A Multiagent Task Associated MDP (MTAMDP) Approach to Resource Allocation

Pierrick Plamondon, Brahim Chaib-draa
DAMAS Laboratory
Laval University
{plamon; chaib}@damas.ift.ulaval.ca

Abder Rezak Benaskeur
Decision Support Systems Section
Defence R&D Canada — Valcartier
Abderrezak.Benaskeur@drdc-rddc.gc.ca

ABSTRACT

We are interested in contributing to solving effectively the a specific type of real-time stochastic resource allocation problem, which is known as NP-Hard, of which the main distinction is the high number of possible interacting actions to execute in a group of tasks. To address this complex resource management problem, we propose an adaptation of the Multiagent Markov Decision Process (MMDP) model which centralizes the computation of interacting resources. This adaptation is called Multiagent Task Associated Markov Decision Process (MTAMDP) and produces a near-optimal solution policy in a much lower time than a standard MMDP approach. In a MTAMDP, a planning agent computes a policy for each resource, and are coordinated by a *central* agent. MTAMDPs enables to practically solve our NP-Hard problem.

1. INTRODUCTION

We are interested in complex stochastic resource allocation problems with hard real-time constraints. In general, Resource allocation problem are known to be NP-Hard [18]. In such problems, a planning process suggests the action (i.e. resources to allocate) to undertake for accomplishing certain tasks, according to the perfectly observable, state of the environment. In our case, the number of possible actions in a state is the combination of each individual possible resource assignment to the current tasks. Thus, the action space is significantly large. The very high number of states and actions in this type of problem coupled with the time constraint makes it very complex, and here we propose a reduction in the computational leverage associated to the action space.

A straightforward approach to solving this problem is to formulate both the resource allocation process and the actual operation of the agents as a large multiagent MDP [4] on the joint state and action spaces of all agents. However, this method suffers from an exponential increase in the size of the state space, and thus very quickly becomes infeasible for teams of reasonable size. A common way of addressing this problem of large state spaces in MDPs is based on problem decomposition ([10], [15]), where a global MDP is decomposed into several independent or looselycoupled sub-MDPs. These sub-MDPs are usually solved independently and the resulting policies are then combined to yield a (perhaps suboptimal) solution to the original global MDP.

In our approach, we decompose the problem such as a planning agent manages each specific resource. The separate policies produced by the agents, if not coordinated, are suboptimal for two reasons. Firstly, some resources may have positive and negative interactions as the expectation of realizing a certain task t_1 by resource res_1 is changed when allocating another resource res_2 simultaneously on task t_1. Secondly, the resources have to be distributed efficiently between the tasks to accomplish. Thus, the planning agents are coordinated together during the planning process through a *central* agent, and produce a near-optimal policy. The planning agents generate a policy to allocate their resources using a Markov Decision Process (MDP) [2], which is very convenient to model a stochastic problem. We call our method "Multiagent Task Associated Markov Decision process" (MTAMDP) because the planning agents are related to each other only because they have the same tasks to accomplish. The result obtained using MTAMDP are near-optimal, while the convergence time is very small compared to a standard Multiagent Markov Decision Process (MMDP) [4] approach.

2. RELATED WORK

In general, there are two known approaches for allocating resources to tasks [8]. Firstly, the episodic approach which uses all the available resources simultaneously. Secondly, the sequential version, as we use it in this paper, which extends the episodic version by having a number of stages in the scenario. In this last context, the planning process is allowed to observe the outcomes of all engagements of the previous stage before assigning and committing resources for the present stage. This approach uses less resources and has a higher expectation to achieve the tasks than the episodic approach.

Resource allocation problem are known to be NP-Hard [18]. Since resources are usually constrained, the allocation of resources to one task restricts the options available for other tasks. The complexity of this problem is exponential with the number of resources and tasks, and many approximations and heuristics have been proposed ([10], [16], [1]). An effective approach, as considered in the current paper, is to plan for the resources separately as proposed by [16]. The authors do not consider positive and negative interactions between resources. Their approach is consequently not very suitable to the type of problem we are interested, since in many real applications there are positive and negative interactions between resources. In their specific approach, a policy is formulated for each resource and a greedy global policy is produced by considering each resource in turn, producing an approximate policy. Also,

their coordination rules are sometime very specific to the problem's characteristics. In our paper, we describe a different and more general approach, where each resource is coordinated by a *central* agent, instead of sequentially.

Salles, Mouaddib, and Chaib-draa [14] proposed also an approach to plan for each resource separately. In this approach each resource is modelled as a Progressive Reasoning Units (PRUs) [11]. A PRU hierarchies the task to accomplish in a degree of importance, such as the most important task is considered first by the PRU. The policy of the first PRU is sent to the next PRU. This next PRU determines its policy upon it and send its policy to the next PRU, and so on. The resulting global policy is sub-optimal. Indeed, this algorithm may also be viewed as greedy since each resource is considered in turn. In our paper, the planning agent which plans for each resource are coordinated with a *central* agent, instead of sequentially, with a minimal complexity increase. Furthermore this approach cannot consider positive and negative interaction between resources.

Since we have local and global resource constraints on the amount of resources that be used, our problem can be viewed as a constrained Markov Decision Process ([3], [7], [17]). In this context, dynamic programming ([3], [7]) or linear programming [17] may be used to obtain a policy. Much work has been done in this field, but none of it diminishes the action space and still provide a near-optimal policy as we do.

Another interesting approach is by Dolgov and Durfee [6]. They propose a mixed integer linear programming formulation of a stochastic resource allocation problem. The planning time is greatly reduced compared to a large Multiagent Markov Decision Process (MMDP) [4] on the joint state and action spaces of all agents. However, their approach supposes a static allocation of resources, while we are interested in a dynamic allocation to consider contingencies. We now model our problem in more detail.

3. PROBLEM FORMULATION

An abstract resource allocation problem is described in Figure 1. In this example, we have four tasks (t_1, t_2, t_3, and t_4) and three types of resources (res_1, res_2, and res_3) each type of resource is constrained by the amount that may be used at a given time (local constraint), and in total (global constraint). The Figure shows the resource allocation to the tasks, in the current state. In this problem, a state represents a conjunction of the particular state of each task, and the available resources. Indeed, when the state of the tasks changes, or when the number of available resources change, then the resource allocation usually changes also. We now describe how we model this type of problem.

3.1 Markov Decision Processes MDPs

A Markov Decision Process (MDP) framework is used to model the stochastic resource allocation problem. MDPs has been widely adopted by today's researchers to model a stochastic process. This is due to the fact that MDPs provide a well-studied and simple, yet a very expressive model of the world. In general, an MDP is given by a state space S, actions $A(s)$ applicable in each state s, transition probabilities $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$, and state rewards $r(s)$.

In a classical MDP, the state s' that results from an action a is not predictable but is perfectly observable, providing

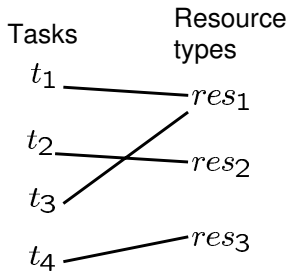

Figure 1: Resource Allocation Problem.

feedback for the selection of the next action a'. A solution of an MDP is a policy π mapping states s into actions $a \in A(s)$. In our case, an optimal policy is one that maximizes the expected total reward to accomplish all tasks. Bellman's *principle of optimality* [2] forms the basis of the stochastic dynamic programming algorithms used to solve MDPs. In particular, the optimal value of a state is the immediate reward for that state added to the expected value of the next state transition probability, assuming that a planning agent chooses the optimal action. That is, the value of a state $V(s)$ is given by:

$$V(s) \leftarrow R(s) + \max_{a \in A(s)} \sum_{s' \in S} P_a(s'|s)V(s') \quad (1)$$

Furthermore, one may compute the Q-Values $Q(a, s)$ of each state action pair using the following equation:

$$Q(a, s) \leftarrow R(s) + \sum_{s' \in S} P_a(s'|s)V(s') \quad (2)$$

where the optimal value of a state is:

$$V(s) \leftarrow \max_{a \in A(s)} Q(a, s) \quad (3)$$

Using Bellman's principle of optimality avoids enumerating all possible solutions, and is sometimes called *pruning by dominance*. Furthermore, it is known that all algorithms to solve an MDP are iterative since a straightforward approach would be too complex. In particular, the standard dynamic programming algorithms to solve MDPs are *value iteration* [2] and *policy iteration* [9]. However, these classical algorithms suffer from the so-called *curse of dimensionality*: the number of states grows exponentially with the number of variables that characterize the domain. Furthermore, the number of actions grows exponentially with the number of available resources. Consequently, a polynomial time algorithm can be prohibitive for a real-time application as in the case for the problem of stochastic resource allocation, addressed in this paper . Furthermore, since the number of possible initial situations is very large, one may not compute a policy a priori. Multi-Agent Markov Decision Processes (MMDP) which extends MDP to multiagent environments, is now introduced.

While computing the solution of an MDP for a resource allocation problem to accomplish a group of task, we have to keep the respective value of each task. This is not a task decomposition in any sense. At each iteration for computing $V(s)$, it is done by computing $V_t(s)$ for each task t, according to the possible actions $A(s)$. $V(s)$ is obtained with the action which maximizes the sum of the value of all tasks.

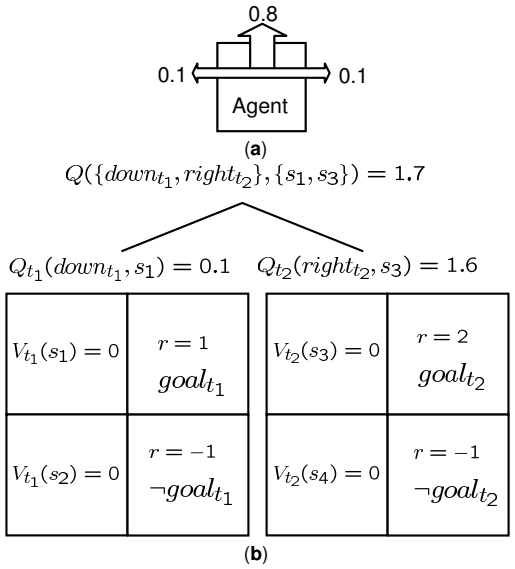

Figure 2: Computation of the value of a state.

Thus, when we will employ $V_t(s)$, or $Q_t(a, s)$ in this text, this is not a decomposition, but it simply means that we consider only the part of the value (or Q-value) related to task t. Figure 2 exemplifies the computation of the Q-value of a state in a problem where two robot agents have the task to reach the *goal* state, as shown on part (a) of the Figure. Each robot have four possible actions $\{left, right, up, down\}$ with the transition probabilities of 0.8 to go in the intended distance and 0.1 to each side, as shown on part (b) of the Figure. If there exist no constraint between the actions of the two robots, we can solve these two independent problems, and the solution will be optimal. On the other hand, if we impose constraint on the actions that may be used, like in a resource allocation problem, an independent optimal solution for each task, will not generate a global optimal solution. For instance, if we impose that the two tasks cannot execute the same action in a same step, some coordination is needed to plan a global solution. This coordination may be achieve by transforming this problem as a global problem where two tasks must be achieved. A possible action in state $s = \{s_1, s_3\}$ may be $a = \{down_{t_1}, right_{t_2}\}$. In this case the *task*-Q-value is 0.1 for task t_1, and 1.6 for task t_2. The global Q-value which is the sum of the specific Q-value of each task is 1.7 in this case. Thus, in a resource allocation we solve the problem for each task individually to reach a global optimal solution, however the action space considers all tasks, and is restricted by local and global resource constraints.

3.2 Multi-Agent Markov Decision Processes (MMDP)

Since Resource allocation problem are known to be NP-Hard [18], one may decompose our previous problem into multiple planning agents. To do so, Multi-Agent Markov Decision Processes (MMDP) [4] may be a very suitable modelling framework. In an MMDP the individual actions of many planning agents interact so that the effect of one agent's actions may depend on the actions taken by others. Furthermore, an MMDP takes the agents to be acting on behalf of some individual; therefore, each has the same utility or reward function R. Indeed, an MMDP is like an MDP, except that P now refers to the probabilities of joint actions.

For a standard MMDP approach, the number of actions to consider in a state is:

$$\prod_{m=1}^{nbAgents} nbChoices_m \tag{4}$$

where $nbChoices_m$ is the number of possible resource allocation for agent m, and $nbAgents$ is the number of agents. In this case, the value of all action combinations of each resource have to be computed. For example, if in a specific state, we have three available resource types, with ten possible actions for each resource type, there are $10^3 = 1000$ Q-values to compute for this state. On the other hand, if an agent plans for each resource type, as with the MTAMDP approach presented in this paper, the number of actions to consider in a state is:

$$\sum_{m=1}^{nbAgents} nbChoices_m \tag{5}$$

Here, the number of actions is the sum of the possible actions of each resource. The number of Q-values to compute in a state, if we have three available resource types, with each ten possible actions, is $10 \times 3 = 30$. The value of $nbChoices_m$ is still, however, exponential for each planning agent, as the number of task augment. Indeed, a resource allocation problem is still an NP-Complete problem, even when we have one resource type [18]. A *central* agent computes the Q-value of all action combinations using the new efficient approach that will be detailed in the subsequent sections. The next section describes in a more formal way MTAMDPs, which significantly diminishes the action space of a resource allocation problem.

3.3 Coordinating a Multiagent Task Associated Multiagent Process (MTAMDP)

An abstract schematization of the approach proposed by Plamondon et al. [13], which extends MMDP, to solve a resource allocation problem is described in Figure 1. This in an extension of Figure 3 where each planning agent (m_1, m_2, and m_3) manages one type of resource to accomplish the tasks. The dotted line in the Figure represent agent m_1 which manages resource type res_1 to accomplish all tasks. This way, each agent can compute a local policy (π_{m_1}, π_{m_2}, π_{m_3}). As a matter of fact, there are two reasons why a local value function of a planning agent needs to be adjusted, according to an action from another agent. First of all, the resources should be divided between the tasks in the case of possible simultaneous actions. We explain this aspect with an example. Suppose there are two identical tasks (t_1 and t_2) in the environment, which are in a particular state. Furthermore, there are two planning agents (m and m') each of which manages one unit of resource (res_1 and res_2). We suppose that using each resource is 50% likely to achieve any task (i.e. 50% likely that the task is realized, and 50% that the task is not realized). Now, suppose that both planning agents assign their respective resource to task t_1. In these conditions, task t_1 is 75% likely to be achieved[1], while

[1] $0.5 + ((1 - 0.5) \times 0.5) = 0.75$

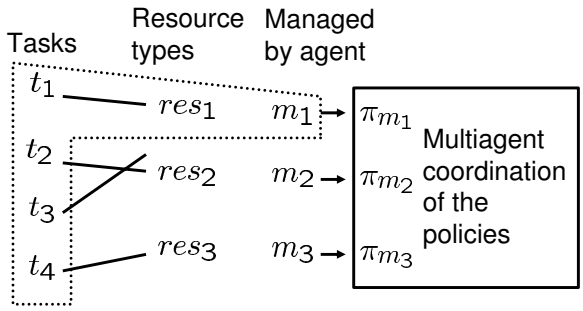

Figure 3: A Multiagent task associated resource allocation problem.

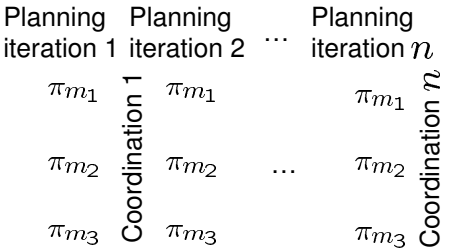

Figure 4: The iterative coordination process.

task t_2 still has 0% of being achieved. As a consequence, it remains $0.25 + 1 = 1.25$ task in the environment. On the other hand, if both planning agents would have coordinated and used their resource on a different task, both task would have been achieved at 50%. In this case, only $0.5 + 0.5 = 1$ task would remains in the environment. This simple example shows why it is better for the planning agents to divide their resources between the tasks in the case of possible simultaneous actions. The second reason why the planning agents should coordinate is to benefit from positives interactions between resources and to alleviate the negative ones. Indeed, the expectation of realizing a certain task t_1 by a unit of resource res_1 could be changed positively or negatively when allocating another unit of resource res_2 simultaneously on task t_1. The general idea we propose in this paper to solve efficiently our resource allocation problem is to coordinate the different agents at each iteration of the planning algorithm. Indeed, all existing algorithms to solve an MDP are iterative, thus our approach should be pretty extensible. Figure 4 describes this process. For example, if n iterations are needed for each planning agent to converge, then n coordination activities are made.

A Multiagent Task Associated Markov Decision Process (MTAMDP) is defined as a tuple $\langle Ag, T, S, A, P, W, R, Res, L_{Res}, G_{Res}\rangle$, where:

- $Res = \{res\}$ is a finite set of resource types available for a global planning process. The planning process has Res resource type. Each resource type has a local resource constraint L_{res} on the amount that may be used on a single step, and a global resource constraint G_{res} on the amount that may be used in total.
- $Ag = \{m\}$ is a finite set of agents. In an MTAMDP, a planning agent manages one or many resources which are used to accomplish its tasks. In this paper, we consider a planning agent for each resource, and the $mNoop$ planning agent which considers only the $noop$ (no operation) action. We have to consider the expected value of the $noop$ action since it may achieve tasks.
- $T = \{t\}$ is a finite set of tasks to be accomplished by the planning agents.
- $S = \{s^m\}$ is a finite set of states available for each planning agent. A state $s^m \in S$, represents a conjunction of the particular state s_t^m, which is the characteristics of each task t in the environment, and the available resources for the planning agent m. Also, S contains a non empty set $G \subseteq S$ of goal states.
- $A = \{a^m\}$ is a finite set of actions available for each planning agent. The actions $A^m(s^m)$ applicable in a state is the combination of all resource assignments that a planning agent m can execute, according to the state s^m. Thus, a^m is simply an allocation of resources to the current tasks, and a_t^m is the resource allocation to task t. The possible actions are limited by L_{res} and G_{res}.
- Transition probabilities $P_a^m(s'^m|s^m)$ for $s^m \in S$ and $a^m \in A^m(s^m)$.
- $W = [w_t]$ is the relative weight of each task, as described in a previous paper [12].
- State rewards $R = [r_s] : \sum_{t=1}^{nbTasks} r_{s_t} \rightarrow \Re \times w_t$ related to the problem. The rewards are not related to any specific planning agent, since they are only associated to the tasks, and not the resources.

The solution of an MTAMDP is a policy π^m for each planning agent in the environment. In particular, $\pi_t^m(s_t^m)$ is the action (i.e. resources to allocate) that should be executed on task t by agent m, considering the specific state s_t^m. As in reinforcement learning, we use the notion of Q-value for a planning agent m in our MTAMDP approach:

$$Q^m(a^m, s^m) \leftarrow R(s^m) + \sum_{s'^m \in S^m} P_{a^m}(s'^m|s^m) V^m(s'^m) \quad (6)$$

, where $Q_t^m(a_t^m, s^m)$ is the part of the Q-value related to task t. We call this part task-Q-value. Each Q-value is subjected to the local resource constraints for each state task s_t of a global state s

$$\sum_{t=1}^{nbTasks} res(\pi_t^m(s^m)) \leq L_{res} \text{ for all } s^m \in S^m \quad (7)$$

where $res(\pi_t^m(s_t^m))$ is the resource used by the policy π^m in state s^m, considering only task t, by agent m. The global constraint is defined according to all system trajectories. A system trajectory is a possible sequence of state-action, until a goal state is reached. For example, we have a state s^m, which may transit to s'^m, or to s''^m, according to action a^m. The two possible system trajectories are $\langle (s^m, a^m), (s'^m)\rangle$ and $\langle (s^m, a^m), (s''^m)\rangle$. The global resource constraint is:

for all system trajectories tra

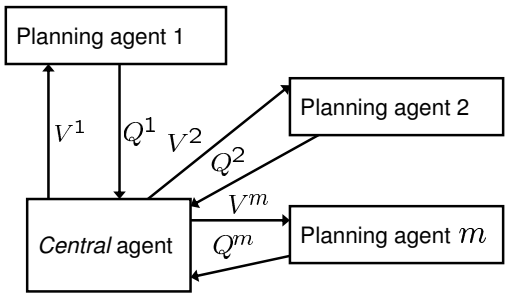

Figure 5: The coordination process.

$$\sum_{s^m \in S^m_{tra}} \sum_{t=1}^{nbTasks} res(\pi^m_t(s^m_t)) \leq G_{res} \quad (8)$$

where the optimal value of a state for a planning agent m is:

$$\sum_{m=1}^{nbAgents} V^m(s^m) \leftarrow \max_{a \in A(s^m)} (Q^m(a^m, s^m)|Q(a, s)) \quad (9)$$

As we can see, the value determined by a planning agent m is the one that maximizes the Q-value of global action a by all planning agents. To do that in an efficient way, the planning agents have to coordinate with each other to produce the best global value function, or policy. In this paper, such a coordination is made through a *central* agent. Figure 5 describes the coordination process between the different planning agents and the *central* agent. At each iteration of an MDP algorithm, for example, value iteration, the planning agents send their Q-values to the *central* agent. With these Q-values, the *central* agent computes the global value of all action combinations. We will describe in the following sections how the central agent calculates the value of a global action, with a set of Q-values in hand. Afterwards, once the *central* agent knows the maximum value of a state, it assigns the value of each agent to its respective contribution (or to their adjusted Q-value as will be defined in the next section). The Algorithm 4 (MTAMDP-VALUE-ITERATION) gives a more formal description of this approach. To help in the reading of the algorithm, the following functions need to be defined first: ADJUST-I(*action a*), ADJUST-SA(*action a*), and GLOBAL-VALUE().

3.4 Adjusting the Q-value of a planning agent

The *central* agent permits to reach a near global optimum for the planning process by coordinating the different planning agents. The two reasons why the respective policies of the planning agents should be coordinated leads to the definition of two near-optimal subfunctions — ADJUST-I(*action a*) (interactions) and ADJUST-SA(*action a*) (simultaneous actions). The ADJUST-I(*action a*) function is defined as follow:

DEFINITION 3.1. ADJUST-I*(action a): This function adjusts the task-Q-values of each planning agent m, considering a global action a, in a global state s. The adjustment is made according to the interactions between the actions of each other planning agent m'.*

Before detailing the calculus to obtain the adjusted Q-value according to the interactions with another action, we introduce $a^m_{Inter_t}(a^m_t, s|a'^{m'}_t) \in a^m_t$ as the part of action a^m_t in interaction with $a'^{m'}_t$. In this context, we define:

DEFINITION 3.2. *$Inter_{a^m_t}(a^m_{Inter_t}(a^m_t, s|a'^{m'}_t), s|a'^{m'}_t)$ is the modified efficiency of action $a^m_{Inter_t}(a^m_t, s|a'^{m'}_t)$ in state s knowing that action $a'^{m'}_t$ is also executed.*

For example, if $Inter_{a^m_t}(a^m_{Inter_t}(a^m_t, s|a'^{m'}_t), s|a'^{m'}_t) = 0.7$, it means that $a^m_{Inter_t}(a^m_t, s|a'^{m'}_t)$, knowing that $a'^{m'}_t$ is also used, has its efficiency at 70% of its regular one. In addition, both the ADJUST-I(*action a*) and ADJUST-SA(*action a*) functions need an upper bound on the value that a state may take in a specific iteration. The definition of this term has a great effect on the quality of the approximation that we obtain for our approach. In particular, we define:

DEFINITION 3.3. *$UpBound^{a^m_t}_s$ represents the maximum value that agent m may expect to obtain in state s, when executing action a^m_t. All upper bounds are specified at each iteration of the planning algorithm.*

The heuristic we used to determine the upper bound for a state s, and action a^m_t by agent m, is the highest value of a possible state transition. We consider as a possible state transition, a state for which $P_{a^m_t}(s'|s) > 0$. This way, the upper bound overestimates the possible value of a state since it is very improbable that an action would guarantee reaching the upper bound. This upper bound provides an approximation of sufficient quality to address the problem at hand. Better approximations remain possible. We now describe the Algorithm 1 to obtain the value of ADJUST-I(*action a*) for a specific global a, in a state s. In the algorithm the *noop* term signifies the no-operation action.

Algorithm 1 Considering interactions.

```
1: Function ADJUST-I(action a)
2: for all m ∈ Ag and  t ∈ T do
3:    if a^m_t ≠ noop then
4:       null_{a^m_t} ← Q^m_t(noop_{a^m_t}, s(||res^m_t − res^m_t(a^m)||))
5:       delta_{a^m_t} ← Q^m_t(a^m_t, s) − null_{a^m_t}
6:       for all m' ∈ Ag do
7:          if a^{m'}_t ≠ noop then
8:             inter ← Inter_{a^m_t}(a^m_{Inter_t}(a^m_t, s|a'^{m'}_t), s|a'^{m'}_t)
9:             if inter ⩽ 1 then
10:               delta_{a^m_t} ← delta_{a^m_t} × inter
11:            else
12:               gain ← delta_{a^m_t} / (UpBound^{a^m_t}_s − null_{a^m_t})
13:               nGain ← 1 − ((1−gain)/inter)
14:               delta_{a^m_t} ← delta_{a^m_t} × nGain/gain
15: for all m ∈ Ag and t ∈ T do
16:    Q^m_t(a^m_t, s) ← null_{a^m_t} + delta_{a^m_t}
```

The line 4 of the algorithm represents the value of an action which has an interaction of 0. Our intuition is that doing nothing ($noop_{a^m_t}$), and subtracting the resource used by the action, has the same value as doing an action which is sure of not realizing its purpose. The results of line 5 is the difference of utility from not considering the interactions to consider an interaction of 0. Afterwards, we have two case to compute the Q-value of the current planning agent, considering an interaction with another planning agent. Firstly, in

Algorithm 2 Considering simultaneous actions.

1: **Function** ADJUST-SA(action a)
2: **for all** $t \in T$ **do**
3: $bound \leftarrow (R_{s_t} | s_t = \neg goal)$
4: **for all** $m \in Ag$ **do**
5: **if** $a_t^m \neq noop$ **then**
6: $null_{a_t^m} \leftarrow Q_t^m(noop_t^m, s(||res_t^m - res_t^m(a^m)||))$
7: **if** $UpBound_s^{a_t^m} > bound$ **then**
8: $bound \leftarrow UpBound_s^{a_t^m}$
9: $noopG \leftarrow null_{a_t^m}$
10: **for all** $m \in Ag$ **do**
11: **if** $a_t^m \neq noop$ **then**
12: $delta_{a_t^m} \leftarrow Q_t^m(a_t^m, s) - null_{a_t^m}$
13: $sum_t \leftarrow sum_t + delta_{a_t^m}$
14: $val_t \leftarrow val_t + ((bound - noopG) - val_t) \times (\frac{delta_{a_t^m}}{UpBound_s^{a_t^m} - null_{a_t^m}})$
15: **for all** $m \in Ag$ **do**
16: **if** $a_t^m \neq noop$ **then**
17: $Q_t^m(a_t^m, s) \leftarrow null_{a_t^m} + (delta_{a_t^m} \times \frac{val_t}{sum_t})$

line 9, we consider negative interactions such that $inter \leqslant 1$. In this case, $delta_{a_t^m}$ is adjusted by multiplying it with the interaction. On the other hand, if the interaction is positive (i.e. line 11) the adjustment is more complex. The line 12 represents the "gain" the current action has made according to the upper bound it may reach. Then, this "gain" is reviewed considering the interaction in line 13. Finally, the new $delta_{a_t^m}$ is obtained by multiplying it with fraction of the gain of the interaction and the gain without interaction. Finally, in line 16 all Q-values are updated considering the new $delta_{a_t^m}$.

To consider simultaneous actions, we use the ADJUST-SA(*action a*) function, which is defined as follow:

DEFINITION 3.4. ADJUST-SA*(action a): This function adjusts all task-Q-values of each planning agent m, considering a global action a, in a global state s. The adjustment is made according to the simultaneous actions of all other planning agents m′.*

Algorithm 2 details the ADJUST-SA(*action a*) function. The first part of the algorithm (i.e. line 6 to 9) finds the highest upper bound, considering all agents. The second part (i.e. line 10 to 14) calculates two terms. Firstly, *sum* represents what the agents expects to gain by planning for their action, by planning independently. Then, *val* computes the maximum value that all agents may have, considering that they are planning on the same task. When we have computed these two terms, we then multiply in line 17 the gain of value of planning for this action by the fraction *val/sum*. In this line, we also affect the new Q-value to this "adjusted" gain.

Now, the *central* agent knows how to compute the adjusted Q-value of each planning agent in a state, given the action of the other planning agents. The other function the *central* agent uses in its coordination process is the GLOBAL-VALUE() one.

3.5 Determining the value of a global action

To determine the action to execute in a state, the *central* agent has to calculate a global Q-value, considering each planning agent Q-values. This is be done in a precise manner by considering the task-Q-values. Before introducing the algorithm, we recall that we have a planning agent for each resource, and we also have a *mNoop* planning agent for the *noop* (no operation) action. The *noop* action have to be considered since this action may modify the probability to achieve certain tasks in a state.

Algorithm 3 Computing the Q-value of a state.

1: **Function** GLOBAL-VALUE(Q-Value [] $Q^{Ag}(a^{Ag}, s)$)
2: $Q(a, s) \leftarrow 0$
3: **for all** $t \in T$ **do**
4: $val \leftarrow Q_t^{mNoop}(a_t^{mNoop}, s)$
5: **for all** $m \in Ag$ **do**
6: $val \leftarrow \max(Q_t^m(a_t^m, s), val + ((1 - val) \times Q_t^m(a_t^m, s) - Q_t^{mNoop}(a_t^{mNoop})))$
7: $Q(a, s) \leftarrow Q(a, s) + val$

Algorithm 3 determines a precise value of the adjusted task-Q-value of many planning agents in a state. The central part of this algorithm is in line 6 where the value of a task is computed, according to all agents. Here, we take as value, the maximum between the Q-value of the current agent m, and the gain that the current agent may offer. In this case, we subtract the Q-value of a planning agent m by the *mNoop* planning agent, because each agent considers also the *noop* action. an example, where we suppose the following adjusted values: We now describe the last function the *central* agent needs to coordinate the planning agent in a near-optimal manner at each iteration; the adaptation of the value iteration [2] algorithm for MTAMDPs.

3.6 Value Iteration for MTAMDPs

The value iteration MTAMDP algorithm is presented in Algorithm 4. In lines 6 to 9 of this algorithm, a Q-value is computed for all task-state-action tuples for each planning agent. The agents are limited by L_{res} and G_{res} while planning their respective policies. Afterwards, in lines 13 and 14, the *central* agent adjusts the value of all action combinations, in all possible states using the ADJUST-I(*action a*) and ADJUST-SA(*action a*) functions (i.e. Algorithm 1 and 2). When the adjusted value of each action is determined, we compute its global value $V'(s)$ in line 15. If this global Q-value is the maximum one up to now, we assign the value of each planning agent to the the adjusted Q-value (i.e. $V'^m(s^m) \leftarrow Q^m(a^m, s^m)$ in line 18). We also assign the new value of the state to the global value obtained by GLOBAL-VALUE() (i.e. $V'(s) \leftarrow Q(a, s)$ in line 19). When the global value function has converged, this policy is used for execution. We do not have a formal theorem to prove the convergence of the value iteration MTAMDP algorithm, but all experiments we made, as presented in the next section, resulted in a convergence.

4. DISCUSSION AND EXPERIMENTATIONS

Modelling a stochastic resource allocation problem using MTAMDPs allows diminishing the number of actions to consider in a given state. The difference in complexity between MMDPs and MTAMDPs resides in the reduction of the computational leverage from using Equation 2, in contrast to

Algorithm 4 MTAMDP-VALUE-ITERATION.

Function MTAMDP-VI(states S, error ϵ)
returns a value function V
{Planning agents part of the algorithm}
repeat
  $V \leftarrow V'$
  $\delta \leftarrow 0$
  for all $m \in Ag$ **do**
    $V^m \leftarrow V'^m$
    for all $s^m \in S^m$ **and** $a^m \in A^m(s)$ **do**
      $Q^m(a^m, s^m) \leftarrow R(s^m) + \sum_{s'^m \in S^m} P_{a^m}(s'^m|s^m) V^m(s'^m)$
  {*Central* agent part of the algorithm}
  for all $s \in S$ **do**
    $V'(s) \leftarrow R(\neg s)$
    for all $a \in A(s)$ **do**
      ADJUST-I(a)
      ADJUST-SA(a)
      $Q(a, s) \leftarrow$ GLOBAL-VALUE()
      if $Q(a, s) > V'(s)$ **then**
        for all $m \in Ag$ **do**
          $V'^m(s^m) \leftarrow Q^m(a^m, s^m)$
        $V'(s) \leftarrow Q(a, s)$
    if $|V'(s) - V(s)| > \delta$ **then**
      $\delta \leftarrow |V'(s) - V(s)|$
until $\delta < \epsilon$
return V

using Algorithms 1, 2, and 3 when computing the value of each action combination.

The MMDP approach is computed as a traditional "flat" MDP on the joint action and state spaces of all agents. The domain of the experiments is an army platform which must counter incoming missiles (i.e. tasks) by using its resources (i.e. weapons, movements). For the experiments, we generated 100 randomly generated resource allocation problems for all combinations of number of tasks and resource type, where one agent manages each resource type. There are three types of states, firstly transitional states where no action is possible to modify the transition probabilities. Then, action states, where actions modify the transition probabilities. Finally, there are final states. The state transitions are all stochastic because when a missile is in a given state, it may always transit in many possible states.

We compare the MTAMDP approach with an MMDP approach in Figure 6 and 7, where each agent manages a distinct resource type. The results are very conclusive. For instance, it takes 1978.0 seconds to plan for an MMDP approach with three tasks (see Figure 6). The MTAMDP approach solves near-optimally the same type of problem in an average of 6.93 seconds. Further results are given on Figure 7. As a matter of fact, we can observe that when the number of possible actions increase, the planning time for MMDPs suffers much more than for MTAMDPs.

Table 1 details how far the expected value of our MTAMDP approach is from an optimal MMDP approach. With one agent, the MTAMDP approach is optimal since no coordination is needed. This result suggests that Algorithm 3 is optimal. All the tests performed with two agents resulted in an optimal policy. This result needs further investigation to draw any valid conclusion.

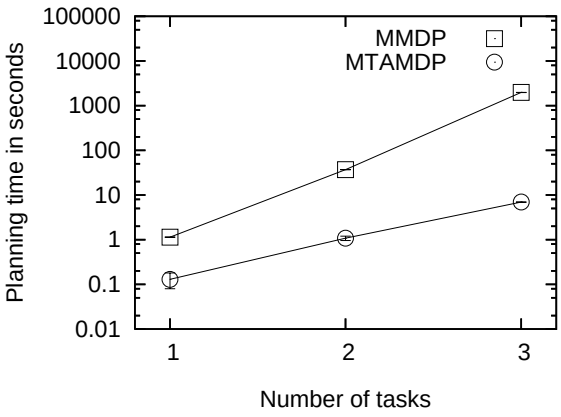

Figure 6: Gains in computational efficiency with five agents.

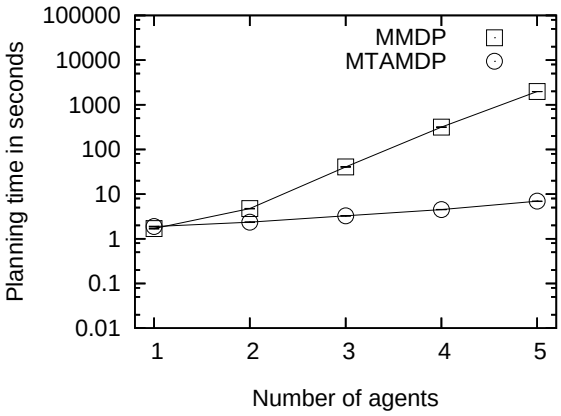

Figure 7: Gains in computational efficiency with three tasks.

Table 1: The percentage of the optimal obtained with MTAMDPs.

	1 task	2 tasks	3 tasks
1 agent	100%	100%	100%
2 agents	100%	100%	100%
3 agents	99, 99%	99, 99%	99, 99%
4 agents	99, 99%	99, 99%	99, 98%
5 agents	99, 99%	99, 98%	99, 98%

5. CONCLUSION AND FUTURE WORK

A new approach has been proposed to tackle the planning problem for resource allocation in a stochastic environment. The Multiagent Task Associated Markov Decision Process (MTAMDP) framework has been introduced to reduce the computational leverage induced by the high number of possible actions. The experimentation has shown that the MTAMDP provides a potential solution to solve efficiently real-time stochastic resource allocation problem with positive and negative interactions.

A way to improve the proposed MTAMDP consists of providing a more precise upper bound on the value that a Q-value may have. Currently, the approach simply considers the maximum value of the possible states transitions. This way, the upper bound overestimates the possible value of a state since it is very improbable that an action would guarantee reaching the upper bound. Other heuristics need to be explored to define the upper bound.

A way to improve the MTAMDP consists of providing a more precise may be to coordinate the agents using a Partial Global Planning (PGP) [5] approach instead of a *central* agent. The PGP approach solves the bottleneck effect induced by the *central* agent. Furthermore, the the coordination will be more precise, as only interacting agents will coordinate with each other. Another promising future work is to extend our works with the one of [15]. The Singh and Cohn approach solves optimally a stochastic resource allocation problem by using an upper bound and a lower bound on the value of each state. Their approaches needs some modifications to be used for our type of problem, as we consider positive and negative interaction. Our idea is to solve the problem using the MTAMDP approach, but with a upper and lower bounds on the value of each states. Finally, we would like to consider the cost of communication between agents.

6. REFERENCES

[1] D. Aberdeen, S. Thiebaux, and L. Zhang. Decision-theoretic military operations planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Whistler, Canada, 3–7 June 2004.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[3] D. Bertsekas. Rollout algorithms for constrained dynamic programming. Technical report 2646, Lab. for Information and Decision Systems, MIT, Mass., USA, 2005.

[4] C. Boutilier. Sequential optmiality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, Stockholm, August 1999.

[5] K. S. Decker and V. R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent Cooperative Information Systems*, 1(2):319–346, 1992.

[6] D. A. Dolgov and E. H. Durfee. Optimal resource allocation and policy formulation in loosely-coupled markov decision processes. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 315–324, June 2004.

[7] E. A. Feinberg and A. Shwartz. Constrained discounted dynamic programming. *Mathematics of Operations Research*, 21:922–945, 1996.

[8] P. A. Hosein and M. Athans. Some analytical results for the dynamic weapon-target allocation problem. *Naval Research Logistics*, February 1990.

[9] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts, 1960.

[10] N. Meuleau, M. Hauskrecht, K. Kim, L. Peshkin, L. P. Kaebling, T. Dean, and C. Boutilier. Solving very large weakly coupled markov decision processes. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 165–172. AAAI Press, 1998.

[11] A. Mouaddib and S. Zilberstein. Optimal scheduling of dynamic progressive processing. In *European conference on artificial intelligence (ECAI)*, pages 499–503, August 1998.

[12] P. Plamondon, B. Chaib-draa, and A. Benaskeur. Decomposition techniques for a loosely-coupled resource allocation problem. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005)*, September 2005.

[13] P. Plamondon, B. Chaib-draa, and A. Benaskeur. A multiagent task associated mdp (mtamdp) approach to resource allocation. In *AAAI 2006 Spring Symposium on Distributed Plan and Schedule Management*, March 2006.

[14] M. Salles, A. Mouaddib, and B. Chaib-draa. Progressive reasonning approach for operational research and military applications. Working paper, Laval University, Canada, 2005.

[15] S. Singh and D. Cohn. How to dynamically merge markov decision processes. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 1057–1063, Cambridge, MA, USA, 1998. MIT Press.

[16] C. C. Wu and D. A. Castanon. Decomposition techniques for temporal resource allocation. Technical report: Afrl-va-wp-tp-2004-311, Air Force Research Laboratory, Air force base, OH, 2004.

[17] J. Wu and E. H. Durfee. Automated resource-vriven mission phasing techniques for constrained agents. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 331–338, August 2005.

[18] W. Zhang. Modeling and solving a resource allocation problem with constraint techniques. Technical report: Wucs-2002-13, Washington University, Saint-Louis, Missouri, 2002.