

Multiple-Goal Reinforcement Learning with Modular Sarsa(0)

Nathan Sprague Dana Ballard

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 798

April 2004

Abstract

We present a new algorithm, GM-Sarsa(0), for finding approximate solutions to multiple-goal reinforcement learning problems that are modeled as composite Markov decision processes. According to our formulation different sub-goals are modeled as MDPs that are coupled by the requirement that they share actions. Existing reinforcement learning algorithms address similar problem formulations by first finding optimal policies for the component MDPs, and then merging these into a policy for the composite task. The problem with such methods is that policies that are optimized separately may or may not perform well when they are merged into a composite solution. Instead of searching for optimal policies for the component MDPs in isolation, our approach finds good policies in the context of the composite task.

This material is based upon work supported by a grant from the Department of Education under grant number P200A000306, a grant from the National Institutes of Health under grant number 5P41RR09283 and a grant from the National Science Foundation under grant number E1A-0080124.

1 Introduction

Traditional reinforcement learning algorithms can successfully solve small, single-goal tasks. The main challenge in the area of reinforcement learning is scaling up to larger and more complex problems. The scaling problem takes a number of forms. We may have a problem that has a very large state space, a problem that is best described as a set of hierarchically organized goals and subgoals, or a problem that requires the learning agent to address several tasks at once. It is the last form of scaling that this paper is concerned with.

The naive approach to learning to solve composite tasks is to create a state space that includes all of the information that is relevant to each sub-task. The agent would then learn in this joint space, receiving reward when any of the sub-goals are accomplished. The problem with this approach is that it suffers from the curse of dimensionality; as additional state dimensions are added for each new sub-task, the size of the joint state space grows exponentially.

A more promising approach to this sort of multiple-goal problem is to use the well known Q-learning algorithm to train one learning module to handle each of the sub-goals. The internal Q-values of the different learning modules can then be used to fairly distribute control among the modules. This approach has been independently explored in [Humphrys, 1996] and [Karlsson, 1997]. It is attractive in its simplicity, and it has shown good empirical performance in a number of domains.

In this paper we will highlight a previously unrecognized problem with existing modular Q-learning algorithms. Existing algorithms learn component policies that may be highly sub-optimal in the context of the composite task, because they do not take into account the fact that the component modules are forced to share control. We will show how to fix this problem by replacing the Q-learning with the closely related Sarsa(0) learning rule. The resulting algorithm shows improved performance on a large sample problem.

2 The problem formalized

The underlying formalism for many reinforcement learning algorithms is the Markov decision process. An MDP, denoted M is described by a 4-tuple (S, A, T, R) , where S is the state space, A is the action space, and $T(s, a, s')$ is the transition function that indicates the probability of arriving in state s' when action a is taken in state s . The reward function $R(s, a)$ denotes the expected one-step payoff for taking action a in state s . The goal of reinforcement learning algorithms is to discover an optimal policy $\pi^*(s)$ that maps states to actions so as to maximize discounted long term reward.

Here we consider the problem of discovering a joint policy for a set of N MDP's $\{M_i\}_1^N$. Throughout we will use subscripts to distinguish the MDPs. These MDP's

each have a distinct state space, but they share a common action space, and are required to execute the same action on each time step. This model is intended to map to the case of a single agent that is simultaneously faced with a set of different goals.

The N component MDPs implicitly define a larger composite MDP. Formally, the goal is to find the optimal policy for this composite MDP. The optimal composite policy is defined as the policy that maximizes summed discounted reward across the component MDPs.

The state space of the composite MDP is the cross product of the state spaces of the component MDPs: $S = S_1 \times S_2 \times \dots \times S_N$. The composite reward function is defined as: $R(s, a) = \sum_{i=1}^N R_i(s_i, a)$. In the case where the component MDPs are independent, the composite transition function can be written as: $T(s, a, s') = \prod_{i=1}^N T_i(s_i, a, s'_i)$. In the case where the component MDPs are not independent, the exact composite transition function will depend on the particular dependencies between the models.

In theory, there is no reason that the composite MDP could not be solved directly using the traditional Q-learning algorithm. However, this is generally not practical because the size of the composite state space may grow exponentially with the number of component MDPs.

3 Modular Q-Learning

Humphrys and Karlsson [Humphrys, 1996; Karlsson, 1997] independently developed similar approaches to the problem of multiple-goal reinforcement learning. The idea is that a separate learning module is created for each component MDP. The agent takes actions in the environment, and each module i is trained with the standard Q-learning update rule:

$$Q_i(s_i, a) \leftarrow (1 - \alpha)Q_i(s_i, a) + \alpha(r_i + \gamma \max_{a'} Q_i(s'_i, a')) \quad (1)$$

Where r_i is the immediate reward, α is the learning rate parameter, and γ is a discount factor applied to future rewards. In single goal reinforcement learning problems, these Q-values are used only to rank order the actions in a given state. The key observation here is that the Q-values can also be used in multiple-goal problems to indicate the degree of preference that modules have for different actions. There are several possible ways these values can be used to select a compromise action to execute. The different approaches will be referred to as action selection mechanisms.

Karlsson’s suggestion, which he calls “greatest mass”, is to generate an overall Q-value as a simple sum of the Q-values of the individual modules: $Q(s, a) = \sum_{i=1}^n Q_i(s_i, a)$. The action with the maximum summed value is then chosen to execute. We will refer to this approach as GM-Q for greatest mass Q-learning.

Humphrys considers the greatest mass approach, but raises the objection that the action with the highest sum may not be particularly good for any of the modules, with the result that no module is able to reach its goal. He explores several winner-take-all alternatives that constrain the chosen action to be optimal for at least one module. For a given state s each of the N modules promotes its own action with a value $W_i(s_i)$. The module with the largest W value is then allowed to execute its preferred action.

The simplest method for generating the W -values, which we will refer to as Top-Q, is to set $W_i(s_i) = \max_a Q_i(s_i, a)$, thus giving control to the module with the highest Q-value in the current state. This method suffers from the drawback that the module with the highest Q-value may have no preference over what action is chosen, while another module stands to lose a great deal if its action is not selected. The method sometimes exhibits reasonable performance, but this is strongly dependent on the structure of the reward functions.

A better alternative, called W-Learning, is to learn W -values according to the update rule:

$$W_i(s_i) \leftarrow \max_a Q_i(s_i, a) - (r_i + \gamma \max_b Q_i(s'_i, b)) \quad (2)$$

Where s'_i is the state that was reached as a result of some other module choosing the executed action. According to this scheme the module with the highest W value is executed at each step, and only those modules that were *not* chosen update their W -values. In this way, modules learn how much long term reward they can expect to lose if they are not chosen in the given state. The module that stands to lose the most is allowed to choose the next action.

Humphrys points out that, in the special case we are considering where the modules share an action space, there is no need to learn the W values. Instead they can be computed directly from the Q-values in a process called Negotiated W-learning. This algorithm explicitly discovers the module that can expect to lose the most long term reward if it is not allowed to choose the next action.

3.1 The problem with modular Q-learning

Q-learning has some attractive qualities as a basis for multiple-goal reinforcement learning. Chief of these is the fact that it is an off-policy learning method. This means that Q-learning for a single MDP is guaranteed to converge to the optimal solution regardless of what policy is followed during training, as long as each state-action pair is visited infinitely often in the limit. This fact makes it easy to prove convergence results for the composite reinforcement learning algorithms introduced above. In particular, it is easy to see that each module is guaranteed to converge to the optimal policy and value function for its own MDP. Since the action selection mechanisms generate a policy deterministically from the component value functions,

```

observe state  $s$ 
initialize leader  $l \leftarrow$  random module,
 $W_l \leftarrow 0$ 
 $a_l \leftarrow \operatorname{argmax}_a Q_l(s_l, a)$ 
loop:
  for all modules  $i$  other than  $l$ 
     $W_i \leftarrow \max_a Q_i(s_i, a) - Q_i(s_i, a_l)$ 
  if highest  $W_i > W_l$ 
     $W_l \leftarrow W_i$ 
     $a_l \leftarrow \operatorname{argmax}_a Q_i(s_i, a)$ 
     $l \leftarrow i$ 
  goto loop
else
  Loop has terminated. Return  $a_l$ .

```

Figure 1: Negotiated W-Learning.

the composite policy is also guaranteed to converge, although there is no guarantee concerning the quality of the composite solution.

Unfortunately, the off-policy character of Q-learning is also a serious limitation. The difficulty is that the one-step value updates for each module are computed under the assumption that all future actions will be chosen optimally for that MDP. This assumption is not valid under the action selection mechanisms described above; future actions will represent some compromise policy in which the different modules share control. This means that the computed Q-values do not converge to the actual expected return under the composite policy. Instead, the max in equation (1) results in Q-values with a positive bias.

4 Modular Sarsa(0)

A possible solution to the problem of positive bias is to replace Q-learning with an on-policy learning algorithm. In particular we will explore the use of Sarsa(0) [Rummery and Niranjan, 1994; Singh and Sutton, 1996; Sutton, 1996]. The update rule for Sarsa(0) is:

$$Q_i(s_i, a) \leftarrow (1 - \alpha)Q_i(s_i, a) + \alpha(r_i + \gamma Q_i(s'_i, a')) \quad (3)$$

This update rule is virtually identical to that for Q-learning except that the max over Q-values on the right has been replaced with the Q-value of the state action pair that is actually observed on the next step. For the case of single MDPs Sarsa(0) has been proved to converge to the optimal policy as long as the exploration rate

is asymptotically decayed toward zero according to an appropriate schedule [Singh *et al.*, 2000].

The key observation for our purposes is that, since Sarsa(0) is an on-policy method, it does not suffer from the problem of positive bias. Since updates are based on the actions that are actually taken, rather than on the best possible action, we expect Sarsa(0) based modules to discover Q-values that are closer to the true expected return under the composite policy.

Any of the action selection mechanisms from Section 3 could be recast to use Sarsa(0) rather than Q-learning to train the modules. However, we focus on the method of greatest mass. We refer to the resulting algorithm as GM-Sarsa(0). Recall that the goal is to maximize the summed reward across all of the component MDPs. Assuming that we have trustworthy utility estimates from each of the modules, it makes sense to choose the action that has the highest summed utility across all of the modules. By definition, this is the action that will lead to the greatest summed long term reward. This reasoning did not hold under Q-learning, because the utility estimates were inaccurate under the composite policy.

5 Convergence

The drawback of switching from Q-Learning to Sarsa(0) based modules is that it becomes more difficult to prove convergence results. The following four possibilities characterize the convergence guarantees that we might seek. 1) The Q-values converge to a bounded region. 2) The Q-values converge to a fixed point. 3) The Q-values converge to a local optimum. 4) The Q-values converge to a global optimum. By local optimum we mean that no module can improve its policy given the current policies of the other modules. By global optimum we mean that the Q-values result in the optimum policy for the composite MDP. At present, 1) is known to be true, 2) and 3) are both unknown, and it is not difficult to contrive a counter-example to demonstrate that 4) is false.

The fact that the Q-values are guaranteed to converge to a bounded region follows from results presented in [Gordon, 2000]. The intuition is that even though many policies may be explored during training, each Sarsa(0) update moves the Q-values closer to the correct values for *some* policy. As long as rewards are bounded, the correct Q-values for all policies must also be bounded and lie within some region. This is a very weak result. It provides no guarantees concerning the performance of the policies that GM-Sarsa(0) discovers.

Thus far, we have not been able to prove that GM-Sarsa(0) necessarily converges to a locally optimal policy, or even that it is guaranteed to converge to any fixed point at all. However, as the examples in the next section demonstrate, GM-Sarsa() does appear to converge to good policies for some problems. This is similar to the status of

Sarsa as it is used in conjunction with function approximation (e.g. [Sutton, 1996]); the empirical results are promising, but as of yet, there are no convergence proofs.

6 Examples

Figure 2 presents a simple T shaped maze task that illustrates the difficulties that may be encountered when attempting to generate a composite policy from the optimal policies of component tasks. In this case there are two component MDPs. One receives reward for reaching A, the other receives reward for reaching B, and they both receive a smaller reward for reaching C. Since these two modules share exactly the same state information, there is no practical benefit to breaking up this task into two MDP's. The point here is to show that, even in this simple case, the use of Q-learning modules can result in very bad policies.

Figure 2 a) shows the optimal composite policy for the task. Since the highest single-step summed reward is achieved by entering state C, the agent will prefer to move toward C unless it is quite close to A or B. GM-Sarsa(0) consistently converges to this optimal policy. Examining the optimal policies for the component MDPs in Figures 2 b) and c), it is clear that there is no simple way to combine them into the optimal composite policy. This is evident if we observe the state immediately above the T-juncture. In this state, both policies agree on an action that is sub-optimal for the composite task.

Figures 2 d)-f) show examples of composite policies discovered by the three Q-learning based algorithms introduced in Section 3. The bold arrows indicate locations where these policies differ from the optimal policy. The best of the three is discovered by Top-Q, which agrees with the optimal policy in 9 of 13 states. Both GM-Q and Negotiated W-Learning consistently converge to highly sub-optimal policies that result in zero expected reward for a majority of the possible starting locations.

Figure 3 demonstrates the performance of GM-Sarsa(0) on a more challenging composite task. The goal of the agent in this task is to gather stationary food items while avoiding a predator in a 5×5 grid. There are three food items present at all times. The positions of the food items as well as the positions of the agent and predator result in $25^5 \approx 10$ million distinct states. This state space is far too large for a monolithic tabular learning algorithm to be practical.

This task is good candidate for a modular reinforcement learning algorithms because it can be decomposed into several small MDPs. One MDP describes the agent's interaction with the predator, and three MDPs describe the interaction with the food items. Each of these component MDPs has $25^2 = 625$ states. Figure 3 shows the performance of GM-Sarsa(0) as well as the three Q-learning based algorithms from Section 3 on this task. Of the four algorithms, GM-Sarsa(0) exhibits the best performance.

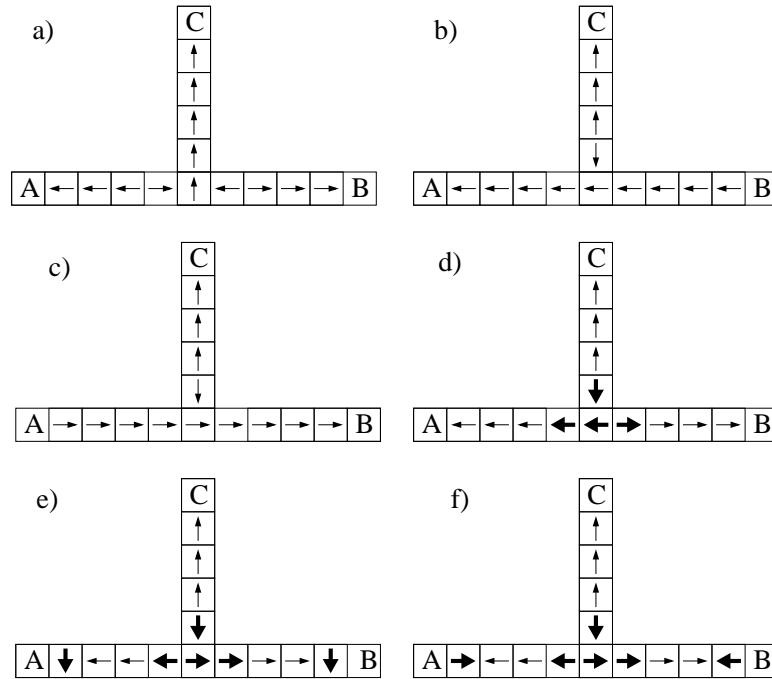


Figure 2: A simple composite maze task. A B and C are absorbing states. There are two component MDPs, LEFT and RIGHT that have identical state spaces, action spaces and transition functions, but have different reward functions. LEFT receives a reward of 1.0 for entering state A, and a reward of 0 for entering state B. RIGHT receives a reward of 1.0 for entering state B, and a reward of 0 for entering state A. Both LEFT and RIGHT receive a reward of .75 for entering state C. There are four possible actions, up, down, left and right, each of which deterministically makes the appropriate transition, or does nothing if there is a wall in the way. The discount factor is .9. a) The optimal composite policy discovered by GM-Sarsa(0). b) The optimal policy of for the LEFT MDP considered in isolation. c) The optimal policy for RIGHT. d) A composite policy discovered by Top-Q. The sub-optimal actions are indicated with bold arrows. e) A composite policy discovered by GM-Q. f) A composite policy discovered by negotiated W-Learning.

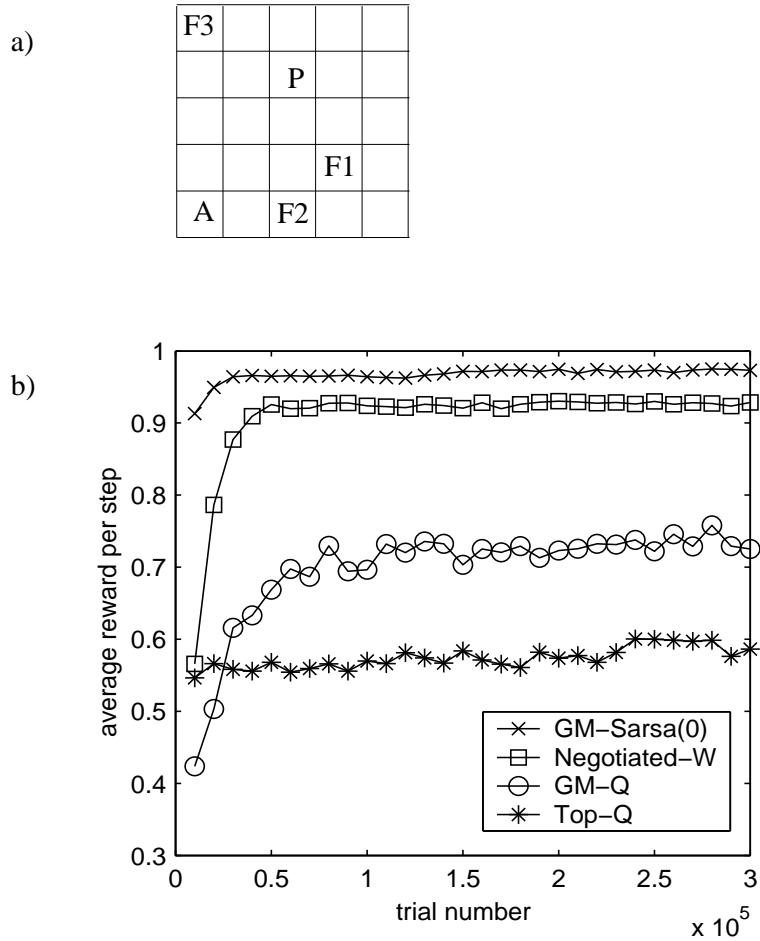


Figure 3: a) The food gathering task (adapted from [Singh and Cohn, 1998]). The agent, A, moves in any of the eight possible directions at each time step. A random move is made with a probability of .1. If the agent contacts any of the food items, F1-F3, it receives a reward of 1.0, and the item is randomly moved to a new position. The agent receives a reward of .5 for every time step that it avoids the predator, P. The predator moves deterministically one position toward the agent on every other time step. b) Results of one training run for GM-Sarsa(0) and three Q-learning based algorithms. Training is divided into trials lasting 100 time steps. Data points are generated by suspending training every 10000 trials and testing the mean performance for 1000 trials without exploration. Each algorithm uses an ϵ -greedy exploration policy with ϵ linearly reduced from .4 to 0 during the first half of trials. All algorithms use a fixed learning rate of .05, and a discount factor of .9.

7 Related work

Much previous work has gone into solving large reinforcement learning problems by breaking up monolithic tasks into sets of hierarchically structured goals and sub-goals (e.g. [Dietterich, 2000; Sun and Sessions, 2000]). The focus of our work is somewhat different. We do not focus on hierarchically structured tasks. Rather, in our work, the agent must discover an interleaved policy that adequately satisfies an ensemble of unordered sub-goals.

The work presented in this paper is related to work in the area of multi-agent reinforcement learning, surveyed in [Stone and Veloso, 2000], in that there are generally multiple reward signals, and an emphasis on solving large problems through decomposition. However, much of the effort in designing multi-agent systems relates to handling the communication between agents. In this work there is no explicit sharing of information between modules, unless different MDPs have overlapping state information.

More immediately related to this research is work such as [Singh and Cohn, 1998] and [Meuleau *et al.*, 1998] that addresses the issue of solving composite MDPs. Both approaches start with solutions, or partial solutions, to the component MDPs. In [Meuleau *et al.*, 1998] the solutions to component MDPs are heuristically combined to find an approximate solution to the composite MDP. In [Singh and Cohn, 1998] partial solutions to the component MDPs are used in a modified version of value iteration that works in the full composite state space. This algorithm is guaranteed to find the optimal composite policy, but its time and space requirements scale exponentially with the number of component MDPs. Neither of these approaches address the reinforcement learning problem.

8 Conclusion

We have presented a method for learning approximately optimal policies for a certain class of composite Markov decision processes. Empirical results demonstrate that our approach performs better than a number of existing algorithms. Future work will focus on proving convergence results for our algorithm.

Another area of future work involves exploring the use of eligibility traces. Adding eligibility traces to the Sarsa(0) algorithm results in the more general class of algorithms Sarsa(λ); GM-Sarsa(0) could easily be converted GM-Sarsa(λ). This is an interesting avenue to explore for a number of reasons. First, the use of eligibility traces often leads to faster learning. Second, Sarsa(λ) appears to perform well when used with function approximation [Sutton, 1996], and when dealing with partial observability [Loch and Singh, 1998]. These issues will be important if this work is to be extended to handle real-world domains.

References

- [Dietterich, 2000] T. G. Dietterich, “Hierarchical reinforcement learning with the MAXQ value function decomposition,” *Journal of Artificial Intelligence Research*, 13, 2000.
- [Gordon, 2000] G. J. Gordon, “Reinforcement Learning with Function Approximation Converges to a Region,” In *Advances in Neural Information Processing Systems*, volume 13, 2000.
- [Humphrys, 1996] Mark Humphrys, “Action selection methods using reinforcement learning,” In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 134–144, Cambridge, MA, 1996. MIT Press, Bradford Books.
- [Karlsson, 1997] J. Karlsson, *Learning to Solve Multiple Goals*, PhD thesis, University of Rochester, 1997.
- [Loch and Singh, 1998] J. Loch and S. Singh, “Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes,” In *Proc. ICML*, 1998.
- [Meuleau *et al.*, 1998] N. Meuleau, M. Hauskrecht, K. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier, “Solving Very Large Weakly Coupled Markov Decision Processes,” In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.
- [Rummery and Niranjan, 1994] G. A. Rummery and M. Niranjan, “On-Line Q-Learning Using Connectionist Systems,” Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.
- [Singh and Cohn, 1998] S. Singh and D. Cohn, “How to Dynamically Merge Markov Decision Processes,” In *Advances in Neural Information Processing Systems*, volume 10, 1998.
- [Singh *et al.*, 2000] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari, “Convergence results for single-step on-policy reinforcement-learning algorithms,” *Machine Learning*, 2000.
- [Singh and Sutton, 1996] S. Singh and R. Sutton, “Reinforcement Learning with Replacing Eligibility Traces,” *Machine Learning*, 22(1-3), 1996.
- [Stone and Veloso, 2000] P. Stone and M. Veloso, “Multiagent Systems: A Survey from a Machine Learning Perspective,” *Autonomous Robots*, 8(3), July 2000.

- [Sun and Sessions, 2000] R. Sun and C. Sessions, “Self-segmentation of sequences: automatic formation of hierarchies of sequential behaviors,” *IEEE Transactions on Systems, Man, and Cybernetics: Part B Cybernetics*, 30(3), 2000.
- [Sutton, 1996] R. Sutton, “Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding,” In *Advances in Neural Information Processing Systems*, volume 8, 1996.