R-FRTDP: A Real-Time DP Algorithm with Tight Bounds for a Stochastic Resource Allocation Problem

Camille Besse, Pierrick Plamondon, and Brahim Chaib-draa

Computer Science & Software Engineering Dept., Laval University,
Québec (Qc), Canada,
{besse,plamon,chaib}@damas.ift.ulaval.ca,
http://www.damas.ift.ulaval.ca/

Abstract. Resource allocation is a widely studied class of problems in Operation Research and Artificial Intelligence. Specially, constrained stochastic resource allocation problems, where the assignment of a constrained resource do not automatically imply the realization of the task. This kind of problems are generally addressed with Markov Decision Processes (MDPs). In this paper, we present efficient lower and upper bounds in the context of a constrained stochastic resource allocation problem for a heuristic search algorithm called Focused Real Time Dynamic Programming (FRTDP). Experiments show that this algorithm is relevant for this kind of problems and that the proposed tight bounds reduce the number of backups to perform comparatively to previous existing bounds.

1 Introduction

Resource Allocation problem is a widely studied class of problem in Operation Research and Artificial Intelligence. This class is known to be NP-complete [10]. Since resources are usually constrained, the allocation of resources to one task restricts the options available for other tasks. As the action space is exponential according to the number of resources, and as the state space is exponential according to number of resources and tasks, this type of problem with time constraints is 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) [1], LRTDP [4], FRTDP [9], HDP [3], LAO* [5] are all state-of-the-art heuristic search approaches in a stochastic environment. Because of its anytime quality, RTDP, introduced by Barto *et al.* [1] is an interesting approach since it updates states in trajectories from an initial state s_0 to a goal state s_g in a very efficient way.

Actually, RTDP is much more effective if the action space can be pruned of sub-optimal actions. To do this, McMahan *et al.* [6], Smith and Simmons [9], and Singh and Cohn [8] proposed solving a stochastic problem using a RTDP type heuristic search with upper and lower bounds on the value of states.

The bounds proposed by McMahan *et al.* [6] 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 the case of our resource allocation problem where rewards are only obtained when tasks are achieved, this approach is not applicable.

The FRTDP approach by Smith and Simmons [9] proposes an efficient trajectory of state updates to further speed up the convergence, given upper and lower bounds. This efficient trajectory of state updates are combined to the approach proposed in this paper as we focus on the definition of tight bounds for a constrained resource allocation problem.

On the other hand, the bounds proposed by Singh & Cohn [8] are suitable to our case, and extended in this paper using, in particular, the concepts of task criticality and feasibility to elaborate tight bounds. These bounds are implemented in the context of a FRTDP heuristic search approach.

Our bounds are compared theoretically and empirically to the bounds proposed by Singh & Cohn using the FRTDP approach. Indeed, when implementing FRTDP with our proposed upper and lower bounds, its convergence to the optimal policy is faster compared to the Singh & Cohn bounds using the same algorithm. 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.

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 $ta_2 = \{\text{clean the floor}\}$. 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 a time interval for which the task are going to be in this state. The resources may be constrained by the amount that may be used simultaneously. Furthermore, the higher is the number of resources allocated to realize a task, the higher is the expectation of realizing the task. 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.

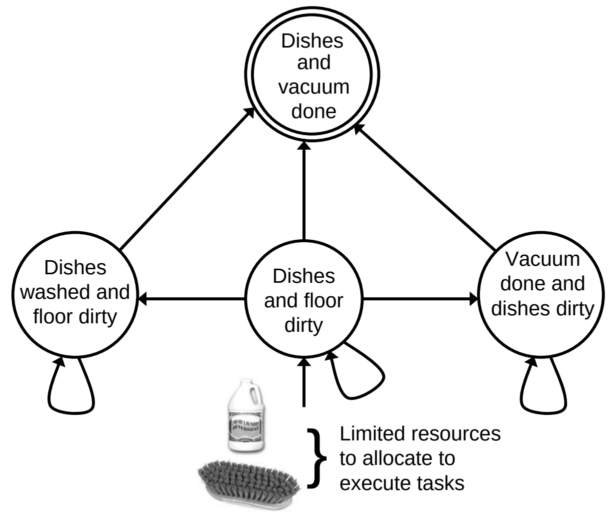
Dishes and vacuum done
Dishes washed and floor dirty
Dishes and floor dirty
Vacuum done and dishes dirty
Limited resources to allocate to execute tasks

Fig. 1. Task transition graph.

2 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.
- Ta is a finite set of tasks with $ta \in Ta$ to be executed.
- S is a finite set of states with $s \in S$. A state s is a tuple $\langle t_{start}, t_{end}, Ta, alloc \rangle$. In particular, t_{start} is the start time of the state, t_{end} is the end time of the state. $alloc$ is a set of allocations which are already in execution at time t_{start}. 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 allocations $a \in A(s)$ applicable in a state are the combination of all resource assignments that may be executed, according to the state s. In particular, a is simply an allocation of resources to the current tasks, and a_{ta} is the resource allocation to task ta. Each action has a start time t_{start} and an end time t_{end}. The possible actions are limited by the amount that may be used on a task at a particular time.
- 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 $R = [r_s] : \sum_{ta \in Ta} r_{s_{ta}} \leftarrow \Re_{s_{ta}} \times w_{ta}$. 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 $1 \times w_{ta}$ 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:

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

The end time t_{end} of s is set to the earliest ending time of an action in allocation ($alloc$) or to execute (a) in state s. The start time t_{start} of state s' is equal to time t_{end} of state s. Furthermore, one may compute the Q-Values $Q(a, s)$ of each state action pair using the following equation:

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

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 resource constraint $res(\pi(s)) \leq C_{s_{Ta}} \forall\ s \in S$, and $\forall\ res \in Res$, where $C_{s_{Ta}}$ is the resource constraint on tasks Ta. Heuristic search may reduce the complexity of a stochastic resource allocation problem by focussing on relevant states. To this end, Real-Time Dynamic Programming (RTDP) heuristic search is now introduced.

3 Real Time Dynamic Programming

Barto *et al.* [1] proposed Real Time Dynamic Programming (RTDP) (Algorithm 1) as an effective real-time heuristic search approach. RTDP 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 RTDP trial (TRIALRECURSE function) is the result of simulating the policy π, through the PICKNEXTSTATE function, while updating the upper bound values $s.U$ using a Bellman backup (Equation 1) over the states s that are visited. $h(s')$ is a heuristic which defines 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 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. The new set of tasks to accomplish is produced by the PICKNEXTSTATE function which randomly picks a none-*solved* state, containing a new set of tasks to realize, by executing the current policy.

It has been proven that RTDP, given an admissible initial heuristic on the value of states cannot be trapped in loops, and eventually yields optimal values [4].

Algorithm 1 RTDP

```
Function initNode($s$): {implicitly called
    the first time each state $s$ is touched}
1: $s.U \leftarrow h_U(s)$
Function RTDP($s_0$,$h_U$):
2: loop TRIALRECURSE($s_0$)
Function BACKUP($s$):
3: $s.U \leftarrow \max_a \mathrm{QU}(s,a)$
Function TRIALRECURSE($s$):
4: if $s \in \mathfrak{G}$ then return
5: $\pi(s) \leftarrow \arg\max_a \mathrm{QU}(s,a)$
6: $s' \leftarrow$ s.PICKNEXTSTATE()
7: TRIALRECURSE($s'$)
8: BACKUP($s$)
Function QU($s,a$):
9: return $R(s,a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T^a_{s,s'} s'.U$
```

3.1 Focused RTDP

FOCUSED RTDP (alg. 2) is an RTDP based algorithm proposed by Smith & Simmons [9]. As in RTDP, FRTDP's execution consists in trials that begin in a given initial state s_0 and then explore reachable states of the state space, selecting actions according to an upper bound. Once a final state is reached, it performs Bellman updates on the way back to s_0.

Unlike RTDP, FRTDP maintains also a lower bound and uses others criteria to select actions outcomes and to detect trial termination. The lower bound is used to establish the policy and it also contributes in the priority calculation of states to expand on the fringe of the search tree. Trial termination detection has been modified and improved by adding an adaptive maximum depth D in the search tree in order to avoid over-committing to long trials early on. In fact, the maximum depth D is updated by $k_D D$ each time the trial is not useful enough. This usefulness is represented by δW where δ measures how much the update changed the upper bound value of s and W the expected amount of time the current policy spends in s, adding up all possible paths from s_0 to s. Refer to the pseudo-code of Algorithm 2 and to Smith & Simmons' article [9] for details.

3.2 Singh & Cohn's Lower and Upper Bounds

Singh & Cohn [8] defined lower and upper bounds for a stochastic problem. Their approach is pretty straightforward. First of all, a value function is computed for

Algorithm 2 Focused RTDP

```
Function INITNODE($s$): {implicitly called the first time each state $s$ is touched}
 1: $(s.L, s.U) \leftarrow (h_L, h_U)$; $s.prio \leftarrow \Delta(s)$
Function FRTDP($s_0$, $\varepsilon$,$h_L$,$h_U$,$D_0$,$k_D$):
 2: $D \rightarrow D_0$
 3: while $s_0.U - s_0.L > \varepsilon$ do
 4:     $(q_p, n_p, q_c, n_c) \leftarrow (0, 0, 0, 0)$
 5:     TRIALRECURSE($s_0$,$W = 1$,$d = 0$)
 6:     if $(q_c/n_c) \geq (q_p/n_p)$ then
            $D \leftarrow k_D D$
 7: end while
Function TRIALRECURSE($s$, $W$, $d$):
 8: $(a^*, s^*, \delta) \rightarrow$ BACKUP($s$)
 9: TRACKUPDATEQUALITY($\delta W, d$)
10: if $\Delta(s) \leq 0$ or $d \geq D$ then return
11: TRIALRECURSE($s^*$,$\gamma T^{a^*}_{s,s^*} W$,$d + 1$)
12: BACKUP($s$)
Function TRIALUPDATEQUALITY($q$, $d$):
13: if $d > D/k_D$ then
        $(q_c, n_c) \leftarrow (q_c + q, n_c + 1)$
14: else $(q_p, n_p) \leftarrow (q_p + q, n_p + 1)$
Function BACKUP($s$):
15: $s.L \leftarrow \max_a \mathrm{QL}(s, a)$
16: $u \leftarrow \max_a \mathrm{QU}(s, a)$
17: $a^* \leftarrow \arg\max_a \mathrm{QU}(s, a)$
18: $\delta \leftarrow |s.U - u|$
19: $s.U \leftarrow u$
20: $p \leftarrow \max_{s' \in \mathcal{S}} \gamma T^{a^*}_{s,s'} s'.prio$
21: $s^* \leftarrow \arg\max_{s' \in \mathcal{S}} \gamma T^{a^*}_{s,s'} s'.prio$
22: $s.prio \leftarrow \min(\Delta(s), p)$
23: return $(a^*, s^*, \delta)$
Function $\Delta(s)$:
24: return $|s.U - s.L| - \varepsilon/2$
Function QL($s, a$):
25: return $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T^a_{s,s'} s'.L$
Function QU($s, a$):
26: return $R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \gamma T^a_{s,s'} s'.U$
```

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 & 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. In this paper, the bounds defined by Singh & Cohn and implemented using FRTDP define the SINGH-FRTDP approach.

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

3.3 Reducing the Upper Bound

In the next two sections, tight bounds are proposed for a stochastic resource allocation problem. The FRTDP approach initiated with these bounds defines the Resource FRTDP (R-FRTDP) approach.

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

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

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 FRTDP approach.

When the time is introduced into the problem, matching task states to the global state is not obvious. Indeed, the start time and end time of a global state are generally different of the start time and end time of the specific state of each task. This is caused by the fact that the end time of a state s is obtained according to the time of the first action to end when considering all tasks. On the other hand, the end time of a task state s_{ta} is obtained with the first action to end, when considering the current task ta only.

To match correctly a task state s_{ta} within a global state s, we find a task state for which:

$$t_{start_{ta}} \leq t_{start} < t_{end_{ta}} \tag{4}$$

where $t_{start_{ta}}$ and $t_{end_{ta}}$ are, respectively the starting and ending time of s_{ta}. Also, s_{ta} has to match the other characteristics of the task ta in the global state s.

Here, the best matching state for a task is found. It is a matching state since the start time of s_{ta} is less or equal than the start time of the global state s. Indeed, the possible actions in state s_{ta} are equal or greater than the possible actions in state s for task ta. Also, it is the *best* matching state because one and only one state for a task can satisfy Equation 4 and it is the state which has the start time the nearest possible of t_{start}, but for which t_{start_ta} is not greater than t_{start}.

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

Proof: The 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 future states of the tasks violates the resource constraint. 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. ■

Complexity of Computing the Upper Bound For the upper bound, SINGH-FRTDP and R-FRTDP 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,

$$|S_{Ta}| = \mathcal{O}(|S_{ta}|^{|Ta|}) \tag{5}$$

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-FRTDP approach, and R-FRTDP is how the value function is used. The SINGH-FRTDP 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 R-FRTDP, all global actions $a \in A(s)$ are computed to determine the maximal possible upper bound, considering the resource 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-FRTDP. 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.

Increasing the Lower Bound As time is integrated in this problem a first heuristic is to use a rule based reactive planner to assign resources as tasks income. Thus, the lower bound we develop assign resources to tasks according to their priority and remaining time while respecting constraints as described by algorithm 3. Practically, trials are made with action chosen by the reactive planner in order to evaluate its policy. Choice of next states of actions are made depending on the same criteria as FRTDP (see section 3.1 or [9]). An adaptive number of trial is also chosen, until the value gained between each trial is no more a fixed threshold. As a result, an approximation of a sub-optimal policy is calculated depending on time wanted to be spend on lower bound calculation.

Theorem 2 *The lower bound proposed in this section is admissible.*

Proof: This is a lower bound since the reactive policy is sub-optimal and resource constraints are checked by the algorithm 3. ■

4 Experimental Results

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, $|S_{ta}|$ was generally 7, thus each task can be in 6 distinct states. In particular, there were generally 5 possible states for a missile where actions can be performed to counter it. In other words, there are 5 possible reallocation for a missile until it is too late to execute it. The number of reallocation depends greatly on the speed of the missile and its time of appearance. For each task, there are two goal states; a state where the missile is realized, and a state where the missile has hit the ship. The state transitions are all stochastic because when a missile is in a given state, it may always transit in many possible states. The number of resource type has been fixed to 3, where each type has constraints on the amount that may be used at a time. In particular, at most 1 resource of any type can be allocated on a task on a particular time. This constraint is also present on a real naval platform because of sensor and launchers constraints and engagement policies.

Algorithm 3 h_L: Reactive algorithm [2]

```
1: Inputs: $Tasks$: Tasks list;
2: $Resources$: Resources list;
3: {Tasks pre-treatment:}
4: $Tasks \leftarrow$ PRIORITIZE($Tasks$)
5: $Resources \leftarrow$ ORDERBYEFFICIENCY($Resources$)
6: $T \leftarrow$ FIRST($Tasks$)
7: $R \leftarrow$ FIRST($Resources$)
8: while $Resources \neq \varnothing$ do
9:    if AVAILABLE($R$) and ASSIGNABLE($R$,$T$) then
10:       ASSIGN($R$,$T$)
11:       $Tasks \leftarrow Tasks \setminus \{T\}$
12:       $Resources \leftarrow Resources \setminus \{R\}$
13:       $T \leftarrow$ FIRST($Tasks$)
14:    else
15:       $T \leftarrow$ NEXT($Tasks$)
16:    end if
17:    $R \leftarrow$ FIRST($Resources$)
18: end while
19: return An allocation of all available resources
```

Each resource type has its specific range of effectiveness. In particular, the first resource type generally has 3 possible reallocations before the threat is too near of the ship to be able to use this resource. The effectiveness of this resource varies between 90% and 95%. These variations in effectiveness depends greatly on the type of threat and its range from the ship. The second resource type has a part of its effectiveness range which overlaps with the first resource type and has 2 possible reallocations on a threat. The effectiveness of this resource is between 35% and 50%. The last resource type can be used when the threat is very near the ship and has no possible reallocation to it. The effectiveness of this resource varies between 65% and 85%. The bulk of the planning work is on made on the decisions of how to allocate the first resource type which has a big range and a high probability of effectiveness.

The optimal R-FRTDP, SINGH-FRTDP, UP-FRTDP and LOW-FRTDP approaches are compared in Figure 2. Two more versions have been added to the results. First of all, UP-FRTDP uses the lower bound of Singh & Cohn [8] and the upper bound of Section 3.3. Then, LOW-RTDP uses the upper bound of Singh & Cohn [8] and the lower bound of Section 3.3.

In terms of experiments, notice that the SINGH-FRTDP approach for resource allocation, which use loose bounds requires the most time for convergence. For instance, it takes an average of 26.6 seconds to plan for an SINGH-FRTDP approach with eight tasks (see Figure 2). The R-FRTDP approach solves optimally the same type of problem in an average of 3.23 seconds. This is a very significant improvement. The number of iterations required for convergence is significantly

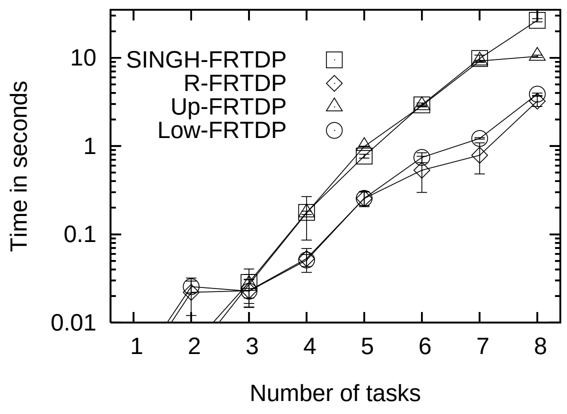

Fig. 2. Computational efficiency of R-FRTDP, SINGH-FRTDP, UP-FRTDP (Lower bound of Singh & Cohn and upper bound of Section 3.3), LOW-FRTDP (upper bound of Singh & Cohn and lower bound of Section 3.3).

smaller for R-FRTDP SINGH-FRTDP. 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 R-FRTDP compared to SINGH-FRTDP is obtained mostly with the lower bound. Indeed, when the number of task to execute is high, the lower bounds by SINGH-FRTDP takes the values of a single task. On the other hand, the lower bound of R-FRTDP 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.

5 Conclusion

The experiments have shown that R-FRTDP provides a potential solution to solve efficiently stochastic resource allocation problems. Indeed, the planning time of R-FRTDP is significantly lower than for FRTDP with no initial heuristic or with the Singh & Cohn [8] heuristic. While the theoretical complexity of R-FRTDP is higher than for SINGH-FRTDP, its ability to produce a tight bound offsets this aspect, as shown in the experiments.

An interesting research avenue would be to experiment R-FRTDP with other heuristic search algorithms than FRTDP. HDP [3], and LAO* [5] 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.

Furthermore, our approach could be improved by considering the probability of new tasks coming in the environment and reserving resources for them as done by Mercier & Van Hentenryck [7]. This would permit to have a more effective model and thus a better allocation.

References

1. A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.
2. D. Blodgett, P. Plamondon, B. Chaib-draa, P. Kropf, and E. Boss. A method to optimize ship maneuvers for the coordination of hardkill and softkill weapons within a frigate. In *7^{th} International Command and Control Research and Technology Symposium (7^{th}ICCRTS)*, Quebec, QC, September 2002.
3. B. Bonet and H. Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, August 2003.
4. B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceeding of the 13Th International Conference on Automated Planning & Scheduling (ICAPS-03)*, pages 12–21, Trento, Italy, 2003.
5. E. A. Hansen and S. Zilberstein. LAO* : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.
6. H. B. McMahan, M. L., and G. J. Gordon. 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*, pages 569–576, New York, NY, USA, 2005. ACM Press.
7. L. Mercier and P. V. Hentenryck. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.
8. S. Singh and D. Cohn. How to dynamically merge markov decision processes. In *Advances in neural information processing systems*, volume 10, pages 1057–1063, Cambridge, MA, USA, 1998. MIT Press.
9. T. Smith and R. Simmons. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, USA, 2006.
10. W. Zhang. Modeling and solving a resource allocation problem with soft constraint techniques. Technical report: WUCS-2002-13, Washington University, Saint-Louis, Missouri, 2002.