−4205
20	11,356	−4810

Table 5
The average number of discarded encodings for different number of tasks.

Number of tasks	The proposed algorithm	Lin and Hu's
5	0	163
6	0	142
7	0	131
8	0	129
9	0	129
10	0	134
11	0	137
12	0	150
13	0	205
14	0	261
15	0	348
16	0	492
17	0	674
18	0	1145
19	0	1533
20	0	1921

As mentioned earlier, when the the number of tasks becomes bigger, the total required resource of all tasks is more and more close to the total resource of all agents, and the possible resource conflicts take place more and more often. Therefore, in Table 5, more and more invalid encodings are discarded by Lin and Hu's algorithm.

8.3. Variable number of capabilities

Given different number of capabilities, Table 6 shows the average total income and Table 7 describes the average number of discarded encodings.

As can be seen, the income of the proposed algorithm is much more than Lin and Hu's when the number of capabilities becomes bigger. This is because when the number of capabilities increases, more and more extra capability costs are produced, and at the same time, the size of coalitions searched by Lin and Hu's algorithm becomes more and more large, which will bring more and more communication costs.

Table 6
The average total income for different number of capabilities.

Number of capabilities	The proposed algorithm	Lin and Hu's
1	6794	3016
2	6554	2131
3	6369	2023
4	6159	1768
5	5977	1574
6	5772	1384
7	5583	1194
8	5370	998
9	5184	817
10	4974	641

Table 7
The average number of discarded encodings for different number of capabilities.

Number of capabilities	The proposed algorithm	Lin and Hu's
1	0	131
2	0	132
3	0	133
4	0	132
5	0	134
6	0	135
7	0	140
8	0	129
9	0	139
10	0	134

As shown in Table 7, the number of discarded encodings does have nothing to do with the number of capabilities. Since it is easily known whether resource conflicts will take place mainly depends on whether the total capabilities of all agents are very close to the total required capabilities of all tasks, and is irrelevant to there are how many dimensional capabilities.

Table 8
The average total income for different communication costs.

Communication costs	The proposed algorithm	Lin and Hu's
0	6605	6605
RAND(1, 3)	6568	3690
RAND(4, 6)	6508	−751
RAND(7, 9)	6455	−5019
RAND(10, 12)	6393	−9006
RAND(13, 15)	6338	−13,883
RAND(16, 18)	6281	−18,203
RAND(19, 21)	6212	−22,045

8.4. Different settings of communication costs

Given different settings of communication costs, Table 8 shows the average total income and Table 9 describes the average number of discarded encodings.

As can be seen, the communication costs have a small impact on the proposed algorithm. In contrast, Lin and Hu's algorithm is very sensitive to the communication costs. When the communication costs are more and more large, the income of Lin and Hu's algorithm becomes more and more little, and even has a negative growth. The reason for this is that the size of each coalition searched by Lin and Hu's algorithm is far more large than the proposed algorithm and thus produces many extra costs.

As shown in Table 9, the number of discarded encodings does have nothing to do with the communication costs.

Table 9
The average number of discarded encodings for different communication costs.

Communication costs	The proposed algorithm	Lin and Hu's
0	0	42
RAND(1, 3)	0	129
RAND(4, 6)	0	133
RAND(7, 9)	0	132
RAND(10, 12)	0	130
RAND(13, 15)	0	126
RAND(16, 18)	0	127
RAND(19, 21)	0	141

Table 10
The average total income for different population size.

Population size	The proposed algorithm	Lin and Hu's
1	6521	1007
5	6545	1590
10	6551	1894
15	6552	2052
20	6556	2090
25	6554	2237
30	6558	2295
35	6555	2237
40	6557	2397
45	6558	2318
50	6561	2399

Table 11
The average number of discarded encodings for different population size.

Population size	The proposed algorithm	Lin and Hu's
1	0	4
5	0	23
10	0	52
15	0	70
20	0	101
25	0	133
30	0	160
35	0	183
40	0	218
45	0	232
50	0	269

8.5. Different parameter settings

Given different population size, Table 10 shows the average total income and Table 11 describes the average number of discarded encodings.

Given different number of iterations, Table 12 shows the average total income and Table 13 describes the average number of discarded encodings.

As shown in the tables, the difference between the proposed algorithm and Lin and Hu's algorithm is very big when the population size or the iteration number is very small. Obviously, the solutions of the proposed algorithm converge quickly in the very early stage of the search algorithm (within less than 300 iterations), and the proposed algorithm can easily get a better solution especially when the population size is between 20 and 30 and the iteration number is between 500 and 1000. This is because the proposed algorithm can revise any invalid encoding into a valid one without any resource conflict, that is any particle at any iteration represents a feasible solution, and this can help the system to explore solution space and quickly find a good solution which is very near the optimum. Therefore, the solution of the proposed algorithm does not improve significantly from iteration 1 and 3000, that is the proposed algorithm is not sensitive to the parameter settings and can easily get a good solution with bad and finite conditions, which establishes a good foundation for practical project applications.

Table 12
The average total income for different number of iterations.

Iteration number	The proposed algorithm	Lin and Hu's
1	6384	−1176
100	6517	1603
200	6537	1844
300	6544	1930
400	6551	2179
500	6557	2206
600	6559	2238
700	6558	2329
800	6563	2356
900	6561	2420
1000	6565	2448
1500	6566	2498
2000	6569	2544
2500	6571	2678
3000	6571	2764

Table 13
The average number of discarded encodings for different number of iterations.

Iteration number	The proposed algorithm	Lin and Hu's
1	0	0
100	0	22
200	0	51
300	0	73
400	0	104
500	0	126
600	0	159
700	0	185
800	0	218
900	0	241
1000	0	278
1500	0	418
2000	0	602
2500	0	766
3000	0	918

In contrast, Lin and Hu's algorithm evolves very slowly for discarding many invalid encodings in Tables 11 and 13. Therefore, it needs more particles or more iteration number to search for a good solution, which results in a large amount of wasting time and memory.

9. Conclusions and future work

In this paper, we have developed a novel algorithm for revising invalid encodings in concurrent formation of overlapping coalitions. We have evaluated the performance of the improved algorithm against Lin and Hu's. This comparison showed that the proposed algorithm is significantly more effective, since the proposed algorithm will not discard any invalid encoding and can revise any invalid encoding into a valid one without any resource conflict.

For future work, we will concentrate on the encoding revision algorithm in ant colony optimization [19] with binary coding.

Acknowledgements

We would like to thank the anonymous referees for their valuable comments as well as helpful suggestions from the Editor-in-Chief which greatly improved the exposition of the paper. This work was supported in part by the National Natural Science Foundation of China under Grant 61004103, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant 20060359004, the Natural Science Foundation of Anhui Province of China under Grant 090412058, the National Engineering Research Center of Special Display Technology under Grant 2008HGXJ0350, and the Specialized Research Fund for Doctoral Scholars, Hefei University of Technology under Grant GDBJ2009-003.

References

[1] B. An, Z. Shen, C. Miao, D. Cheng, Algorithms for transitive dependence-based coalition formation, IEEE Transactions on Industrial Informatics 3 (3) (2007) 234–245.
[2] Y.M. Chen, P.N. Huang, Agent-based bilateral multi-issue negotiation scheme for e-market transactions, Applied Soft Computing 9 (3) (2009) 1057–1067.
[3] V.D. Dang, R.K. Dash, A. Rogers, N.R. Jennings, Overlapping coalition formation for efficient data fusion in multi-sensor networks, in: Proceedings of the

21st National Conference on Artif. Intell., Boston, MA, United States, 2006, pp. 635–640.
[4] V.D. Dang, N.R. Jennings, Generating coalition structures with finite bound from the optimal guarantees, in: Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems, New York, United States, 2004, pp. 564–571.
[5] M. El-Abd, H. Hassan, M. Anis, M.S. Kamela, M. Elmasrya, Discrete cooperative particle swarm optimization for FPGA placement, Applied Soft Computing 10 (1) (2010) 284–295.
[6] G. Erli, K. Takahasi, L. Chen, I. Kurihara, Transmission expansion cost allocation based on cooperative game theory for congestion relief, Electrical Power and Energy Systems 27 (4) (2005) 61–67.
[7] K. Ghoseiri, S.F. Ghannadpour, A hybrid genetic algorithm for multi-depot homogenous locomotive assignment with time windows, Applied Soft Computing 10 (1) (2010) 53–65.
[8] T. Gong, A. Tuson, Differential evolution for binary encoding, in: Proceedings of the 11th Online World Conference on Soft Computing in Industrial Applications, 2006, pp. 251–262.
[9] A.J. Kulkarni, K. Tai, Probability collectives: a multi-agent approach for solving combinatorial optimization problems, Applied Soft Computing 10 (3) (2010) 759–771.
[10] C. Lin, S. Hu, Multi-task overlapping coalition parallel formation algorithm, in: AAMAS 2007, Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Syst., Honolulu, HI, USA, 2007, pp. 1260–1262.
[11] M.A.A. Pedrasa, T.D. Spooner, I.F. MacGill, Scheduling of demand side resources using binary particle swarm optimization, IEEE Transactions on Power Systems 24 (3) (2009) 1173–1181.
[12] T. Rahwan, S.D. Ramchurn, N.R. Jennings, An anytime algorithm for optimal coalition structure generation, Journal of Artificial Intelligence Research 34 (1) (2009) 521–567.
[13] S. Sen, P. Dutta, Searching for optimal coalition structures, in: Proceedings of the 4th International Conference on Multiagent Systems, Boston, USA, 2000, pp. 286–292.
[14] K.T. Seow, K.M. Sim, S.Y.C. Kwek, Coalition formation for resource coallocation using BDI assignment agents, IEEE Transactions on Systems, Man and Cybernetics—Part C: Applications and Reviews. 37 (4) (2007) 682–693.
[15] O. Shehory, S. Kraus, Formation of overlapping coalitions for precedence-ordered task-execution, in: Proceedings of 2nd International Conference on Multi-agent Systems, Kyoto, Japan, 1996, pp. 330–337.
[16] O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial Intelligence 101 (1–2) (1998) 165–200.
[17] Ö. Tombuş, T. Bilgiç, A column generation approach to the coalition formation problem in multi-agent systems, Computers and Operations Research 31 (10) (2004) 1635–1653.
[18] L. Vig, J.A. Adams, Coalition formation: from software agents to robots, Journal of Intelligence and Robotic Systems 50 (1) (2007) 85–118.
[19] L. Xing, Y. Chen, P. Wang, Q. Zhao, J. Xiong, A knowledge-based ant colony optimization for flexible job shop scheduling problems, Applied Soft Computing 10 (3) (2000) 888–896.
[20] S.R. Yang, S.B. Cho, Co-evolutionary learning with strategic coalition for multi-agents, Applied Soft Computing 5 (2) (2005) 193–203.
[21] J. Yang, Z. Luo, Coalition formation mechanism in multi-agent systems based on genetic algorithms, Applied Soft Computing 7 (2) (2007) 561–568.
[22] G. Zhang, J. Jiang, N. Xia, Z. Su, Particle swarms cooperative optimization for coalition generation problem, in: SEAL'06, Proceedings of the 6th International Conference on Simulated Evolution and Learning, LNCS, vol. 4247, 2006, pp. 166–173.