An Efficient Resource Allocation Approach in Real-time Stochastic Environment

Pierrick Plamondon[1], Brahim Chaib-draa[1], and Abder Rezak Benaskeur[2]

[1] Computer Science & Software Eng. Dept, Laval University
{plamon, chaib}@damas.ift.ulaval.ca
[2] Decision Support Systems Section, Defence R&D Canada — Valcartier
Abderrezak.Benaskeur@drdc-rddc.gc.ca

Abstract. We are interested in contributing to solving effectively a particular type of real-time stochastic resource allocation problem. Firstly, one distinction is that certain tasks may create other tasks. Then, positive and negative interactions among the resources are considered, in achieving the tasks, in order to obtain and maintain an efficient coordination. A standard Multiagent Markov Decision Process (MMDP) approach is too prohibitive to solve this type of problem in real-time. To address this complex resource management problem, the merging of an approach which considers the complexity associated to a high number of different resource types (i.e. Multiagent Task Associated Markov Decision Processes (MTAMDP)), with an approach which considers the complexity associated to the creation of task by other tasks (i.e. Acyclic Decomposition) is proposed. The combination of these two approaches produces a near-optimal solution in much less time than a standard MMDP approach.

1 Introduction

Resource allocation problems are known to be NP-Complete [12]. Since resources are usually constrained, the allocation of resources to one task restricts the options available for other tasks. The action space is exponential according to the number of resources, while the state space is exponential according to the number of resources and tasks. The very high number of states and actions in this type of problem coupled with the time constraint makes it very complex, and here a reduction in the computational burden associated to the high number of different resource types is proposed. Many approximations and heuristics have been proposed ([2], [10], [1]). However, all these cited authors do not consider positive and negative interactions between resources. These interactions mean that a resource efficiency to realize a task is changed when another resource is used on the same task. Their approaches are consequently not very suitable to the type of problem tackled in this paper, since in many real applications there are positive and negative interactions between resources. An effective approach, as considered in the current paper, is to plan for the resources separately as proposed by Wu and Castanon [10]. Wu and Castanon formulates a policy for each resource and a greedy global policy is produced by considering each resource in turn, producing an approximate policy. Their coordination rules are sometime very specific to the problem's characteristics.

Since resources have local and global resource constraints on the number that can be used, the problem here can be viewed as a constrained Markov Decision Process ([2], [11]). In this context, dynamic programming [2] or linear programming [11] may be used to obtain a policy. Much work has been done in this field, but none of it considers positives and negative interactions among resource, as well as creation of tasks by other tasks.

This paper combines two approaches in a synergic manner to reduce the planning time. In the first approach, a planning agent manages each specific resource. These planning agents are coordinated together during the planning process by a *central* agent, and produce a near-optimal policy. On the other hand, the second approach is a decomposition technique which solves the problem efficiently by grouping cyclic states in separate components. The results obtained by the merging of these two approaches are very satisfying. The policy is near-optimal, while the convergence time is very small compared to a standard Multiagent Markov Decision Process (MMDP) [3] approach on the joint action and state space of all agents. The problem is now formulated in more detail.

2 Problem Formulation

An abstract resource allocation problem is described in Figure 1 (a). In this example, there are 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 being constrained by the number 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. An action is considered as the resource allocation to a group of tasks. 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. The solution of this type of problem is called a policy. A policy π maps all states s into actions $a \in A(s)$ to maximize the expectation of realizing all tasks. The realization of a task is associated with a reward r. Thus, a policy maximizes the expected reward. The modelling of this type of problem is now detailed.

2.1 Multiagent Task Associated Markov Decision Process (MTAMDP)

Since resource allocation problems are known to be NP-Complete [12], one may decompose the previous problem into multiple planning agents. To do so, Multi-Agent Markov Decision Processes (MMDP) [3] may be a very suitable

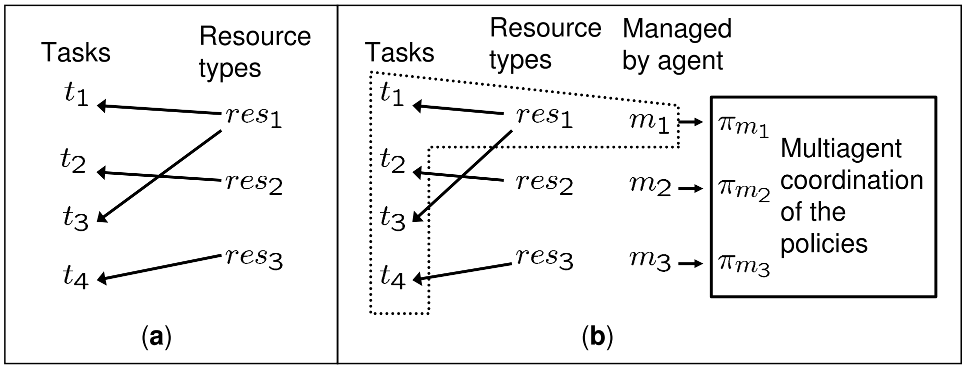

Fig. 1. Resource Allocation Problem.

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. Indeed, an MMDP is like a Markov Decision Process (MDP), except that the probability of reaching a state by executing an action now refers to the probabilities of joint actions. An abstract schematization of the approach proposed by Plamondon et al. [7], which extends MMDP, to solve efficiently a resource allocation problem is described in Figure 1 (b). This in an extension of Figure 1 (a) 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 represents 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}). The policies of the agents are needs to be coordinated for two reasons. First of all, positive and negative interactions among resource have to be considered 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. The second reason why the planning agents should coordinate is because an efficient allocation divides the resources between the tasks. Thus, coordination should be considered in the case of simultaneous actions on a given task.

Multiagent Task Associated Markov Decision Processes (MTAMDP) [7] proposes to coordinate the different agents at each iteration of the planning algorithm considering positive and negative interactions, and simultaneous actions. Indeed, all existing algorithms to solve an MDP are iterative, thus the approach presented here should be pretty extensible. Figure 2 (a) describes this process. For example, if n iterations are needed for each planning agent to converge, then n coordination activities are made. MTAMDP is now formally described.

A Multiagent Task Associated Markov Decision Process (MTAMDP) [7] is defined as a tuple $\langle Res, Ag, T, S, A, P, W, R, \rangle$, where:

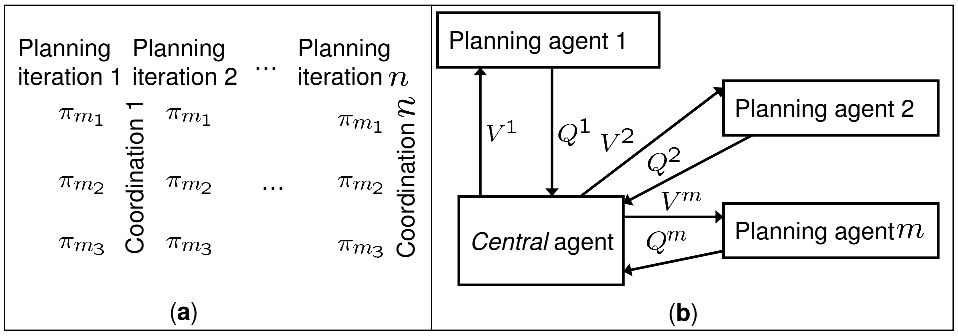

Fig. 2. The iterative coordination process.

- $Res = \{res\}$ is a finite set of resource types available for the global planning process. The planning process has Res resource types of res number of resources. Each resource type has a local resource constraint L_{res} on the number that may be used on a single step, and a global resource constraint G_{res} on the number 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, a planning agent for each resource, and a *mNoop* planning agent for the *noop* (no operation) action are considered. The expected value of the *noop* action has to be considered 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 characteristic 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 [6].
- State rewards $R = [r_s] : \sum_{t=1}^{nbTasks} r_{s_t}$. The relative reward of the state of a task r_{s_t} is the product of a real number $\Re$ by the weight factor w_t. 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, the notion of Q-value is used for a planning agent m in the MTAMDP approach:

$$Q^m(a^m, s^m) = R(s^m) + \sum_{s'^m \in S^m} P_{a^m}(s'^m|s^m)V^m(s'^m(|Res^m - \{a^m\}|)) \quad (1)$$

Let's consider $Q_t^m(a_t^m, s^m)$ as the part of $Q^m(a^m, s^m)$ related to task t. This part is called task-Q-value. A task-Q-value is not a decomposition, it simply means that the specific Q-value of a task within the global Q-value is referred to. Each Q-value is subjected to the local resource constraints for each state task s_t of a global state s at a particular step. Furthermore, a Q-value is constrained on the total amount of resource that may be used by a planning agent m.

In this paper, the agents are coordinated through a *central* agent. Figure 2 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. A description is made 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 1 (MTAMDP-VI(states S, error ϵ)) gives a more formal description of this approach, which uses the following functions: ADJUST-I(action a), ADJUST-SA(action a), and GLOBAL-VALUE().

2.2 The MTAMDP Functions

Adjusting Considering Interactions The ADJUST-I(*action a*) function [7] adjusts all task-Q-values of each planning agent m in a global state s according to the interactions among the actions of each other planning agent m'. In brief, this function uses an interaction parameter which quantifies the degree of interaction among two resources. For example an interaction of 0.5 means that the efficiency of a given resource is half its normal one when another resource is used simultaneously. In the case when the interaction is negative, the task-Q-value $Q_t^m(a_t^m, s)$ of an agent m is adjusted as follow:

$$Q_t^m(a_t^m, s) = null_{a_t^m} + ((Q_t^m(a_t^m, s) - null_{a_t^m}) \times inter(a_t^m, s|a_t'^{m'})) \quad (2)$$

, where $null_{a_t^m} = Q_t^m(noop_{a_t^m}, s(|Res_t^m - \{a_t^m\}|))$ represents the value of an action which has an interaction of 0. The intuition is that doing nothing ($noop_{a_t^m}$), and subtracting the resource used by the action, has the same value as doing an action which is sure of not realizing its purpose. $inter(a_t^m, s|a_t^{\prime m'})$ is the value of the interaction between the action of the planning agents m with another agent m'.

Furthermore, to adjust the value in the case of a positive interaction (i.e. interaction > 1), an upper bound on the Q-value is needed. The heuristic used to determine the upper bound for a state s, and action a_t^m, by agent m, is the highest value of a possible state transition. A possible state transition is considered as, a state for which $P_{a_t^m}(s'|s) > 0$. 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 and scheduled for future work.

Adjusting Considering Simultaneous Actions The ADJUST-SA (*action a*) function [7] adjusts all task-Q-values of each planning agent m in a global state s according to the simultaneous actions of all other planning agents m'. An upper bound on the Q-value that an agent may obtain is also used when adjusting considering simultaneous actions. This function reduces in a formal way the Q-value of two agents planning their resource simultaneously as exemplified in Section 2.1. To do so, the algorithm calculates two main terms: the *sum* and *val*. Firstly, the *sum* term computes the global value gain the agents make by planning independently. Then, the *val* term, computes the maximum gain the agents may have globally, considering the other agents actions, and the upper bound. Then all Q-value of the agents are "adjusted" by multiplying the initial gain to plan for this action by the ratio val/sum. An equation which summarizes the ADJUST-SA (*action a*) function is as follow:

$$\sum_{m\in Ag} Q_t^m(a_t^m, s) = null_{a_t^m} + ((Q_t^m(a_t^m, s) - null_{a_t^m})\times \frac{\sum_{m\in Ag} val = val + (((bound - noopG) - val) \times (\frac{Q_t^m(a_t^m,s) - null_{a_t^m}}{UpBound_s^{a_t^m} - null_{a_t^m}}))}{sum = \sum_{m\in Ag} Q_t^m(a_t^m, s) - null_{a_t^m}}) \quad (3)$$

, where $val = 0$ and $sum = 0$ a priori. $UpBound_s^{a_t^m}$ is the upper bound of an action by an agent in a state. *bound* is the maximum upper bound of all planning agents. *noopG* is $null_{a_t^m}$ for the agent which has the highest upper bound.

Global Q-value 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 done in a precise manner by considering the task-Q-values.

Before introducing the algorithm, we recall that a planning agent for each resource type, and a *mNoop* planning agent for the *noop* (no operation) action are considered. The *noop* action has to be considered since this action may modify the probability to achieve certain tasks in a state. The global Q-value $Q(a, s)$ of a state is:

$$\sum_{t \in T} Q(a,s) = Q(a,s) + val \sum_{m \in Ag} val = \max(Q_t^m(a_t^m, s), val + ((1 - val) \times Q_t^m(a_t^m, s) - Q_t^{mNoop}(a_t^{mNoop}))) \quad (4)$$

, where $Q(a, s) = 0$ a priori. $val = 0$ every time the $m \in Ag$ loop is entered. The main function that the *central* agent uses to coordinate the planning agent in a near-optimal manner at each iteration is now described.

2.3 Value Iteration for MTAMDPs

The value iteration MTAMDP algorithm is presented in Algorithm 1. 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. When the adjusted value of each action is determined, the global value $V'(s)$ is computed in Line 15. If this global Q-value is the maximum one at present, the value of each planning agent is assigned to the adjusted Q-value (i.e. $V'^m(s^m) = Q^m(a^m, s^m)$ in Line 18). The new value of the state is also assigned to the global value obtained by GLOBAL-VALUE() (i.e. $V'(s) = Q(a, s)$ in Line 19). When the global value function has converged, this policy is used for execution. All the performed experiments (Section 3) resulted in a convergence. This paper does not present a formal theorem to prove the convergence, or the near-optimality of the algorithm. These proofs are for future work.

An important characteristic of the resource allocation problem in this paper is that tasks may create other tasks. The task transitions in global are acyclic since these transitions always result in a group of tasks that were never visited previously, thus implying a partial order on the set of tasks. The acyclic decomposition algorithm [6] to effectively consider this characteristic is now described.

2.4 Acyclic Decomposition Algorithm

An efficient decomposition technique generally tries to regroup cyclic states. In the same sense, it is known that a planning problem which may be represented with an acyclic graph, instead of a cyclic one, is generally easier to

Algorithm 1 MTAMDP-VALUE-ITERATION from [7].

```
1: **Fun** MTAMDP-VI(states $S$, error $\epsilon$)
2: **returns** a value function $V$
   {Planning agents part of the algorithm}
3: **repeat**
4:    $V \leftarrow V'$
5:    $\delta \leftarrow 0$
6:    **for all** $m \in Ag$ **do**
7:       $V^m \leftarrow V'^m$
8:       **for all** $s^m \in S^m$ **and** $a^m \in A^m(s)$ **do**
9:          $Q^m(a^m, s^m) \leftarrow R(s^m) + \sum_{s'^m \in S^m} P_{a^m}(s'^m|s^m) V^m(s'^m(|Res^m - \{a^m\}|))$
      {*Central* agent part of the algorithm}
10:   **for all** $s \in S$ **do**
11:      $V'(s) \leftarrow R(\neg s)$
12:      **for all** $a \in A(s)$ **do**
13:         ADJUST-I($a$)
14:         ADJUST-SA($a$)
15:         $Q(a, s) \leftarrow$ GLOBAL-VALUE()
16:         **if** $Q(a, s) > V'(s)$ **then**
17:            **for all** $m \in Ag$ **do**
18:               $V'^m(s^m) \leftarrow Q^m(a^m, s^m)$
19:            $V'(s) \leftarrow Q(a, s)$
20:      **if** $|V'(s) - V(s)| > \delta$ **then**
21:         $\delta \leftarrow |V'(s) - V(s)|$
22: **until** $\delta < \epsilon$
23: **return** $V$
```

solve. We recall that an acyclic graph is one where all state transitions always result in states that were never previously visited, thus implying a partial order on the set of states. On the other hand, a cyclic graph may visit certain states many times, which is not computationally efficient. The idea here is to transform our problem into an abstract acyclic one, which contains many cyclic components. A component corresponds to a group of tasks, and the graph contains a component for each possible task combination. On the other hand, the acyclic graph represents the possible task transitions. One may use Tarjan's [9] linear algorithm to detect the *strongly-connected components* of a directed graph to create the acyclic graph. Figure 3 shows the acyclic graph when task t_1 (a) or t_1 and t_2 (b), are in the environment. t_1 may create task t_3, and t_2 may create task t_4. So, task t_3 and t_4, which are leafs, may produce dire consequences for an executing agent. All nodes represent a cyclic component. This graph supposes that a task may create one other task at maximum. The significance of a link in Figure 3, simply means that the planning agent has a different group of tasks to achieve. Thus, it has a task transition meaning. Once the abstract acyclic graph is formed, it is solved using a backward approach just like in the AO* algorithm [5]. Algorithm 2 describes how the value V_c of each cyclic component c of an acyclic graph AcG is calculated.

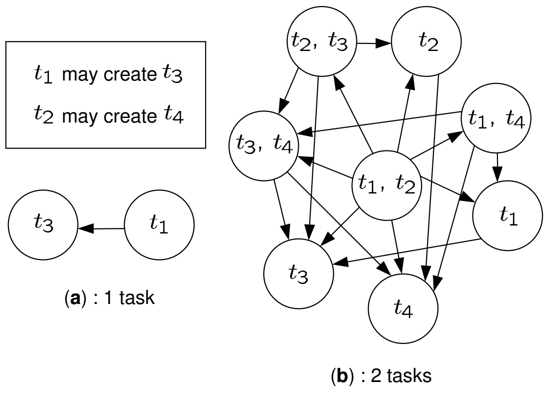

Fig. 3. The acyclic graph of cyclic components.

Algorithm 2 Acyclic decomposition from [6].

```
1: **Function** ACYCLIC-DEC(error, $\epsilon$, graph $AcG$)
2: **while** $AcG \neq null$ **do**
3:     Remove from $AcG$ a component $c$, such that no descendent of $c$ is in $AcG$
4:     $V_c \leftarrow$MDP-ALGO($c$, $\epsilon$)
```

This algorithm solves each component, using an MDP algorithm ("MDP-ALGO(c)"), from the leaf to the root of the graph. This way, each component may only transit to a solved component, and thus each component has to be solved once. "MDP-ALGO(c)" may be any algorithm to solve an MDP, such as standard approaches like value iteration or policy iteration. For example, in Figure 3 (b), one can choose to remove whichever of t_3 or t_4, and solve it using MDP-ALGO(c). Then, if t_3 is removed in the first iteration, we may now remove t_4, or vice versa. When both t_3 and t_4 are removed, t_1 and t_2 may be removed. Components are removed in this order, until the component t_1, t_2 is removed and solved. In [6] the optimality of the acyclic decomposition algorithm is proved.

2.5 Merging the Acyclic Decomposition and MTAMDP approaches (MTAMDP Decomposition)

The merging of the acyclic decomposition and the MTAMDP approaches is pretty straightforward. Indeed the line $V_c \leftarrow$MDP-ALGO(c, ϵ) in Algorithm 2 is now $V_c \leftarrow$MTAMDP-VI(c, ϵ). The fact that only this simple change is needed demonstrates the flexibility and extensibility of both these approaches.

3 Discussion and Experimentations

Modelling a stochastic resource allocation problem using the MTAMDP decomposition approach allows reducing the number of actions to consider in a given state. In particular, the difference in complexity between MMDPs and MTAMDPs resides in the reduction of the computational complexity from using the complex Bellman equation, in contrast to using the ADJUST-I(*action a*), ADJUST-SA(*action a*), and GLOBAL-VALUE() functions when computing the value of each action combination.

The domain of the experiments is a naval platform which must counter platforms, which may launch incoming missiles (i.e. tasks) by using its resources (i.e. weapons, movements). The different resources have their efficiency modified when used in conjunction on a same task, thus producing positive and negative interactions among resources. In this kind of problem, a platform may create a missile, but not vice versa, thus the task transition is acyclic. Thus the acyclic decomposition algorithm may be employed efficiently. For the experiments, 100 randomly resource allocation problems for all combinations of number of tasks and different number of resources, where one agent manages each resource type were generated. 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 platform or a missile is in a given state, it may always transit in many possible states.

We have compared four different approaches. The first one is the MMDP approach as described briefly in Section 2.1. MMDP is computed as a traditional "flat" MDP on the joint action and state spaces of all agents. The second approach is the MTAMDP as described in Algorithm 1. The third one is the "*one step* MTAMDP" where the adjustment of the Q-values is made only at the last iteration for the planning agents. Thus, when each planning agent have converged, their Q-values are adjusted, and used for execution. The fourth approach is the "*no coordination*" where each planning agent plans their resource completely independently of each other. We have compared these four approaches in both the standard (no acyclic decomposition), and acyclic decomposition mode, to efficiently consider the creation of tasks by other tasks.

We compare the MTAMDP approach with an MMDP approach in Figure 4, where each agent manages a distinct resource type. The acyclic decomposition algorithm further reduces the planning time for the four different approaches as presented in Figure 5. The results are very encouraging. For instance, it takes 51.64 seconds to plan for an acyclic decomposition MMDP approach with five agents. The acyclic decomposition MTAMDP approach solves the same type of problem in an average of 0.72 seconds.

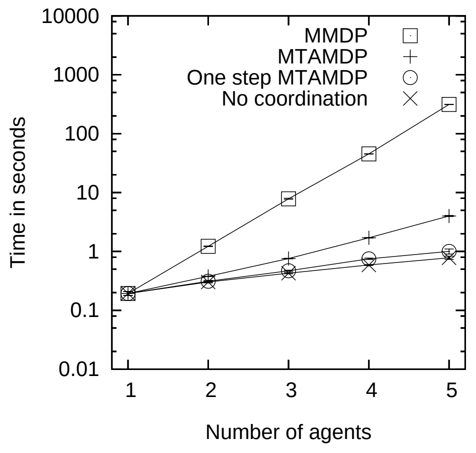

Fig. 4. Planning time using no acyclic decomposition.

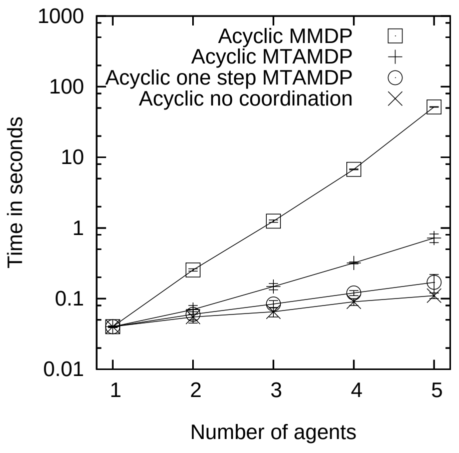

Fig. 5. Planning time using the acyclic decomposition.

Table 1 details how far the expected value of the MTAMDP, *one step* MTAMDP, and *no coordination* approaches are from an optimal MMDP approach. With one agent, all approaches are optimal since no coordination is needed. This result suggests that the GLOBAL-VALUE() function is optimal. All the tests performed with two agents resulted in an optimal policy for the MTAMDP approach. This result suggests that the GLOBAL-VALUE() function is optimal, and a formal theorem to proof that is for future work. One can also observe that when the agents do not coordinate, the resulting policy is far from the optimal, which is not the case for the MTAMDP coordination approach. The one step MTAMDP could be a viable approach in certain critical situations, since the solution is produced much faster than the MTAMDP approach while providing a near-optimal policy.

Table 1. The percentage of the optimal obtained with the different approaches.

	MTAMDP	One step MTAMDP	No Coordination
1 agent	100%	100%	100%
2 agents	100%	99.89%	97.48%
3 agents	99.84%	99.63%	94.20%
4 agents	99.79%	99.55%	91.63%
5 agents	99.67%	99.41%	89.37%

4 Conclusion and Future Work

The Multiagent Task Associated Markov Decision Process (MTAMDP) framework has been introduced to reduce the computational burden induced by the high number of different resource types. The acyclic decomposition approach aims at reducing the computational leverage burden associated to the state space in a problem where tasks create other tasks. The merging of these two approaches gives an efficient, and novel way to tackle the planning problem for resource allocation in a stochastic environment.

A way to improve the MTAMDP consists of coordinating the agents using the efficient Partial Global Planning (PGP) [4] approach instead of the *central* agent. The PGP approach solves the bottleneck effect induced by the *central* agent. Furthermore, the coordination will be less complex, as only interacting agents will coordinate with each other. The Q-decomposition approach proposed by Russell and Zimdars [8] may enable to approximate efficiently the resource allocation problem considered here. Indeed, the Q-decomposition would decompose the problem in tasks, and the MTAMDP method decomposes the problem in resources. This would permit two degrees of decomposition.

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. D. Bertsekas. Rollout algorithms for constrained dynamic programming. Technical report 2646, Lab. for Information and Decision Systems, MIT, Mass., USA, 2005.
3. C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, Stockholm, August 1999.
4. 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.
5. N. J. Nilsson. *Principles or Artificial Intelligence*. Tioga Publishing, Palo Alto, Ca, 1980.
6. 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.
7. 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.
8. S. J. Russell and A. Zimdars. Q-decomposition for reinforcement learning agents. In *ICML*, pages 656–663, 2003.
9. R. E. Tarjan. Depth first search and linear graph algorithm. *SIAM Journal on Computing*, 1(2):146–172, 1972.
10. 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.
11. J. Wu and E. H. Durfee. Automated resource-driven 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.
12. W. Zhang. Modeling and solving a resource allocation problem with soft constraint techniques. Technical report: Wucs-2002-13, Washington University, Saint-Louis, Missouri, 2002.