Sequential Coalition Formation for Uncertain Environments

Hosam Hanna
Computer Sciences Department
GREYC - University of Caen
14032 Caen - France
hanna@info.unicaen.fr

Abstract

In several applications, an agent can not execute a task by himself perfectly, thus, agents must form coalitions in order to execute tasks. We address the problem of coalition formation for applications where tasks are executed sequentially: a set of tasks per time. We consider situations where task execution is uncertain. In such applications, an agent is uncertain regarding the execution of subtasks which are allocated to him. In addition, the formation of a coalition at a time has results on the possibilities to form another coalitions in the next time. That is why forming coalition to maximize the system’s real reward, as in classical approaches, is an unrealizable operation. In this paper, we propose a new approach to form coalition sequentially taking into account the uncertain task execution. We formalize the problem by a Markov Decision Process (MDP). We will show that solving the MDP allows to obtain an optimal coalition formation policy.

Keywords: Coalition formation, Group decision making, Markov decision process.

1 Introduction

In several applications, an agent can’t efficiently execute a task by himself, thus agents have to form coalitions in order to execute tasks and to obtain rewards. The coalition formation problem has widely been studied and many approaches were proposed. In game theory, we find some works that treated this problem without taking into account the limited time calculation [1, 3, 7]. In cooperative environments, many algorithms were suggested to answer the question of group formation [15]. In multiagent systems, there are several coalition formation mechanisms that include a protocol as well as strategies to be implemented by agents given the protocol [17, 9, 12]. All these works have common assumptions: resources consumption is perfectly controlled by agents and the formation of a coalition to execute a task is a certain source of reward. In other words, an agent can determine exactly the quantity of resources he will consume to execute any subtask, and the formation of a coalition to execute a task is sufficient to obtain the corresponding reward. In this study, we relax this assumption in order to adapt coalition formation to more real cases, and we investigate the problem of formation coalition in environments where agents have uncertain behaviors.

Several works have investigated the coalition formation problem where coalition value is uncertain or known only to a limited degree of certainty. In [8], author considered the case where agents do not have access to coalition value function, and he proposed a two-agents auction mechanism that allows to determine coalitions of agents that will work together, and to decide how to reward the agents. In [5], authors studied situations where coalition value is known only to a limited degree of certainty. They proposed to use fuzzy quantities instead of real numbers in order to express the coalitions value. A fuzzy Kernel concept has
Consider two tasks execution on the coalition formation process. An example shows the impact of the uncertain task execution to their uncertain behaviors. The following assumption when executing tasks as good as they can in laboratory. A coalition formation planning can not then be provided. In applications as planetary rovers, for example, agents are confronted with an ambiguous environment where they can not control their resources consumption when executing tasks as good as they do in laboratory. A coalition formation planning is important so that agents adapt coalition formation to their uncertain behaviors. The following example shows the impact of the uncertain task execution on the coalition formation process.

<table>
<thead>
<tr>
<th>Agent</th>
<th>$t^1_1$</th>
<th>$t^2_1$</th>
<th>$t^1_2$</th>
<th>$t^2_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>20</td>
<td>30</td>
<td>25</td>
<td>18</td>
</tr>
<tr>
<td>$a_2$</td>
<td>15</td>
<td>17</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>$a_3$</td>
<td>30</td>
<td>15</td>
<td>30</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 1: Agents’ resources consumption for each subtask

been introduced in order to yield stable solutions. Although the complexity of the fuzzy kernel is exponential, it has been shown that this complexity can be reduced to polynomial complexity by placing a cap on the size of coalitions. The uncertainty on coalition value can be due to the unknown execution cost. In fact, when agents reason in term of utility, the net benefits of a coalition is defined as the coalition value minus the execution cost of all the coalition’s members. When an agent of the coalition does not know with certainty the execution costs of the other members, it is uncertain regarding both the coalition’s net benefits and its net benefits. A protocol allowing agents to negotiate and form coalition in such a case has been proposed in [10] and [11]. Another source of uncertainty on coalition value can be the imperfect or deceiving information. A study for this case has been proposed in [4]. In [6], authors proposed a reinforcement learning model to allow agents to refine their beliefs about others’ capabilities.

Although these previous works deal with an important uncertainty issue (uncertain coalition value), they have several restrictive assumptions regarding another possible sources of uncertainty as the uncertain resources consumption (uncertain task execution) that can be due to the uncertain agent’s behavior and to the environment’s dynamism. In addition, they do not take into account the effects of forming a coalition on the future possible formations, a long-term coalition formation planning can not then be provided. In applications as planetary rovers, for example, agents are confronted with an ambiguous environment where they can not control their resources consumption when executing tasks as good as they do in laboratory. A coalition formation planning is important so that agents adapt coalition formation to their uncertain behaviors. The following example shows the impact of the uncertain task execution on the coalition formation process.

Example 1.1 Consider two tasks $T_1$ and $T_2$ that are composed of subtasks as follows: $T_1 = \{t^1_1, t^2_1\}$ and $T_2 = \{t^1_2, t^2_2\}$. Let $a_1$, $a_2$, and $a_3$ be three bounded-resources agents. We assume that the available resources of agents $a_1$, $a_2$, and $a_3$ are respectively 40, 35, and 18. When an agent executes a subtask, a quantity of resources is consumed. The different resources quantities that can be consumed to execute subtasks by the agents are presented on table 1. For example, the execution of subtask $t^1_1$ by agent $a_2$ necessitates a resources quantity equal to 15. Let $\{\langle c_1 = \langle a_1, a_3\rangle, T_1 \rangle, \langle c_2 = \langle a_2, a_1\rangle, T_2 \rangle\}$ be a task allocation provided by some coalition formation protocol without taking into account the uncertain task execution. Now, at the time of the execution, if each agent executes his subtasks, tasks $T_1$ and $T_2$ are then executes and the corresponding rewards are obtained. Unfortunately, when resources consumption is uncertain, maybe agent $a_1$ consumes more than 20 to execute subtask $t^1_1$. Supposing that agent $a_1$ consumed 30 to execute $t^1_1$, $a_1$’s available resources are then 10. Let’s observe the impact of this uncertainty on the execution of $T_2$. In fact, agent $a_2$ consumes 20 to execute $t^1_2$, but $a_1$ can not execute $t^2_2$. Consequently, the reward corresponding to $T_2$’s execution can not be obtained because task $T_2$ is not completely achieved and agents have needlessly consumed resources.

The problem is more complex when resources consumption is uncertain for all the agents. Unfortunately, in such a system, an agent can’t be sure whether he (or another agent) will be able to execute all the subtasks that are allocated to him or he will ignore some of them. So, forming coalitions to maximize the agents’ real reward is a complex (even unrealizable) operation. In fact, a task is considered as non executed if at least one of its subtasks is not executed. That is why, forming a coalition to execute a task is a necessary but not sufficient constraint to obtain a reward, and the agents’ reward must be subjected to the task execution and not only to the coalition formation and task allocation.

In this paper, we take into account these issues and we present a probabilistic model, based on Markov Decision Process (MDP), that provides a coalition formation planning for environments.
where resources consumption is uncertain. We will show that according to each possible resources consumption, agents can decide by an optimal way which coalition they must form.

We begin in Section 2 with a presentation of our framework. In section 3, we sketch our solution approach. We explain how to form coalition via MDP in Section 4.

2 Problem Statement

We consider a situation where a set of $m$ fully-cooperative agents, $A = \{a_1, \ldots, a_m\}$ have to cooperate to execute $N$ sets of tasks by a sequential way: one set per time. We let $T^i = \{T^i_1, \ldots, T^i_n\}$ denote the set of tasks to execute at time $i$. For simplicity, we will let $T$, instead of $T^i_j$, denote a task of set $T_i$. Each task $T$ consists of subtasks: for simplicity, we assume that it is composed by $q$ subtasks such as $T = \{t^1, \ldots, t^q\}$.

Each agent $a_k$ has a bounded quantity of resources $\mathcal{R}_k^i$ that he uses to execute tasks. Agent $a_k$ is able to perform only a subset $E_k(T)$ of the subtasks of a given task $T$. We assume that each task $T \in T_i$ satisfies the condition $T \subseteq \bigcup_{a_k \in A} E_k(T)$, otherwise it is an unrealizable task.

For each subtask $t$ of a given task $T$, we can define the set of agents $\mathcal{AE}(t)$ that are able to perform $t$ as follows: $\mathcal{AE}(t) = \{a_k \in A | t \in E_k(T)\}$. Since an agent can’t execute task $T$ by himself, a coalition must be formed in order to execute this task. Such a coalition can be defined as a $q$-tuple: $(a^1, \ldots, a^q)$ where subtask $t^i$ is allocated to agent $a^i$. We can also say that agent $a^i$ will execute subtask $t^i$. We let $\mathcal{C}(T)$ denote the set of all possible coalitions that can perform task $T$. We call a coalition structure the set of $n$ coalitions that are formed to execute the task of the set $T_i$. We let $\mathcal{cs}(T^i) = \{c_1, \ldots, c_n\}$ denote a coalition structure for set $T_i$, where $c_j$ is the coalition that will execute task $T^i_j$. We assume that, at each time $i$, an agent can be member of only one coalition, formally $\forall c_l, c_j \in \mathcal{cs}(T^i), l \neq j, c_l \cap c_j = \emptyset$. Finally, we let $\mathcal{CS}(T^i)$ denote the set of all possible coalition structures the can perform the tasks of set $T^i$.

Set $T^i$ is considered as realized if and only if all its tasks have been performed. For each realized set, agents obtain a reward. We consider a general situation where the tasks can be executed with different qualities. For example, two agents can take photos for the same object, but the resolution can be different. The reward corresponding to the execution of a task depends then on the coalition that executes the task. Thus, the reward the agents obtain for the execution of a set of tasks depends on the coalition structure formed to execute these tasks. We assume that agents have a function $w(T^i, cs)$ that expresses the reward that can be obtained if the coalitions of $cs$ execute the tasks $T^i_i$. The problem now is, at each time, which coalition structure should be validated in order to maximize the agents’ benefits taking into account the uncertain task execution.

3 Our Approach

The key idea, in our approach, is to view the validation of a coalition structure to execute a set of tasks as a decision to make that provides an expected reward instead of a real gain. What one expects to gain by validating coalition structure $cs$ to execute set $T^i_i$? In fact, when the tasks of $T^i_i$ is allocated to the coalitions of $cs$, the agents expect to obtain two values. The first one is the value $w(T^i, cs)$ which is subjected to the execution of the set $T^i$. The second expected value expresses the gain that can be obtained from future formation and allocation taking into consideration resources quantity consumed to execute the set $T^i_i$. Indeed, when a coalition structure executes tasks, the agents’ available resources is reduced. The chances to execute another tasks can then be reduced. As the resources collection consumed to execute the set $T^i_i$ depends on the coalition structure $cs$ executing $T^i_i$, the gain the agents can obtain from future formation and allocation depends also on coalition structure $cs$. Finally, the expected reward associated to the validation of a coalition to execute a set of tasks is the sum of these two expected values.

Differently from known coalition formation methods that maximize the agents’ real gain, the goal of our agents is defined as follows: for each set of tasks $T^i_i$, form (validate) a coalition structure $cs$ by such a way that it maximizes agents’ long-term expected reward. To realize this objec-
tive, we have to treat the uncertain resources consumption and to formalize the expected reward associated to coalition structure validation.

### 3.1 Uncertain Resources Consumption

In order to deal with the uncertain resources consumption, we assume that the execution of sub-task \( t \in T \) by agent \( a_k \) can consume one quantity of resources of a finite set \( R_k \), of possible quantities of resources. For simplicity, we assume that there are \( p \) resources quantities in the set \( R_k \). Agent \( a_k \) doesn’t know which quantity of resources will be consumed, but he can anticipate it using some probability distribution:

**Definition 3.1** With each agent \( a_k \in A \) is associated an execution probability distribution \( PE_k \) where \( \forall t \in T, \forall r \in R_k, PE_k(r, t) \) represents the probability to consume the resources quantity \( r \) at the time of the execution of subtask \( t \) by agent \( a_k \).

Consider a coalition structure \( cs(T^i) = \{c_1, \ldots, c_n\} \). If coalition \( c_j = \{a^1, \ldots, a^q\} \) executes task \( T_j \), a resources collection such as \( \langle r^1, \ldots, r^q \rangle \) can be consumed, where agent \( a^k \) consumes quantity \( r^k \) to perform subtask \( T^k \).

Generally, we can say that the execution of the set \( T^i \) by the agents of \( cs \) consumes a resources collection such as \( \langle r^1, \ldots, r^{n.q} \rangle \). We let \( H^{cs} \) denote the set of all resources collections that can be consumed by the agents of \( cs \) to execute the set \( T^i \). The probability \( Pr(\langle r^1, \ldots, r^{n.q} \rangle, T^i) \) to consume collection \( \langle r^1, \ldots, r^{n.q} \rangle \in H^{cs} \) at the time of the execution of set \( T^i \) by \( cs \) is then the probability that each agent \( a_k \) consumes the quantity \( r^k \). Using definition (3.1), this probability can be defined as follows:

\[
Pr(\langle r^1, \ldots, r^{n.q} \rangle, T^i) = \prod_{k=1}^{n.q} PE_{a_k}(r^k, T^k) \tag{1}
\]

### 3.2 CS Validation Expected Reward

In our context, a specific agent, “controller”, is charged to form coalitions and to allocate tasks. Controller views the validation of a coalition structure to execute a set of tasks as a decision to make. When such a decision is made, several coalitions are formed, tasks are allocated to these coalitions, and a resources collection will be consumed at the time of execution. As we have shown in Section 3, the decision to validate a coalition structure to execute a set of tasks is associated to an expected reward. In the following, we show how controller can calculate this expected reward.

The controller observes the state of the system as the couple of available resources of all the agents, the set of validated coalition structures and the task allocation. Being in a state \( S \), the decision that consists in validating a coalition structure \( cs \) to execute set \( T^i \) drives the system into a new state \( S^h \) in which the tasks of \( T^i \) have been allocated to the coalitions of \( cs \) and a resources collection \( h \in H^{cs} \) is anticipated to be consumed at the time of task execution. In order to take into account the uncertain task execution, controller must anticipate all the possible resources collections that can be consumed; each possible consumption drives the system into a different state.

If agents of coalition structure \( cs \) have enough resources to execute \( T^i \) (collection \( h \) is less than \( cs \)'s agents available resources), then the system receives in state \( S^h \) an immediate gain \( w(T^i, cs) \) (first expected value), else it receives zero. From state \( S^h \) another decision can be made and another reward can be so obtained (second expected value). We let \( V[S^h] \) denote the gain in state \( S^h \) and we define it as the sum of both last rewards (see Section 4.3 for mathematical definition). Being in state \( S \), the probability to gain \( V[S^h] \), if coalition structure \( cs \) is validated to execute \( T^i \), is expressed by the probability to consume resources collection \( h \) because the system reaches state \( S^h \) if collection \( h \) has been consumed. This probability is defined by equation 1. We can say now that being in state \( S \) the decision to validate the coalition structure \( cs \) to execute the tasks of the set \( T^i \) drives to state \( S^h \) and allows to gain \( V[S^h] \) with probability \( Pr(h, T^i) \), where \( h \in H^{cs} \). The expected reward of this decision can be defined as follows:

\[
E(\text{Validate } cs \text{ to execute } T^i) = \sum_{h \in H^{cs}} Pr(h, T^i) \times V[S^h] \tag{2}
\]

We note that the expected reward associated to a decision made in the state \( S \) depends on the gain
that can be obtained in each state $S^h$, and so on. The question is then: being in a state $S$ and knowing that there are $|CS(T^i)|$ coalition structures capable to execute $T^i$, which decision controller has to make in order to maximize his long-term expected reward?

To answer this question, we formalize the sequential coalition formation problem using a Markov Decision Process (MDP). We will show that the MDP allows to determine an optimal coalition structure validation policy that defines for each system state the coalition structure to validate in order to maximize the system’s long-term expected reward.

4 CS Validation Via MDP

The coalition structure validation can be viewed as a sequential decision process. At each step of this process, the decision to validate a coalition structure to execute a set of tasks has to be made. In the next step, another decision concerning the next set of tasks is made, and so on.

The validation of a coalition structure changes the system’s current state into a new one. As it has been shown above, the probability to transit between the system’s current state and a new state depends only on the system’s current state and on the taken decision. So, this process is a Markovian one [13, 2].

A Markov decision process consists of a set of all system’s states $S$, a set of actions $AC$ and a model of transition [2]. With each state is associated a reward function and with each action is associated an expected reward. In the following, we describe our MDP via: the states, the actions, the transition model and the expected reward.

4.1 States Representation

A state $S$ of the set $S$ represents a situation of coalition structure validation and resources consumption for all the agents. We let $S_i = (B_i, (R^1_i, \ldots, R^m_i))$ the system state at time $i$ where:

- $B_i$ is the set of coalition structures representing the coalition formation until time $i$: $B_i = \{(T^f, cs^f)\mid f = 1, \ldots, i\}$, coalition structure $cs^f$ is formed to execute the tasks of the set $T^f$;

- $R^k_i, k = 1, \ldots, m$ is the available resources of the agent $a_k$ at time $i$.

At time 0 the system is in the initial state $S_0 = (\emptyset, (R^1, \ldots, R^m))$, where $R^k$ is the initial resources of agent $a_k$. At time $N$ (number of sets of tasks), system reaches a final state $S_N$ where there are no more tasks to allocate or no more resources to execute tasks.

4.2 Actions and transition model

With each state $S_{i-1} \in S$ is associated a set of actions $AC(S_{i-1}) \subseteq AC$. An action of $AC(S_{i-1})$ consists in validating a coalition structure $cs \in CS(T^i)$ to execute the tasks of the set $T^i$ and in anticipating the resources collection which can be consumed. We let $Validate(cs, T^i)$ denote such an action. Being in state $S_{i-1} = (B_{i-1}, (R^1_{i-1}, \ldots, R^m_{i-1}))$, the application of action $Validate(cs, T^i)$ drives the system into a new state $S_i$ which can be one of the following states:

$$S_i = (B^h_i, (R^1_i, \ldots, R^m_i)) \quad (3)$$

where:

- $cs = (c_1, \ldots, c_n)$

- $\forall h \in (r^1, \ldots, r^n) \in H^{cs}$

- $B^h_i = B_{i-1} \cup \{(T^i, cs)\}$

- $\forall a_k \in A, a_k \notin cs, R^k_i = R^k_{i-1}$

- $\forall a^l = a_k \in cs$,

$$R^k_i = \begin{cases} R^k_{i-1} - r^l, & \text{if } R^k_{i-1} \geq r^l \\ 0, & \text{if } r^l > R^k_{i-1} \end{cases}$$

In fact, there are $|H^{cs}|$ possible future states because the execution of the tasks of $T^i$ by coalition structure $cs$ can consume one resources collection of the set $H^{cs}$. The case where $r^l > R^k_{i-1}$ corresponds to the situation when agent $a^l = a_k$ tries
to execute \( t^l \in T \) and he consumes all his resources \( R_{i-1}^k \) but \( t^l \) is not completely performed because it necessitates more resources (\( r^l \)). The \( \alpha \)'s available resources is then 0 and task \( T \) can't be considered as a realized task. If \( \alpha \)'s agents have enough resources to execute the tasks of \( T^i \), an immediate gain equal to \( w(T^i, cs) \) will be received in state \( S^h_i \). In the other case (\( \alpha \)'s available resources are not sufficient to completely execute the tasks of \( T^i \)), the immediate gain is equal to 0. We let \( \alpha(S^h_i) \) denote the immediate gain in state \( S^h_i \), thus:

\[
\alpha(S^h_i) = \begin{cases} 
  w(T^i, cs), & \text{if } \forall a^l = a_k \in cs, r^l \leq R_{i-1}^k \\
  0, & \text{otherwise: } \exists d^l = a_k \in cs, r^l > R_{i-1}^k 
\end{cases}
\]  

(4)

Furthermore, the probability of the transition from state \( S_{i-1} \) to a state \( S^h_i \) knowing that the action Validate\((cs, T^i)\) is applied can be expressed by the probability to consume resources collection \( h \) by coalition structure \( cs \), thus \( Pr(S^h_i | S_{i-1}, Validate(cs, T^i)) = Pr(h, T^i) \). It’s important to know that state \( S^h_i \) is inevitably different from the state \( S_{i-1} \). In fact, the set of tasks to allocate in \( S_{i-1} \) was \( T^i \), while in any state \( S^h_i, h \in H^{cs} \) we validate a coalition structure to execute the tasks of \( T^{i+1} \). In other words, being in a state \( S \) at time \( i \), there are no actions that can drive the system to a state \( S' \) which was the system’s state at time \( i' \leq i \). Consequently, the developed MDP doesn’t contain loops, it is a finite horizon MDP [16]. This is a very important property as we will show in the following.

### 4.3 Expected Reward and Optimal Policy

The decision to apply an action depends on the reward that the system expects to obtain by applying this action. We let \( E(Validate(cs, T^i), S_{i-1}) \) the expected reward associated to the action \( Validate(cs, T^i) \) applied in state \( S_{i-1} \). We recall that this expected reward represents what the system, being in state \( S_{i-1} \), expects to gain if coalition structure \( cs \) is formed to execute the tasks of \( T^i \).

A policy \( \pi \) to follow is a mapping from states to actions. For state \( S_{i-1} \in S \), \( \pi(S_{i-1}) \) is an action from \( AC(S_{i-1}) \) to apply. The expected reward of a policy \( \pi(S_{i-1}) = Validate(cs, T^i) \) is \( E(Validate(cs, T^i), S_{i-1}) \). An optimal policy is the policy that maximizes the expected reward at each state. In state \( S_{i-1} \) an optimal policy \( \pi^*(S_{i-1}) \) is then the action whose expected reward is maximal. Formally,

\[
\pi^*(S_{i-1}) = \arg\left( \max_{cs \in CS(T^i)} \left\{ E(Validate(cs, T^i), S_{i-1}) \right\} \right) 
\]  

(5)

Solving equation 5 allows to determine an optimal coalition structure validation (coalition formation) policy at each state \( S_{i-1} \). To do this, the expected reward associated to action \( Validate(cs, T^i) \) has to be defined. Defining this expected reward requires, basing on equation 2, the definition of the reward associated with each state. We define the reward \( V[S_{i-1}] \) associated with a state \( S_{i-1} = (B_{i-1}, (R^1_{i-1}, \ldots, R^m_{i-1})) \) as an immediate gain \( \alpha(S_{i-1}) \) accumulated by the expected reward of the followed policy (reward-to-go). We can formulate \( V[S_{i-1}] \) and \( E(Validate(cs, T^i), S_{i-1}) \) using Bellman’s equations [2], thus:

- for each nonterminal state \( S_{i-1} \):

\[
V[S_{i-1}] = \alpha(S_{i-1}) + E(\pi^*(S_{i-1})) 
\]  

(6)

\[
E(\pi^*(S_{i-1})) = \max_{cs \in CS(T^i)} \left\{ E(Validate(cs, T^i), S_{i-1}) \right\} 
\]  

(7)

\[
E(Validate(cs, T^i), S_{i-1}) = \sum_{h \in H^{cs}} Pr(h, T^i) \times V[S^h_i] 
\]  

(8)

where state \( S^h_i \) corresponds to the consumption of resources collection \( h \).

- for every terminal state \( S_N \):

\[
V[S_N] = \alpha(S_N) 
\]  

(9)

Since the obtained MDP is a finite horizon with no loops, several known algorithms, as Value Iteration and Policy Iteration, solve BELLMAN’s equations in a finite time [14], and an optimal policy is obtained.
4.4 Sequential Coalition Formation

An optimal coalition structure validation can be obtained by solving Bellman’s equations and then applying the optimal policy at each state starting from initial state $S_0$. Here, we distinguish two cases according to the execution model.

The first case corresponds to the execution model where the sets of tasks must be sequentially executed immediately after each coalition structure validation. In this case, a coalition structure to execute the tasks of $T^{i+1}$ is validated at the end of $T^i$’s execution. Let $\pi^*(S_{i-1}) = \text{Validate} (cs, T^i)$ be the optimal policy to apply in the state $S_{i-1}$. The application of this policy means that the coalition structure $cs$ must be validated to execute the tasks of $T^i$. Assuming that resources collection $h$ has been consumed by $cs$ to execute the tasks of $T^i$, system then reaches the state $S_i = S_i^h$ defined by equation 3. From this new state $S_i$, controller applies the calculated optimal policy $\pi^*(S_i)$, and so on.

The second case corresponds to the execution model where controller validates all the possible coalition structures before agents start the execution. In this case, after each coalition structure validation, controller has to anticipate the state the system will reach when executing the tasks. Let $\pi^*(S_{i-1}) = \text{Validate} (cs, T^i)$ be the optimal policy to apply in the state $S_{i-1}$. By applying this optimal policy, coalition structure $cs$ is validated to execute the tasks of $T^i$. The state $S_i$ the system will reach when $cs$ executes $T^i$ can be any state $S_i^h, h \in H^{cs}$. The state that the system has big chances to reach is the state corresponding to the resources collection that can be consumed with a probability maximal. Formally, the state $S_i^h$ the system has a big probability to reach when $cs$ executes the tasks of $T^i$ is the state corresponding to the consumption of the resources collection $h$ that satisfies: $Pr(h, T^i) = \max_{h \in H^{cs}} \{Pr(h, T^i)\}$.

Controller considers the state $S_i = S_i^h$ as the system’s new current state. From this new state, controller applies the calculated optimal policy $\pi^*(S_i)$, and so on until reaching a terminal state $S_N = (B_N,ush, R^N, N)$. Finally, the set $B_N$ contains the formed coalitions and validated coalition structures.

5 Discussion and Conclusion

Coalition formation is an important approach in several multiagent and robotic applications. This problem is very difficult when the environment is characterized by uncertain agents’ behaviors. In fact, the uncertain behaviors can have results on the coalition value and thus on the agents’ reward.

Several studies have been proposed to deal with uncertainty issue. The proposed solutions for coalition formation problem with uncertain coalition value do not take into account the uncertain task execution and the impact of the formation of a coalition on the gain that can be obtained from next formations.

In this paper, we addressed the problem of sequential coalition formation in environments where the resources consumption is uncertain. We considered a general case: big number of tasks and big number of agents. The tasks are represented by sets of tasks where agents must execute one set per time. We showed that in such an environment, forming a coalition to execute a task have impacts on the possibility to form another coalitions. Thus, this issue must be taken into account at each time agents decide to validate a coalition structure. We introduced the notion of expected reward that represents what agents expect to gain by validating a coalition structure (forming coalitions to execute a set of tasks). The expected reward is defined as the sum of (1) what agents immediately gain if the coalition structure executes the set of tasks and (2) what they expects to gain by future formation.

Our idea is to view the validation of a coalition structure as a decision to make that provides, due to the uncertain task execution, an expected reward. Agents’ aim is then to validate coalition structures by a way that it maximizes their long-term expected reward instead of real reward.

The coalition structure validation problem has been formalized by a Markov decision process. The fact that the obtained MDP is a finite horizon guarantees its resolution in a finite time. After solving the MDP, the controller can optimally decide, for each set of tasks, which coalition structure must be validated.

The proposed model allows agents to form coali-
tion for two large classes of applications. The first class includes applications where the validated coalition structure must immediately execute its allocated tasks, while in the second class the execution step starts after validating coalition structures for all the sets of tasks.

In future works, we will extend our model to allow agents to make decentralized decisions. In this case, the communication between agents can replace the complete observability of the system state by the controller agent. Communication Protocols with low cost must be developed.

Furthermore, in systems with self-interested agents, coalition formation mechanism must guarantees some stability. Thus, agents must validate, by a decentralized way, a coalition structure that maximizes the long-term individual expected reward.

References


