A Q-decomposition LRTDP Approach to Resource Allocation

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

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

Abstract

*This paper contributes to solve effectively stochastic resource allocation problems known to be NP-Complete. To address this complex resource management problem, the merging of two approaches is made: The Q-decomposition model, which coordinates reward separated agents through an arbitrator, and the Labeled Real-Time Dynamic Programming (*LRTDP*) approaches are adapted in an effective way. The Q-decomposition permits to reduce the set of states to consider, while* LRTDP *concentrates the planning on significant states only. As demonstrated by the experiments, combining these two distinct approaches permits to further reduce the planning time to obtain the optimal solution of a resource allocation problem.*

1 Introduction

This paper aims to contribute to solve complex stochastic resource allocation problems. In general, resource allocation problems are known to be NP-Complete [6]. 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.

Another interesting approach is that of heuristic search where many algorithms [1] [2] 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. Because of its anytime quality, an interesting approach is LRTDP introduced by Bonet and Geffner [1] which updates states in trajectories from an initial state s_0 to a goal state s_g in a very efficient manner. Then, authors proposed a labelling procedure to guarantee accelerate the convergence of LRTDP.

In this paper, a merging of the Q-decomposition adapted from Russell and Zimdars [4] with the LRTDP approach is presented. The original Q-decomposition reinforcement learning coordination process determines the Q-values which maximize the sum of all combinations of possible agent Q-value for each visited state. An important assumption of this method is that each agent should have its own separate reward function. As a consequence, for a resource allocation problem, there would be an agent for each task to achieve. The problem is now modelled.

2. Resource Allocation using MDPs

In our problem, the transition function and the reward function are both known. Thus, a Markov Decision Process (MDP) framework is used to model our stochastic resource allocation problem.

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:

- $Res = \langle res_1, ..., res_m \rangle$ is a finite set of resource types available for a planning process. Each resource type has 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.
- 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 $\langle Ta, \langle res_1, ..., res_m \rangle \rangle$, which represents the particular state s_{ta}, which is the characteristic of each unaccomplished task $ta \in Ta$ in the environment, and the available 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 there 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. In particular, 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 $R = [r_s] : \sum_{ta \in Ta} r_{s_{ta}} \leftarrow \Re_{s_{ta}} \times w_{ta}$. 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}.
- 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 resources in state s' are $Res \setminus res(a)$, where $res(a)$ is the resources used by action a. Indeed, since an action a is a resource assignment, $Res \setminus res(a)$ 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:

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

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 $\{\pi(s)\} \leq L_{res} \forall\ s \in S$, and $\forall\ res \in Res$. and global resource constraint is $res(tra) \leq G_{res} \forall\ tra \in TRA$,and $\forall\ res \in Res$ where $res(tra)$ is a function mapping the resources used by trajectory $tra \in TRA$. Here, a system trajectory $tra \in TRA$ is a possible sequence of state-action pairs, until a goal state is reached under the optimal policy π. Since resource allocation in a stochastic environment is NP-Complete, heuristics should be employed. Q-decomposition which decomposes a planning problem to many agents to reduce the computational complexity associated to the state and/or action spaces is now introduced.

2.1. Q-decomposition for Resource Allocation

Russell and Zimdars [4] proposed Q-decomposition in the context of reinforcement learning. The primary assumption underlying Q-decomposition is that the overall reward function R can be additively decomposed into separate rewards R_i for each distinct agent $i \in Ag$, where $|Ag|$ is the number of agents. That is, $R = \sum_{i \in Ag} R_i$. It requires each agent to compute a value, from its perspective, for every action. To coordinate with each other, each agent i reports its action values $Q_i(a_i, s_i)$ for each state $s_i \in S_i$ to an arbitrator at each learning iteration. The arbitrator then chooses an action maximizing the sum of the agent Q-values for each global state $s \in S$. The next time state s is updated, an agent i considers the value as its respective contribution, or Q-value, to the global maximal Q-value. That is, $Q_i(a_i, s_i)$ is the value of a state such that it maximizes $\max_{a \in A(s)} \sum_{i \in Ag} Q_i(a_i, s_i)$. The fact that the agents use a determined Q-value as the value of a state is an extension of the Sarsa on-policy algorithm [3] to Q-decomposition. Russell and Zimdars called this approach local Sarsa. In this way, an ideal compromise can be found for the agents to reach a global optimum.

2.2. Q-decomposition LRTDP

Bonet and Geffner [1] proposed the optimal Labeled Real-Time Dynamic Programming LRTDP heuristic search algorithm. LRTDP is a simple algorithm that involves a sequence of trial runs, each starting in the initial state s_0 and ending in a goal state s_g. 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 admissible heuristic which define an initial value for state s. The ϵ convergence on the Bellman error of LRTDP is accomplished by means of a labelling procedure called CHECKSOLVED(s, ϵ).

We now show how to merge efficiently Q-decomposition and LRTDP to produce an optimal decomposition heuristic search algorithm. The method is presented in Algorithm 1. The modifications from the original LRTDP algorithm are as follows. The global Q-value of an action a in a state s is computed in Lines 9 to 19. In Line 12, each agent managing a task computes its respective Q-value. An agent i uses as the value of a possible state transition s' the Q-value for this agent which determines the maximal global Q-value for state s' as in the original Q-decomposition approach. In brief, for each visited states $s \in S$, each agent

computes its respective Q-values with respect to the global state s. So the state space for an agent is the same as the state space for LRTDP. The gain in complexity to use Q-decomposition resides in the $\sum_{s'_i \in S_i} P_{a_i}(s'_i|s,a)$ part of the equation. An agent considers as a possible state transition only the possible states of the task it manages. Since the number of states is exponential with the number of tasks, using Q-decomposition should reduce the planning time significantly. The number of resources in a possible state transition s'_i is $Res \setminus res(a)$. Indeed, even if an agent i plans only for its task with the part a_i of the global action a, it has to consider the resources used by the other agents. The less the number of resources is available, the less the value of a state is expected to be, so the global resource consumption has to be taken into account. Also, the state transition P_{a_i} for an agent considers the global action a because it is possible that the action executed on another task ta' influence the task ta managed by agent i.

The global Q-value is the sum of the Q-values produced by each agent managing each task as shown in Line 13. Lines 15 to 18 simply allocate the current value and policy with respect to the highest global Q-value. Then, Lines 20 and 21 generate the next state to compute a value. The notion of arbitrator is integrated within the Q-DEC-LRTDP algorithm for efficiency reasons. Indeed, since the action space has to be taken as a whole, the role of the arbitrator can be incorporated easily within the Q-DEC-LRTDP algorithm.

The behavior of Q-DEC-LRTDP is now discussed and we start it by proving the optimality of Q-DEC-LRTDP.This proof requires two steps. First of all, the fact that the value of a state in a given trajectory for Q-DEC-LRTDP is updated in the same manner as with the original LRTDP is shown. Then, the convergence of Q-DEC-LRTDP is proved.

Lemma 2.1 *A state in a trajectory for* Q-DEC-LRTDP *is updated in the same manner as for* LRTDP.

Proof: The following equation is used in Q-DEC-LRTDP to compute the value of a state:

$$V(s) = \sum_{i \in Ag} R_i(s) + \max_{a \in A(s)} \gamma \sum_{s'_i \in S'_i} P_{a_i}(s'_i|s,a) Q_i(a', s') \tag{3}$$

Since the reward can be additively decomposed for each task, Equation 3 may be rewritten as:

$$V(s) = R(s) + \max_{a \in A(s)} \sum_{i \in Ag} \gamma \sum_{s'_i \in S'_i} P_{a_i}(s'_i|s,a) Q_i(a', s') \tag{4}$$

Since $Q(a,s) = \sum_{i \in Ag} Q_i(a,s)$ when the transition probability of each task considers the actions performed on other tasks, Equation 4 may be rewritten as:

$$V(s) = R(s) + \max_{a \in A(s)} \gamma \sum_{s' \in S'} P_a(s'|s) Q(a', s') \tag{5}$$

where $Q(a', s')$ is the maximal Q-value for state s'. Indeed, since the arbitrator determines the maximal Q-value for a state, $Q(a', s') = V(s')$. Since Equation 5 is the same as a Bellman backup, and $Q(a', s')$ is the same as $V(s)$, a state in a trajectory for Q-DEC-LRTDP is updated in the same manner as for LRTDP. ■

Algorithm 1 The Q-decomposition LRTDP algorithm.

```
1:  Function Q-DEC-LRTDP(S)
2:  returns a value function V
3:  repeat
4:    s ← s_0
5:    visited ← null
6:    repeat
7:      visited.push(s)
8:      V(s) ← 0
9:      for all a ∈ A(s) do
10:       Q(a,s) ← 0
11:       for all i ∈ Ag do
12:         Q_i(a,s) ← R_i(s)+
                       γ Σ_{s'_i ∈ S_i} P_{a_i}(s'_i|s,a) Q_i(a',s')
            {where Q_i(a',s') = h_i(s') when s' is not yet
            visited, and s' has Res \ res(a) remaining
            resources}
13:         Q(a,s) ← Q(a,s) + Q_i(a,s)
14:       end for
15:       if Q(a,s) > V(s) then
16:         V(s) ← Q(a,s)
17:         π(s) ← a
18:       end if
19:     end for
20:     Res ← Res \ {π(s)}
21:     s ← s.PICKNEXTSTATE(Res)
22:   until s is a goal
23:   while visited ≠ null do
24:     s ← visited.pop()
25:     if ¬ CHECKSOLVED(s, ε) then
26:       break
27:     end if
28:   end while
29: until s_0 is solved
30: return V
```

Lemma 2.2 *The* Q-DEC-LRTDP *algorithm converges.*

Proof: It has been proven that the value of a state is updated similarly as in the centralized LRTDP algorithm. Also, the

newly visited state s' from the current state s is produced using the global action a, as for LRTDP. Since the states have the same value and the state transition has the same behavior as LRTDP, the convergence properties of Q-DEC-LRTDP are the same as the one of LRTDP. ■

Theorem 2.1 *The* Q-DEC-LRTDP *algorithm is optimal.*

Proof: With lemma 2.1 and 2.2 being true, the result follows. Indeed, the initial s which is updated optimally, is guaranteed to converge in the same manner as LRTDP. ■

3. Discussion and Experiments

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. There are three types of states; firstly, transitional states where no action is possible to modify the transition probabilities; then, action states, where actions modify the transition probabilities; finally, there are final states. The state transitions are all stochastic because when a missile is in a given state, it may always transit in many possible states.

For this problem, an admissible heuristic has to be formulated. 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 of overestimating its real value since the value of achieving a task ta is the highest the planner may get for ta. No other heuristic has been implemented since the objective here is only to compare Q-DEC-LRTDP and LRTDP in the same settings. However, even when using this simple heuristic, a huge part of the state space is still not visited when computing the optimal policy.

The optimal Q-DEC-LRTDP and LRTDP approaches are compared in Figure 1. 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 1). The Q-DEC-LRTDP approach solves optimally the same type of problem in an average of 6.58 seconds. Indeed, the time reduction is quite significant compared to LRTDP, which demonstrates the efficiency of Q-DEC-LRTDP to decompose the state space.

4. Conclusion

The experiments have shown that Q-DEC-LRTDP provides a potential solution to solve efficiently stochastic resource allocation problems. Indeed, the planning time of

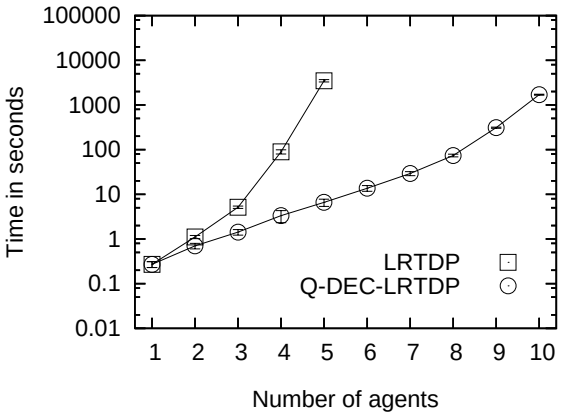

Figure 1. Computational efficiency of LRTDP **and** Q-DEC-LRTDP**.**

Q-DEC-LRTDP is significantly lower than for LRTDP, as suggested by the theoretical complexities. While the discussion in this paper has focussed on resource allocation problems, Q-DEC-LRTDP may be used in any type of problem where the overall reward function can be additively decomposed into separate rewards for distinct agents. Indeed, the general principle of the original Q-decomposition approach has to be applicable to justify using Q-DEC-LRTDP.

An interesting research avenue would be to experiment Q-decomposition with other heuristic search algorithms than LRTDP. For instance, the efficiency of LAO* may be greatly improved if combined with Q-decomposition.

References

[1] 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.

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

[3] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report CUED/FINFENG/TR 166, Cambridge University Engineering Department, 1994.

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

[5] 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.

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