Tight Bounds for a Stochastic Resource Allocation Algorithm Using Marginal Revenue

Pierrick Plamondon and **Brahim Chaib-draa**
Laval University
Computer Science & Software Engineering Dept
{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

This paper contributes to solve effectively stochastic resource allocation problems known to be NP-Complete. To address this complex resource management problem, previous works on pruning the action space of real-time heuristic search is extended. The pruning is accomplished by using upper and lower bounds on the value function. This way, if an action in a state has its upper bound lower than the lower bound on the value of this state, this action may be pruned in the set of possible optimal actions for the state. This paper extends this previous work by proposing tight bounds for problems where tasks have to be accomplished using limited resources. The marginal revenue bound proposed in this paper compares favorably with another approach which proposes bounds for pruning the action space.

Introduction

This paper aims to contribute to solve complex stochastic resource allocation problems. In general, resource allocation problems are known to be NP-Complete (Zhang 2002). In such problems, a scheduling process suggests the action (i.e. resources to allocate) to undertake to accomplish certain tasks, according to the perfectly observable state of the environment. When executing an action to realize a set of tasks, the stochastic nature of these actions induces probabilities on the next visited state. The number of states is the combination of all possible specific states of each task and available resources. In this case, the number of possible actions in a state is the combination of each individual possible resource assignment to the tasks. The very high number of states and actions in this type of problem makes it very complex.

A common way of addressing this large stochastic problem is by using Markov Decision Processes (MDPs), and in particular real-time search where many algorithms have been developed recently. For instance Real-Time Dynamic Programming (RTDP) (Barto, Bradtke, & Singh 1995), LRTDP (Bonet & Geffner 2003b), HDP (Bonet & Geffner 2003a), and LAO* (Hansen & Zilberstein 2001) are all state of the art heuristic search approaches in a stochastic environment. Because of its anytime quality, an interesting approach is RTDP introduced by Barto *et al.* (Barto, Bradtke, & Singh 1995) which updates states in trajectories from an initial state s_0 to a goal state s_g in a very efficient way. Then, Bonet and Geffner (Bonet & Geffner 2003b) proposed a labeling procedure to accelerate the convergence of RTDP in their L(Labeled)RTDP algorithm. RTDP is used in this paper to solve efficiently a constrained resource allocation problem.

RTDP is much more effective if the action space can be pruned of sub-optimal actions. To do this, McMahan *et al.* (McMahan, L., & Gordon 2005), Smith and Simmons (Smith & Simmons 2006), and Singh and Cohn (Singh & Cohn 1998) proposed solving a stochastic problem using a RTDP type heuristic search with upper and lower bounds on the value of states. The approach by Smith and Simmons proposes an efficient trajectory of state updates to further speed up the convergence, when given upper and lower bounds. This efficient trajectory of state updates can be combined to the approach proposed here as this paper focusses on the definition of tight bounds for a constrained resource allocation problem.

The bounds proposed by McMahan *et al.* are related to a stochastic shortest path problem. Their approach is well suited for problems where the rewards (or costs) are obtained whenever an action is executed in a state. In our case, rewards are only obtained when tasks are achieved, thus the McMahan *et al.* approach is not applicable to our problem.

On the other hand, the approach by Singh and Cohn is suitable to our case, and extended in this paper using, in particular, the concept of *marginal revenue* (Pindyck & Rubinfeld 2000) to elaborate tight bounds. This paper proposes new algorithms to define upper and lower bounds in the context of a RTDP heuristic search approach. Our marginal revenue bounds are compared theoretically and empirically to the bounds proposed by Singh and Cohn. Also, even if the algorithm used to obtain the optimal policy is RTDP, our bounds can be used with any other algorithm to solve an MDP. The only condition on the use of our bounds is to be in the context of stochastic constrained resource allocation. The problem is now modelled.

Problem Formulation

Figure 1 gives an example of a stochastic resource allocation problem to execute tasks. In this problem, there are two tasks to realize: $ta_1 = \{\text{wash the dishes}\}$, and $ta_2 =$

{clean the floor}. These two tasks are either in the realized state, or not realized state. Thus, the combination of the specific states of the individual tasks determines the four global states in Figure 1. To realize the tasks, two type of resources are assumed: $res_1 = \{\text{brush}\}$, and $res_2 = \{\text{detergent}\}$. A computer has to compute the optimal allocation of these resources to the cleaner robots to realize their tasks. In this problem, a state represents a conjunction of the particular state of each task, and the available resources. The resources may be constrained by the amount that may be used simultaneously (local constraint), and in total (global constraint). Furthermore, the higher is the number of resources allocated to realize a task, the higher is the expectation of realizing the task. For this reason, when the specific states of the tasks change, or when the number of available resources changes, the value of this state may change.

As discussed, a state s includes the joint states of the tasks, and the available resources. When executing an action a in state s, the specific states of the tasks change stochastically, and the remaining resource are determined with the resource available in s, subtracted from the resources used by action a, if the resource is consumable. Indeed, our model may consider *consumable* and *non-consumable* resource types. A consumable resource type is one where the amount of available resource is decreased when it is used. On the other hand, a non-consumable resource type is one where the amount of available resource is unchanged when it is used. In the example of Figure 1, the brush is a non-consumable resource, while the detergent is a consumable resource. Figure 2 describes the state transition process. The system is in a state s with a set of task Ta to realize, and a set Res of resource available. A possible action in this state may be to allocate one unit of detergent to task ta_1, and one brush to task ta_2. The state of the system changes stochastically, as each task's state does. For example, the floor may be clean or not with a certain probability, after having allocated the brush to clean it. In this example, the state of the tasks may change, for example, in n new possible combinations. For all these n possible state transitions after s, the consumable resources available (Res_c) are $Res_c \setminus res(a)$, where $res(a)$ is the consumable resources used by action a.

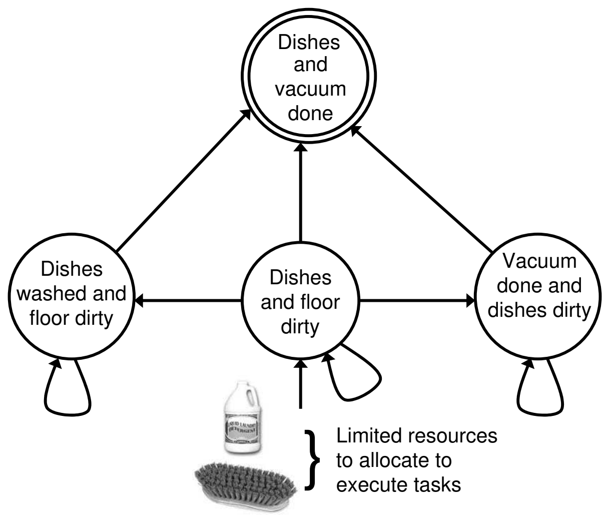
Dishes and vacuum done
Dishes washed and floor dirty
Dishes and floor dirty
Vacuum done and dishes dirty
Limited resources to allocate to execute tasks

Figure 1: Task transition graph.

$$\begin{tikzcd}
& s(Ta, Res) & \\
& a(s) = \langle detergent_{ta_1}, brush_{ta_2} \rangle \arrow[dl] \arrow[d] \arrow[dr] & \\
s_1(Ta_1, Res \setminus \{a\}) & \cdots & s_n(Ta_n, Res \setminus \{a\})
\end{tikzcd}$$

Figure 2: State transition graph.

Markov Decision Processes (MDPs) in the Context of Resource Allocation

A Markov Decision Process (MDP) framework is used to model our stochastic resource allocation problem. MDPs have been widely adopted by researchers today to model a stochastic process. This is due to the fact that MDPs provide a well-studied and simple, yet very expressive model of the world.

An MDP in the context of a resource allocation problem with limited resources is defined as a tuple $\langle Res, Ta, S, A, P, W, R, \rangle$, where:

- $Res = \langle res_1, ..., res_{|Res|} \rangle$ is a finite set of resource types available for a planning process. Each resource type may have a local resource constraint L_{res} on the number that may be used in a single step, and a global resource constraint G_{res} on the number that may be used in total. The global constraint only applies for consumable resource types (Res_c) and the local constraints always apply to consumable and non-consumable resource types.
- Ta is a finite set of tasks with $ta \in Ta$ to be accomplished.
- S is a finite set of states with $s \in S$. A state s is a tuple $\langle Ta, \langle res_1, ..., res_{|Res_c|} \rangle \rangle$, which represents the particular state s_{ta}, which is the characteristic of each unaccomplished task $ta \in Ta$ in the environment, and the available consumable resources. Also, S contains a non empty set $s_g \subseteq S$ of goal states. A goal state is a sink state where an agent stays forever.
- A is a finite set of actions (or assignments). The actions $a \in A(s)$ applicable in a state are the combination of all resource assignments that may be executed, according to the state s. Thus, a is simply an allocation of resources to the current tasks, and a_{ta} is the resource allocation to task ta. The possible actions are limited by L_{res} and G_{res}.
- Transition probabilities $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$.
- $W = [w_{ta}]$ is the relative weight (criticality) of each task.
- State rewards $R = [r_s] : \sum\limits_{ta \in Ta} r_{s_{ta}} \leftarrow \Re_{s_{ta}} \times w_{ta}$. The relative reward of the state of a task $r_{s_{ta}}$ is the product of a real number $\Re_{s_{ta}}$ by the weight factor w_{ta}. For our problem, a reward of 1 is given when the state of a task (s_{ta}) is in an achieved state, and 0 in all other cases.

- A discount factor γ, which is a real number between 0 and 1. The discount factor describes the preference of an agent for current rewards over future rewards.

A solution of an MDP is a policy π mapping states s into actions $a \in A(s)$. In particular, $\pi_{ta}(s)$ is the action (i.e. resources to allocate) that should be executed on task ta, considering the global state s. In this case, an optimal policy is one that maximizes the expected total reward for accomplishing all tasks. The optimal value of a state, $V(s)$, is given by:

$$V^{\star}(s) = R(s) + \max_{a \in A(s)} \gamma \sum_{s' \in S} P_a(s'|s)V(s') \qquad (1)$$

where the remaining consumable resources in state s' are $Res_c \setminus res(a)$, where $res(a)$ are the consumable resources used by action a. Indeed, since an action a is a resource assignment, $Res_c \setminus res(a)$ is the new set of available resources after the execution of action a. Furthermore, one may compute the Q-Values $Q(a,s)$ of each state action pair using the following equation:

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

where the optimal value of a state is $V^{\star}(s) = \max_{a \in A(s)} Q(a,s)$. The policy is subjected to the local resource constraints $res(\pi(s)) \leq L_{res} \forall\ s \in S$, and $\forall\ res \in Res$. The global constraint for consumable resources is defined according to all system trajectories $tra \in TRA$. A system trajectory tra is a possible sequence of state-action pairs, until a goal state is reached under the optimal policy π. For example, state s is entered, which may transit to s' or to s'', according to action a. The two possible system trajectories are $\langle(s,a),(s')\rangle$ and $\langle(s,a),(s'')\rangle$. The global resource constraint is $res(tra) \leq G_{res} \forall\ tra \in TRA$,and $\forall\ res \in Res_c$ where $res(tra)$ is a function which returns the resources used by trajectory tra. Since the available consumable resources are represented in the state space, this condition is verified by itself. In other words, the model is Markovian as the history has not no be considered in the state space. Furthermore, the time is not considered in the model description, but it may also include a time horizon by using a finite horizon MDP. Since resource allocation in a stochastic environment is NP-Complete, heuristics should be employed. Bounded Real-Time Dynamic Programming (BOUNDED-RTDP) permits to focuss the search on relevant states, and to prune the action space A by using lower and higher bound on the value of states. BOUNDED-RTDP is now introduced.

BOUNDED-RTDP

Bonet and Geffner (Bonet & Geffner 2003b) proposed LRTDP as an improvement to RTDP (Barto, Bradtke, & Singh 1995). LRTDP is a simple dynamic programming algorithm that involves a sequence of trial runs, each starting in the initial state s_0 and ending in a goal or a *solved* state. Each LRTDP trial is the result of simulating the policy π while updating the values $V(s)$ using a Bellman backup (Equation 1) over the states s that are visited. $h(s)$ is a heuristic which define an initial value for state s. This heuristic has to be admissible — The value given by the heuristic has to overestimate (or underestimate) the optimal value $V^{\star}(s)$ when the objective function is maximized (or minimized). For example, an admissible heuristic for a stochastic shortest path problem is the solution of a deterministic shortest path problem. Indeed, since the problem is stochastic, the optimal value is lower than for the deterministic version.

It has been proven that LRTDP, given an admissible initial heuristic on the value of states cannot be trapped in loops, and eventually yields optimal values (Bonet & Geffner 2003b). The convergence is accomplished by means of a labeling procedure called CHECKSOLVED(s, ϵ). This procedure tries to label as solved each traversed state in the current trajectory. When the initial state is labelled as solved, the algorithm has converged.

In this section, a bounded version of RTDP (BOUNDED-RTDP) is presented in Algorithm 1 to prune the action space of sub-optimal actions. This pruning enables to speed up the convergence of LRTDP. BOUNDED-RTDP is similar to RTDP except there are two distinct initial heuristics for unvisited states $s \in S$; $h_L(s)$ and $h_U(s)$ in Line 10. Also, the CHECKSOLVED(s, ϵ) procedure can be omitted because the bounds can provide the labeling of a state as solved. On the one hand, $h_L(s)$ defines a lower bound on the value of s such that the optimal value of s is higher than $h_L(s)$. For its part, $h_U(s)$ defines an upper bound on the value of s such that the optimal value of s is lower than $h_U(s)$. The values of the bounds are computed in Lines 9 and 10. Computing these two Q-values is made simultaneously as the state transitions are the same for both Q-values. Only the values of the state transitions change. Thus, having to compute two Q-values instead of one does not augment the complexity of the approach. In fact, Smith and Simmons (Smith & Simmons 2006) state that the additional time to compute a Bellman backup for two bounds, instead of one, is no more than 10%, which is also what we obtained. In particular, $L(s)$ is the lower bound of state s, while $U(s)$ is the upper bound of state s. Similarly, $Q_L(a,s)$ is the Q-value of the lower bound of action a in state s, while $Q_U(a,s)$ is the Q-value of the upper bound of action a in state s. Using these two bounds allow significantly reducing the action space A. Indeed, in Lines 11 and 12, if $Q_U(a,s) \leq L(s)$ then action a may be pruned from the action space of s. In Line 19, a state can be labeled as $solved$ if the difference between the lower and upper bounds is lower than ϵ. The next state in Line 22 has a fixed number of consumable resources available Res_c, determined in Line 21. In brief, PICKNEXTSTATE(res) selects a none-$solved$ state s reachable under the current policy which has the highest Bellman error ($|U(s) - L(s)|$).

As discussed by Singh and Cohn (Singh & Cohn 1998), this type of algorithm has a number of desirable anytime characteristics: if an action has to be picked in state s before the algorithm has converged (while multiple competitive actions remains), the action with the highest lower bound is picked. Since the upper bound for state s is known, it may be estimated how far the lower bound is from the optimal. If the difference between the lower and upper bound is too

Algorithm 1 The BOUNDED-RTDP algorithm. Adapted from (Bonet & Geffner 2003b) and (Singh & Cohn 1998).

```
1:  Function BOUNDED-RTDP($S$)
2:  returns a value function $V$
3:  repeat
4:     $s \leftarrow s_0$
5:     $visited \leftarrow null$
6:     repeat
7:        $visited.push(s)$
8:        for all $a \in A(s)$ do
9:           $Q_U(a,s) \leftarrow R(s) + \gamma \sum_{s' \in S} P_a(s'|s)U(s')$
10:          $Q_L(a,s) \leftarrow R(s) + \gamma \sum_{s' \in S} P_a(s'|s)L(s')$
             {where $L(s') \leftarrow h_L(s')$ and $U(s') \leftarrow h_U(s')$
             when $s'$ is not yet visited and $s'$ has $Res_c \setminus$
             $res(a)$ remaining consumable resources}
11:          if $Q_U(a,s) \leq L(s)$ then
12:             $A(s) \leftarrow A(s) \setminus res(a)$
13:          end if
14:       end for
15:       $L(s) \leftarrow \max_{a \in A(s)} Q_L(a,s)$
16:       $U(s) \leftarrow \max_{a \in A(s)} Q_U(a,s)$
17:       $\pi(s) \leftarrow \arg\max_{a \in A(s)} Q_U(a,s)$
18:       if $|U(s) - L(s)| < \epsilon$ then
19:          $s \leftarrow solved$
20:       end if
21:       $Res_c \leftarrow Res_c \setminus \{\pi(s)\}$
22:       $s \leftarrow s.$PICKNEXTSTATE($Res_c$)
23:    until $s$ is a goal
24: until $s_0$ is $solved$ or $|A(s)| = 1 \; \forall \; s \in S$ reachable
    from $s_0$
25: return $V$
```

high, one can choose to use another greedy algorithm of one's choice, which outputs a fast and near optimal solution. Furthermore, if a new task dynamically arrives in the environment, it can be accommodated be redefining the lower and upper bounds which exist at the time of its arrival. Singh and Cohn (Singh & Cohn 1998) proved that an algorithm that uses admissible lower and upper bounds to prune the action space is assured of converging to an optimal solution.

The next sections describe two separate methods to define $h_L(s)$ and $h_U(s)$. First of all, the method of Singh and Cohn (Singh & Cohn 1998) is briefly described. Then, our own method proposes tighter bounds, thus allowing a more effective pruning of the action space.

Singh and Cohn's Lower and Upper Bounds

Singh and Cohn (Singh & Cohn 1998) defined lower and upper bounds to prune the action space. Their approach is pretty straightforward. First of all, a value function is computed for all tasks to realize, using a standard RTDP approach. Then, using these *task*-value functions, a lower bound h_L, and upper bound h_U can be defined. In particular, $h_L(s) = \max_{ta \in Ta} V_{ta}(s_{ta})$, and $h_U(s) = \sum_{ta \in Ta} V_{ta}(s_{ta})$. The admissibility of these bounds has been proven by Singh and Cohn, such that, the upper bound always overestimates the optimal value of each state, while the lower bound always underestimates the optimal value of each state. To determine the optimal policy π, Singh and Cohn implemented an algorithm very similar to BOUNDED-RTDP, which uses the bounds to initialize $L(s)$ and $U(s)$. The only difference between BOUNDED-RTDP, and the RTDP version of Singh and Cohn is in the stopping criteria. Singh and Cohn proposed that the algorithm terminates when only one competitive action remains for each state, or when the range of all competitive actions for any state are bounded by an indifference parameter ϵ. BOUNDED-RTDP labels states for which $|U(s) - L(s)| < \epsilon$ as *solved* and the convergence is reached when s_0 is *solved* or when only one competitive action remains for each state. This stopping criteria is more effective since it is similar to the one used by LRTDP. In this paper, the bounds defined by Singh and Cohn and implemented using BOUNDED-RTDP define the SINGH-RTDP approach.

The next sections propose to tighten the bounds of SINGH-RTDP to permit a more effective pruning of the action space.

Reducing the Upper Bound

The upper bound of SINGH-RTDP includes actions which may not be possible to execute because of resource constraints. To consider only possible actions, the upper bound is now:

$$h_U(s) = \max_{a \in A(s)} \sum_{ta \in Ta} Q_{ta}(a_{ta}, s_{ta}) \tag{3}$$

where $Q_{ta}(a_{ta}, s_{ta})$ is the Q-value of task ta for state s_{ta}, and action a_{ta} computed using a standard LRTDP approach.

Theorem 1 *The upper bound of Equation 3 is admissible.*

Proof: The local resource constraints are satisfied because the upper bound is computed using all global possible actions a. However, $h_U(s)$ still overestimates $V^\star(s)$ because the global resource constraint is not enforced. Indeed, each task may use all consumable resources for its own purpose. Doing this produces a higher value for each task, than the one obtained when planning for all tasks globally with the shared limited resources. ■

Increasing the Lower Bound

The idea to increase the lower bound of SINGH-RTDP is to allocate the resources a priori among the tasks. When each task has its own set of resources, each task may be solved independently. The lower bound of state s is $h_L(s) = \sum_{ta \in Ta} Low_{ta}(s_{ta})$, where $Low_{ta}(s_{ta})$ is a value function for each task $ta \in Ta$, such that the resources have been allocated a priori. The allocation a priori of the resources is made using *marginal revenue*, which is a highly used concept in microeconomics (Pindyck & Rubinfeld 2000), and has recently been used for coordination of a Decentralized MDP (Beynier & Mouaddib 2006). In brief, marginal revenue is the extra revenue that an additional unit of product will bring to a firm. Thus, for a stochastic resource allocation problem, the marginal revenue of a resource is the additional expected value it involves. The marginal revenue of a resource res for a task ta in a state s_{ta} is defined as following:

$$mr_{ta}(s_{ta}) = \max_{a_{ta} \in A(s_{ta})} Q_{ta}(a_{ta}, s_{ta}) - \max_{a_{ta} \in A(s_{ta})} Q_{ta}(a_{ta} | res \notin a_{ta}, s_{ta}) \quad (4)$$

Algorithm 2 The marginal revenue lower bound algorithm.

```
1: Function MARGINAL-REVENUE-BOUND($S$)
2: returns a lower bound $Low_{Ta}$
3: for all $ta \in Ta$ do
4:     $V_{ta} \leftarrow$ LRTDP($S_{ta}$)
5:     $value_{ta} \leftarrow 0$
6: end for
7: $s \leftarrow s_0$
8: repeat
9:     $res \leftarrow$ Select a resource type $res \in Res$
10:    for all $ta \in Ta$ do
11:        $mr_{ta}(s_{ta}) \leftarrow V_{ta}(s_{ta}) - V_{ta}(s_{ta}(Res \setminus res))$
12:        $mrrv_{ta}(s_{ta}) \leftarrow mr_{ta}(s_{ta}) \times \frac{V_{ta}(s_{ta}) - value_{ta}}{R(s_{g_{ta}})}$
13:    end for
14:    $ta \leftarrow$ Task $ta \in Ta$ which maximize $mrrv_{ta}(s_{ta})$
15:    $Res_{ta} \leftarrow Res_{ta} \bigcup \{res\}$
16:    $value_{ta} \leftarrow value_{ta} + ((V_{ta}(s_{ta}) - value_{ta}) \times \frac{\max_{a_{ta} \in A(s_{ta}(res))} Q_{ta}(a_{ta}, s_{ta}(res))}{V_{ta}(s_{ta})})$
17: until all resource types $res \in Res$ are assigned
18: for all $ta \in Ta$ do
19:     $Low_{ta} \leftarrow$ LRTDP($S_{ta}$)
20: end for
21: return $Low_{Ta}$
```

The concept of marginal revenue of a resource is used in Algorithm 2 to allocate the resources a priori among the tasks which enables to define the lower bound value of a state. In Line 4 of the algorithm, a value function is computed for all tasks in the environment using a standard LRTDP approach. These value functions are computed considering that each task may use all available resources. The Line 5 initializes the $value_{ta}$ variable. This variable is the estimated value of each task $ta \in Ta$. In the beginning of the algorithm, no resources are allocated to a specific task, thus, the $value_{ta}$ variable is initialized to 0 for all $ta \in Ta$. Then, in Line 9, a resource type res (consumable or non-consumable) is selected to be allocated. Here, a domain expert may separate all available resources in many types or parts to be allocated. The resources are allocated in the order of its specialization. In other words, the more a resource is efficient on a small group of tasks, the more it is allocated early. Allocating the resources in this order improves the quality of the resulting lower bound. The Line 11 computes the marginal revenue of resource res for each task $ta \in Ta$. In Line 12, the marginal revenue is updated in function of the resources already allocated to each task. $R(s_{g_{ta}})$ is the reward to realize task ta. Thus, $\frac{V_{ta}(s_{ta}) - value_{ta}}{R(s_{g_{ta}})}$ is the residual expected value that remains to be achieved, knowing current allocation to task ta, and normalized by the reward of realizing the tasks. The marginal revenue is multiplied by this term to indicate that, the more a task has a high residual value, the more its marginal revenue is going to be high. Then, a task ta is selected in Line 14 with the highest marginal revenue, adjusted with residual value. In Line 15, the resource type res is allocated to the group of resources Res_{ta} of task ta. Afterwards, Line 16 recomputes $value_{ta}$. The first part of the equation to compute $value_{ta}$ represents the expected residual value for task ta. This term is multiplied by $\frac{\max_{a_{ta} \in A(s_{ta})} Q_{ta}(a_{ta}, s_{ta}(res))}{V_{ta}(s_{ta})}$, which is the ratio of the efficiency of resource type res. In other words, $value_{ta}$ is assigned to $value_{ta}$ + (*the residual value* $\times$ *the value ratio of resource type* res). All resource types are allocated in this manner until Res is empty. All consumable and non-consumable resource types are allocated to each task. When all resources are allocated, the lower bound components Low_{ta} of each task are computed in Line 19. When the global solution is computed, the lower bound is as follow:

$$h_L(s) = \max_{a \in A(s)} \sum_{ta \in Ta} Low_{ta}(s_{ta}) \quad (5)$$

Theorem 2 *The lower bound of Equation 5 is admissible.*

Proof: $Low_{ta}(s_{ta})$ is computed with the resource being shared. Summing the $Low_{ta}(s_{ta})$ value functions for each $ta \in Ta$ does not violates the local and global resource constraints. Indeed, as the resources are shared, the tasks cannot overuse them. Thus, $L(s)$ is a realizable policy. ■

The upper bound defined in Section , and the lower bound defined here, solved using Algorithm 1, determines the M(Marginal)R(Revenue)-RTDP approach.

Complexity of Computing the Bounds

For the upper bound, SINGH-RTDP and MR-RTDP compute a value function for each task. We consider $|S_{ta}|$ as the number of possible states for task ta. Also, $|S_{Ta}|$ is the number of possible joint states for all tasks $ta \in Ta$. Since $|S_{Ta}|$ is combinatorial with the number of tasks, thus $|S_{ta}| \ll |S_{Ta}|$. Indeed,

$$|S_{Ta}| = \mathcal{O}(|S_{ta}|^{|Ta|}) \quad (6)$$

When the number of tasks is high, the complexity of computing a value function for each task is negligible compared to computing a global value function for all tasks. The main difference in complexity between the SINGH-RTDP approach, and MR-RTDP is how the value function is used. The SINGH-RTDP approach simply sums the value function $V_{ta}(s_{ta})$ of each task ta to determine the upper bound of a state s. As for MR-RTDP, all global actions $a \in A(s)$ are computed to determine the maximal possible upper bound, considering local constraints of a state s. Thus, the complexity to determine the upper bound of a state is $\mathcal{O}(|A| \times |Ta|)$. The computation of all global actions is much more complex than simply summing the value functions, as in SINGH-RTDP. A standard Bellman backup, when computing the global solution sums $|S|$ for each $a \in A(s)$, thus has complexity $\mathcal{O}(|A| \times |S|)$. Since $|A| \times |Ta| \ll |A| \times |S|$, the computation time to determine the upper bound of a state, which is done one time for each visited state, is much less than the computation time required to compute a standard Bellman backup for a state, which is usually done many times for each state. Thus, the computation time of the upper bound is negligible.

The analysis of the complexity of Algorithm 2, used for MR-RTDP is now made. Line 11 is the part of the algorithm which induces complexity, compared to the SINGH-RTDP approach. This line has a complexity of $\mathcal{O}(|A_{ta}| \times |Ta| \times |Res|)$. It is important, here, to not divide the set of resources in too much types to be allocated, since it has a great incidence on the complexity to produce the lower bound. If the number of resource types is low, $|A_{ta}| \times |Ta| \times |Res|$ is a pretty straightforward calculus. The SINGH-RTDP and MR-RTDP approaches are compared and discussed in the next section.

Discussion and Experiments

Modelling a stochastic resource allocation problem using upper and lower bounds on the value function allows reducing the number of action to consider in each state. The domain of the experiments is a naval platform which must counter incoming missiles (i.e. tasks) by using its resources (i.e. weapons, movements). For the experiments, 100 randomly resource allocation problems were generated for each approach, and possible number of tasks. In our problem, $|S_{ta}| = 4$, thus each task can be in four distinct states. There are two types of states; firstly, states where actions modify the transition probabilities; and then, there are goal states. The state transitions are all stochastic because when a missile is in a given state, it may always transit in many possible states. There are also local and global resource constraint on the amount that may be used. For the local constraints, at most 1 resource of each type can be allocated to execute tasks on a single step. This constraint is also present on a real naval platform because of sensor and launchers constraints and engagement policies. Furthermore, for consumable resources, the total amount of available consumable resource is between 1 and 2 for each type. The global constraint is generated randomly at the start of a scenario for each consumable resource type. The number of resource type has been fixed to 5. The transition probabilities are also generated randomly. Each resource type has a probability to counter a missile between 35% and 55% depending on the state of the task. When a missile is not countered, it transits to another state, which may be preferred or not to the current state, where the most preferred state for a task is when it is countered. The effectiveness of each resource is modified randomly by $\pm 15\%$ at the start of a scenario.

For this problem a standard LRTDP approach has been implemented. A simple heuristic has been used where the value of an unvisited state is assigned as the value of a goal state such that all tasks are achieved. This way, the value of each unvisited state is assured to overestimate its real value since the value of achieving a task ta is the highest the planner may get for ta. Since this heuristic is pretty straightforward, the advantages of using better heuristics as in SINGH-RTDP and MR-RTDP are more evident. Nevertheless, even if the LRTDP approach uses a simple heuristic, still a huge part of the state space is not visited when computing the optimal policy.

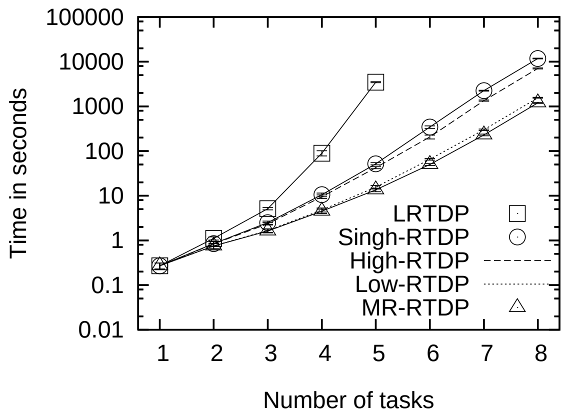

Figure 3: Computational efficiency of LRTDP, SINGH-RTDP, HIGH-RTDP (lower bound of Singh and Cohn and upper bound of Section), LOW-RTDP (upper bound of Singh and Cohn and lower bound of Section), and MR-RTDP.

The optimal LRTDP, SINGH-RTDP, and MR-RTDP approaches are compared in Figure 3. Two more versions have been added to the results. First of all, HIGH-RTDP uses the lower bound of Singh and Cohn (Singh & Cohn 1998) and the upper bound of Section). Then, LOW-RTDP uses the upper bound of Singh and Cohn (Singh & Cohn 1998) and the lower bound of Section . To compute the lower bound of LOW-RTDP and MR-RTDP, all available resources have to be separated in many types or parts to be allocated. For our problem, we allocated each resource of each type in the order of of its specialization like we said in Section .

In terms of experiments, notice that the LRTDP approach for resource allocation, which is computed on the joint action and state spaces of all agents, is much more complex. For instance, it takes an average of 3487 seconds to plan for an LRTDP approach with five tasks (see Figure 3). The SINGH-RTDP approach solves optimally the same type of problem in an average of 52.6 seconds and MR-RTDP in 12.2 seconds. Indeed, the time reduction is quite significant compared to LRTDP, which demonstrates the efficiency of using bounds to diminish the action space. The number of iterations required for convergence is significantly smaller for MR-RTDP and SINGH-RTDP than for LRTDP. Indeed, the more tight the bounds are, the faster these bounds converge to the optimal value.

On the figure results, we may also observe that the reduction in planning time of MR-RTDP compared to SINGH-RTDP is obtained mostly with the lower bound. Indeed, when the number of task to execute is high, the lower bounds by SINGH-RTDP takes the values of a single task. On the other hand, the lower bound of MR-RTDP takes into account the value of all task by using a heuristic to distribute the resource. Indeed, an optimal allocation is one where the resources are distributed in the best way to all tasks, and our lower bound heuristically does that.

Conclusion

The experiments have shown that MR-RTDP provides a potential solution to solve efficiently stochastic resource allocation problems. Indeed, the planning time of MR-RTDP is significantly lower than for LRTDP. While the theoretical complexity of MR-RTDP is higher than for SINGH-RTDP, its ability to produce a tight bound offsets this aspect, as shown in the experiments.

An interesting research avenue would be to experiment MR-RTDP with other heuristic search algorithms than LRTDP. HDP (Bonet & Geffner 2003a), and LAO* (Hansen & Zilberstein 2001) are both efficient heuristic search algorithms which could be implemented using our bounds. As a matter of fact, the bounds proposed in this paper can be used with any stochastic algorithm which solves a perfectly observable resource allocation problem. Also, Smith and Simmons (Smith & Simmons 2006), proposed an efficient trajectory of state updates to further speed up the convergence, when given upper and lower bounds, which may be used here to further speed up the convergence. Other future work would be to reduce the state space using Q-decomposition (Russell & Zimdars 2003). An important assumption of this method is that each agent should have its own independent reward function. As a consequence, for a resource allocation problem, there would be an agent for each task to achieve and still permit an optimal solution.

References

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1):81–138.

Beynier, A., and Mouaddib, A. I. 2006. An iterative algorithm for solving constrained decentralized markov decision processes. In *Proceeding of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

Bonet, B., and Geffner, H. 2003a. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*.

Bonet, B., and Geffner, H. 2003b. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceeding of the 13Th International Conference on Automated Planning & Scheduling (ICAPS-03)*, 12–21.

Hansen, E. A., and Zilberstein, S. 2001. LAO* : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129(1-2):35–62.

McMahan, H. B.; L., M.; and Gordon, G. J. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 569–576. New York, NY, USA: ACM Press.

Pindyck, R. S., and Rubinfeld, D. L. 2000. *Microeconomics*. Prentice Hall.

Russell, S. J., and Zimdars, A. 2003. Q-decomposition for reinforcement learning agents. In *ICML*, 656–663.

Singh, S., and Cohn, D. 1998. How to dynamically merge markov decision processes. In *Advances in neural information processing systems*, volume 10, 1057–1063. Cambridge, MA, USA: MIT Press.

Smith, T., and Simmons, R. 2006. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

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