12TH ICCRTS “Adapting C2 to the 21st Century”

Modeling and Simulation
C2 Concepts, Theory, and Policy
Network-Centric Experimentation and Applications

A Real-Time Dynamic Programming Approach for the Resource Allocation of a Frigate

Pierrick Plamondon*, Brahim Chaib-draa
Laval University Québec, PQ, Canada, G1K 7P4
(418) 656-2131 x3226
Computer Science Department
{plamon; chaib}@damas.ift.ulaval.ca

Abder Rezak Benaskeur
Defence R&D Canada — Valcartier
2459 Pie-XI Blvd. North, Val-Bélair PQ, Canada, G3J 1X5
(418) 844-4000 x4478
Decision Support Systems Section
abderrezak.benaskeur@drdc-rddc.gc.ca

* Point of contact and student author

Abstract

This paper contributes to solve effectively stochastic resource allocation problems known to be NP-Complete. To address this complex resource management problem the Labeled Real-Time Dynamic Programming (LRTDP) approaches is applied in an effective way. LRTDP concentrates the planning on significant states of the environment only, as the search is guided by an initial heuristic. As demonstrated by the experiments, LRTDP permits to obtain a very high survivability for the frigate, when compared to two other well known techniques.

1 Introduction

The Combat System of a typical Frigate includes above water warfare (AWW) weapon systems for hardkill. Increasing complexity in threat technology, and increasing speed and diversity in open-ocean and littoral threat scenarios makes efficient and effective planning for weapons resources more and more difficult. To counter these problems, research is ongoing to design and implement resource management decision aids, based on intelligent agent technology to perform AWW hardkill resource allocation scheduling for a Frigate.

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 problem of large Markov Decision Processes (MDPs) is based on heuristic search where many algorithms have been developed recently. The idea of heuristic search is to start from the initial state, and given an admissible heuristic for the value of unvisited states, a huge part of the state space may be omitted from the search. For instance Real-Time Dynamic Programming (RTDP) by Barto *et al.* [1995], LRTDP by Bonet and Geffner [2003b], HDP by Bonet and Geffner [2003a], and LAO* by Hansen and Zilberstein [2001] are all state of the art heuristic search approaches used in a stochastic environment. Because of its anytime quality, an interesting approach is RTDP introduced by Barto *et al.* [1995] which updates states in trajectories from an initial state s_0 to a goal state s_g in a very efficient manner. Then, Bonet and Geffner [2003b] proposed a labelling procedure to accelerate the convergence of RTDP in their L(Labeled)RTDP algorithm. This paper proposes a model of LRTDP for the resource allocation of a Frigate.

We have compared our LRTDP implementation to a tabu search by Blodgett *et al.* [2003] and a reflex approach through the Ship Air Defence Model (SADM[1]) simulator in Section 3. The implementation of tabu search by Blodgett et al. has been made on a similar problem of resource allocation for a Frigate. The reflex approach is a rule based algorithm which engages the top two priority threats whenever it is possible. The problem is now modelled.

2 Problem Formulation

Our problem of interest is military naval operations which are known to be very complex. In this context, a ship's Commanding Officer needs to set his resources to maximum efficiency in real-time where he is in face to multi-threats situations. An efficient resource allocation can increase the chance of survival of the ship. In the case of Above-Water Warfare (AWW), the list of main operations are as follows :

- *Threat detection*: Based on data from several sensors.
- *Resource allocation*: Resources are assigned to engage each threat.
- *Engagement control*: The process by which decisions in the two preceding steps are executed in real-time.

Here, the focuss is on the *Resource allocation* and *engagement control* processes. To this end, the *threat detection* is considered as a black box. Not working on threat detection reduces the large volume of data that needs to be processed, which helps reducing the system's complexity to focuss on the resource allocation and execution parts.

The AWW hardkill weapons are weapons that are directed to intercept a threat and actively destroy it through direct impact or explosive detonation in the proximity of the threat. The range of different types of hardkill weapons varies, and the effectiveness of these weapons depends on a variety of factors, like distance to the threat, type of threat, speed of the threat, environment, etc. The AWW hardkill weapons for a typical Frigate include surface-to air missiles (SAMs) that have the greatest range, an intermediate range Gun, and a Close-In Weapons System (CIWS) that is a short-range, rapid-fire gun. The

[1]https://www.sadm.biz

Gun has a blind zone of ± 35 deg at the back of the Frigate. Closely allied to these weapons are two Separate Tracking and Illuminating Radars (STIRs) that are used to guide a SAM to a threat, and to point the Gun. There is one STIR to the front and one STIR to the back of the ship. In his case, both stirs can be used simultaneously at ± 30 deg to both sides of the Frigate. In all other areas, only one STIR can be used. The CIWS has its own pointing radar which has a blind zone of ± 15 deg to the front of the ship.

2.1 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. This paper consider the SAM and the Gun as possible resources.

- Ta is a finite set of threats with $ta \in Ta$ to be countered.

- 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 there forever. A goal state could be when all threats are countered or when the Frigate has sunk.

- A is a finite set of actions (or assignments). The allocation $a \in A(s)$ 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 threats, and a_{ta} is the resource allocation to threat ta. The possible actions are limited by the STIRs and blind zones constraints.

- Transition probabilities $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$.
- State rewards $R = [r_s] : \sum\limits_{ta \in Ta} r_{s_{ta}} \leftarrow \Re_{s_{ta}}$. The relative reward of the state of a threat $r_{s_{ta}}$ is the product of a real number $\Re_{s_{ta}}$. 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:

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

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 $V^{\star}(s) = \max\limits_{a \in A(s)} Q(a,s)$. The policy is subjected to the local resource constraints $\{\pi(s)\} \le L_{res} \forall\ s \in S$, and $\forall\ res \in Res$. Heuristic search may reduce the complexity of a stochastic resource allocation problem by focussing on relevant states. To this end, the Labeled Real-Time Dynamic Programming (LRTDP) heuristic search algorithm is now introduced.

2.2 LRTDP

Bonet and Geffner [2003b] proposed LRTDP (Algorithm 2.1) as an improvement to RTDP by Barto *et al.* [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

Algorithm 2.1 The LRTDP algorithm by Bonet and Geffner [2003b].

```
1: Function LRTDP($S$)
2: returns a value function $V$
3: repeat
4:    $s \leftarrow s_0$
5:    $visited \leftarrow null$
6:    repeat
7:       $visited.push(s)$
8:       $V(s) \leftarrow R(s) + \max_{a \in A(s)} \gamma \sum_{s' \in S} P_a(s'|s)V(s')$
         {where $V(s') = h(s')$ when $s'$ is not yet visited}
9:       $s \leftarrow s$.PICKNEXTSTATE()
10:   until $s$ is a goal
11:   while $visited \neq null$ do
12:      $s \leftarrow visited.pop()$
13:      if $\neg$ CHECKSOLVED($s, \epsilon$) then
14:         break
15:      end if
16:   end while
17: until $s_0$ is solved
18: return $V$
```

trial (Line 6 to 10) is the result of simulating the policy π, through the PICKNEXTSTATE(Res_c) function, 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 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. Then, the new set of tasks to accomplish is produced in Line 9. In brief, the PICKNEXTSTATE() function randomly picks a none-*solved* state, containing a new set of tasks to realize, by executing the current policy.

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 as proved by Bonet and Geffner [2003b]. The convergence is accomplished by means of a labelling procedure called CHECKSOLVED(s, ϵ) (Line 13 of the algorithm). 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. The next section describes how the LRTDP algorithm in the context of resource allocation for a Frigate.

2.3 LRTDP for Resource Allocation

The LRTDP-RES function (Algorithm 2.2) specializes LRTDP to our resource allocation problem. This algorithm is called whenever a new threat is perceived by the Electronic Support Measure (ESM) radar. Lines 3 and 4 of the Algorithm set the start time t_start and allocation *alloc* of the initial state s_0. Afterwards, if the the action set of state s is empty, it is generated in Line 11 by the GENERATEACTION function (Algorithm 2.3).

2.4 Complexity of lrtdp for aww

To measure the complexity of the QDEC-LRTDP algorithm for resource allocation, a comparison with the LRTDP algorithm is made for a state value update. Indeed, since both algorithms have the same expectation of having the same behavior, comparing both algorithms on a single value update seems fair. A Bellman backup iteration of a standard LRTDP approach for resource allocation has the following complexity:

$$\mathcal{O}(|A| \times |S_{Ta}|) \tag{3}$$

where $|S_{Ta}| = |S|$ is the number of joint states for the tasks, and $|A|$ is the number of joint actions. LRTDP is compared with other approaches and discussed in the next section.

3 Discussion and Experiments

The experiments have been performed on the Ship Air Defence Model (SADM[2]) simulator. We have compared our approach to a tabu search by Blodgett *et al.* [2003] and a reflex approach. In a tabu search , an initial policy

[2]https://www.sadm.biz

Algorithm 2.2 The LRTDP-RES algorithm for resource allocation.

```
1: **Function** LRTDP-RES($S$)
2: **returns** a value function $V$
3: $s_0.t_start$ is the current time of the simulation
4: $s_0.alloc$ are the actions which are currently in execution
5: **repeat**
6:    $s \leftarrow s_0$
7:    $visited \leftarrow null$
8:    **repeat**
9:       $visited.push(s)$
10:      **if** $A(s) = NULL$ **then**
11:         $A(s) \leftarrow$GENERATEACTION($s$)
12:         $s.t_{end} \leftarrow$FIRSTKILLTIME($A(s)$)
13:      **end if**
14:      $V(s) \leftarrow R(s) + \max_{a \in A(s)} \gamma \sum_{s' \in S} P_a(s'|s)V(s')$
         $\{$where $V(s') = h(s')$ when $s'$ is not yet visited$\}$
15:      $s \leftarrow s.$PICKNEXTSTATE()
16:   **until** $s$ is a goal
17:   **while** $visited \neq null$ **do**
18:      $s \leftarrow visited.pop()$
19:      **if** $\neg$ CHECKSOLVED($s, \epsilon$) **then**
20:         break
21:      **end if**
22:   **end while**
23: **until** $s_0$ is $solved$
24: **return** $V$
```

is improved through the removal or addition of defence actions, followed by update operations aimed at maintaining the consistency of the plan. It is based on an iterative neighborhood search method where modifications to the current solution that degrade the solution value are admissible. The latter move allows the method to escape from bad local optima (as opposed to a pure local search approach). To avoid cycling, a short-term memory, known as the tabu list, stores previously visited solutions or components of previously visited solutions. It is then forbidden or tabu to come back to these solutions for a certain number of iterations.

Algorithm 2.3 The LRTDP-RES algorithm for resource allocation.

```
1: Function GENERATEACTION($s$)
2: returns a set of possible actions.
   {here we determine possible actions against each threats}
3: $action \leftarrow NULL$
4: for all $ta \in S$ do
5:   for all $res \in Res$ do
6:     if !(res is a Gun $\wedge$ ISBLINDZONEGUN($ta$) = true) then
7:       determine the interception $range$ and $time$ of $res$ with $ta$
8:       if ISINRANGE(FIT($res$), LIT($res$) then
9:         $pk \leftarrow getPk(ta, res)$
10:        if $STIR_A(ta) = true$ then
11:          $action$.INSERT($stirA, ta, pk, res, time$)
12:        end if
13:        if $STIR_B(ta) = true$ then
14:          $action$.INSERT($stirB, ta, pk, res, time$)
15:        end if
16:      end if
17:    end if
18:  end for
19: end for
    {here we determine the possible joint actions}
20: for all $a_1 \in |action + 1|$ do
21:   $joint \leftarrow a_1$
22:   for all $a_2 \in |action + 1|$ do
23:     $joint \leftarrow joint \cup a_2$
24:     if VERIFYSTIR($a_2, s.alloc$) = true then
25:       $A(s) \leftarrow A(s) \cup joint$
26:     end if
27:   end for
28: end for
    return $joint$
```

The reflex approach simply launches a sam, or a gun if it is not possible to fire a sam to the two nearest threats to the Frigate whenever they are not engaged.

4 Conclusion

The experiments have shown that LRTDP provides a potential solution to solve efficiently stochastic resource allocation problems.

Other future work of LRTDP would be to reduce the action space. In particular, the action space can be reduced by merging MDPs using a lower and higher bound on the value of states as done by Singh and Cohn [1998] and McMahan *et al.* [2005]. Using these bounds, if an action has its upper bound lower than the lower bound of a state, it can be pruned from the action space for this state. In the case of LRTDP, this pruning would enable omitting a large part of the action space when computing the maximal global Q-value of a state.

References

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.

Dale E. Blodgett, Michel Gendreau, Fran&#231;ois Guertin, Jean-Yves Potvin, and Ren&#233; S&#233;guin. A tabu search heuristic for resource management in naval warfare. *Journal of Heuristics*, 9(2):145–169, 2003.

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.

B. Bonet and H. Geffner. Labeled LRTDP approach: 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.

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

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.

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.

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