Hierarchically Optimal Average Reward Reinforcement Learning

Mohammad Ghavamzadeh MGH@CS.UMASS.EDU
Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts Amherst, Amherst, MA 01003-4610, USA

Abstract

Two notions of optimality have been explored in previous work on hierarchical reinforcement learning (HRL): *hierarchical optimality*, or the optimal policy in the space defined by a task hierarchy, and a weaker local model called *recursive optimality*. In this paper, we introduce two new average-reward HRL algorithms for finding hierarchically optimal policies. We compare them to our previously reported algorithms for computing recursively optimal policies, using a grid-world taxi problem and a more real-world AGV scheduling problem. The new algorithms are based on a three-part value function decomposition proposed recently by Andre and Russell, which generalizes Dietterich's MAXQ value function decomposition. A key difference between the algorithms proposed in this paper and our previous work is that there is only a single global gain (average reward), instead of a gain for each subtask. Our results show the new average-reward algorithms have better performance than both the previous recursively optimal counterparts, as well as the corresponding discounted hierarchical optimal algorithms.

1. Introduction

Average reward reinforcement learning is an undiscounted infinite-horizon framework for finding *gain-optimal* policies of an unknown Markov decision process (MDP) (Mahadevan, 1996). It is generally appropriate in modeling cyclical control and optimization tasks, such as queuing, scheduling, and flexible manufacturing (Gershwin, 1994). Average reward RL has been extensively studied, using both the discrete-time Markov decision process (MDP) model (Schwartz, 1993; Mahadevan, 1996; Tadepalli & Ok, 1996; Abounadi et al., 2001) as well as the continuous-time semi-MDP (SMDP) model (Mahadevan et al., 1997; Wang & Mahadevan, 1999). Much of this work focuses on "flat" value function representations, and does not exploit the inherent hierarchical structure of sequential decision tasks. Work on hierarchical reinforcement learning has investigated how task structure can be explicitly used to decompose value functions, and accelerate the search for optimal policies. Two notions of optimality have been investigated: *hierarchical optimality* achieved in the options (Sutton et al., 1999) and HAM (Parr, 1998) models, and a weaker *recursive optimality* obtained in the MAXQ value function decomposition (Dietterich, 2000). Hierarchical optimality finds the policy optimal within the space of policies defined by the hierarchy. Recursive optimality only guarantees that the policy at each node is optimal given the policies of its children. Therefore, each subtask converges to a local optimal policy without reference to the context in which it is executed.

In our previous work (Ghavamzadeh & Mahadevan, 2001), we extended the HRL paradigm to both average-reward and continuous-time SMDP models, and introduced several new algorithms for finding recursively optimal policies. In that work, we extended the MAXQ decomposition, using a separate gain for each subtask in the task hierarchy. Recently, Andre and Russell (Andre & Russell, 2002) proposed a three part decomposition of the value function that extends the MAXQ decomposition to (discounted) hierarchically optimal policies. In this paper, we generalize their decomposition to the average reward model and propose two new average-reward algorithms that find hierarchically optimal policies. Unlike our earlier algorithms, both the proposed algorithms in this paper use only a global gain for the entire hierarchy. We test the proposed algorithms on a grid-world taxi problem, and a real-world AGV scheduling task. We find that the new hierarchical optimal average-reward algorithms perform better than not only the previous average-reward recursively optimal methods, but also the corresponding discounted methods.

The rest of this paper is organized as follows. Section 2 briefly introduces the discrete-time average reward SMDP model. Continuous-time SMDP models under both discounted and average reward paradigms have been explained in (Puterman, 1994; Bradtke & Duff, 1995) and also in our previous work (Ghavamzadeh & Mahadevan, 2001). In section 3, we illustrates recursive and hierarchical optimality. Section 4 describes the hierarchically optimal MAXQ (HO-MAXQ) value function decomposition. In section 5, we describe the proposed discrete-time and continuous-time average reward HRL algorithms using the HO-MAXQ decomposition. Section 6 presents experimental results of using proposed algorithms in a grid-world taxi problem as well as an AGV scheduling task. Finally, section 7 summarizes the paper and discusses some direction for future work.

2. Semi-Markov Decision Processes

Hierarchical RL studies how lower-level policies over primitive actions can themselves be composed into higher level policies. Policies over primitive actions are "semi-Markov" when composed at the next level up, because the "flat" policy defined by composing two lower-level policies need not be Markov with respect to the lower level state (e.g., it is not possible to predict when a robot that cleans a room by repeating a lower-level policy of vacuuming each state in the room twice will finish, purely as a function of the current state). Thus, semi-Markov decision processes (SMDPs) have become the preferred language for modeling temporally extended actions. Unlike Markov decision processes (MDPs), the time between transitions may be several time units and can depend on the transition that is made. An SMDP is defined as a five tuple (S,A,P,R,F), where S is a finite set of states, A is the set of actions, P is the state and action transition probability function, R is the reward function, and F is a function giving probability of transition times for each state-action pair. The transitions are at decision epochs only. The SMDP represents snapshots of the system at decision points, whereas the so-called *natural process* describes the evolution of the system over all times. $F(t|s,a)$ is the probability that the next decision epoch occurs within t time units after the agent chooses action a in state s at a decision epoch.

2.1 Average Reward Models

The theory of infinite-horizon semi-Markov decision processes with the average reward criterion is more complex than that for discounted models (Howard, 1971; Puterman, 1994; Mahadevan, 1996). To simplify exposition we assume that for every stationary policy, the embedded Markov chain has a *unichain* transition probability matrix. Under this assumption, the expected average reward of every stationary policy does not vary with the initial state.

In discrete-time average reward SMDP model, $v_N^\pi(s)$ denotes the expected total reward generated by the policy π up to time step N, given that the system occupies state s at time 0 and is defined as

$$v_N^\pi(s) = E_s^\pi\{\sum_{u=0}^{N-1} r(s_u, a_u)\}$$

The average expected reward or gain $g^\pi(s)$ for a policy π at state s can be expressed as the ratio

$$g^\pi(s) = \lim_{N\to\infty} \frac{E_s^\pi\{\sum_{u=0}^{N-1} r(s_u, a_u)\}}{N}$$

For unichain MDPs, the gain of any policy is state independent and we can write $g^\pi(s) = g^\pi$. For each transition, the expected number of transition steps is defined as:

$$y(s,a) = E_s^a\{N\} = \sum_{N=0}^{\infty} \sum_{s'\in S} P(s', N|s,a)$$

In discrete-time unichain average reward SMDPs, the expected average adjusted sum of rewards h^π for stationary policy π is defined as

$$h^\pi(s) = E_s^\pi\{\sum_{u=0}^{\infty}[r(s_u, a_u) - g^\pi]\} \qquad (1)$$

The Bellman equation is defined based on the h function in Equation (1) as

$$h^\pi(s) = r(s,\pi(s)) - g^\pi y(s,\pi(s)) + \sum_{s'\in S} P(s'|s,\pi(s))h^\pi(s')$$

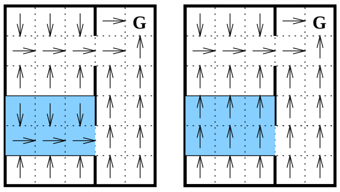
The average adjusted action value function $R^\pi(s,a)$ is similarly defined as

$$R^\pi(s,a) = r(s,a) - g^\pi y(s,a) + \sum_{s'\in S} P(s'|s,a)R^\pi(s',\pi(s'))$$

3. Recursive versus Hierarchical Optimality

Recursive and hierarchical optimality are two important forms of optimality in hierarchical reinforcement learning. Recursive optimality only guarantees that the policy of each subtask is optimal given the policies of its children. It is an important form of optimality becuase it permits each subtask to learn a locally optimal policy while ignoring the behavior of its ancestors in the hierarchy. This increases the opportunities for subtask sharing and state abstraction. The original MAXQ HRL algorithm (Dietterich, 2000) converges to recursively optimal policy. On the other hand, hierarchical optimality is a stronger form of optimality and is global optimality consistent with the given hierarchy. In this form of optimality, the policy for each individual subtask is not necessarily optimal, but the policy for the entire hierarchy is optimal. The HAMQ HRL algorithm (Parr, 1998) and the SMDP learning algorithm for a fixed set of options (Sutton et al., 1999) both converge to hierarchically optimal policy.

The following example demonstrates the difference between recursively and hierarchically optimal policies. Consider the simple maze problem in figure 1. Suppose a robot starts somewhere in the left room and it must reach the goal G in the right room. In addition to three primitive actions, *North*, *South* and *East*, robot has high level task *Exit Room* in the left room and high level task *Go to Goal* in the right room. *Exit Room* terminates when the robot exits the left room and *Go to Goal* terminates when the robot reaches the goal G. The arrows in figure 1(a) show the locally optimal policy within each room. The arrows in the left room seek to exit the room by the shortest path. The arrows in the right room follow the shortest path to the goal. However, the resulting policy is not hierarchically optimal. Figure 1(b) shows the hierarchically optimal policy that would always exit the left room by the upper door. This policy would not be locally optimal because the states in the shaded region would not follow the shortest path to the doorway.



(a) **(b)**

Figure 1. The policy shown in the left diagram is recursively optimal but not hierarchically optimal. The policy in the right diagram is hierarchically optimal but not recursively optimal. The shaded cells indicate points where the recursively optimal policy is not hierarchically optimal.

4. Hierarchically Optimal MAXQ Value Function Decomposition

A value function decomposition splits the value of a state or a state-action pair into multiple additive components. In Dietterich's MAXQ decomposition (Dietterich, 2000), the expected return for executing subtask a and then following policy π until the end of the current task i, $Q^\pi(i,s,a)$, is split into two parts and is written as $Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a)$ where *value function* $V^\pi(a,s)$ is the expected reward from executing action a in state s and *completion function* $C^\pi(i,s,a)$ is the expected reward for finishing subtask i after a is executed. Since the expected reward after subtask i execution is not a component of the action value function, this two-part decomposition allows only for recursive optimality.

Andre and Russell (Andre & Russell, 2002) recently proposed a way of achieving hierarchical optimality in the MAXQ framework, by adding a third component for the expected reward outside the current subtask i to the above decomposition and rewrite action value function as

$$Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a) + CE^\pi(i,s,a)$$

where $CE^\pi(i,s,a)$ is called the *external completion function* and is the expected reward external to the current subtask i. Each component of this three-part value function decomposition is defined as follows:

$$V^\pi(i,s) = \begin{cases} \sum_{s_1} P(s_1|s,i) r(s_1|s,i) & if\ i\ is\ primitive \\ V^\pi(child(i),s) + C^\pi(child(i),s,\pi_{child(i)}(s)) & \\ & if\ i\ is\ composite \end{cases}$$

$$C^\pi(i,s,a) = \sum_{s_2,N_1} P_i(s_2,N_1|s,a)\gamma^{N_1}[V^\pi(\pi_i(s_2),s_2) + C^\pi(i,s_2,\pi_i(s_2))]$$

$$CE^\pi(i,s,a) = \sum_{s_3,N_2} P_{EX(i)}(s_3,N_2|s,a) \gamma^{N_2} Q^\pi(parent(i),s_3,\pi_{parent(i)}(s_3))$$

where s_2 and s_3 are terminal states for subtasks a and i, N_1 and N_2 are number of time steps of execution subtasks a and i respectively, $P_i(s_2,N_1|s,a)$ is the probability of transition to the terminal state of subtask a and $P_{EX(i)}(s_3,N_2|s,a)$ is the probability of transition to the terminal state of subtask i.

5. Hierarchically Optimal Average Reward RL Algorithms

In our previous work, we extended the original MAXQ decomposition to continuous-time SMDPs and also to average reward domains. The original MAXQ HRL algorithm converges to recursively optimal policies. Recursive optimality only guarantees that the policy at each node is optimal given the policies of its children. Therefore, each subtask converges to a local optimal policy without reference to the context in which it is executed. Therefore, in the average reward algorithms described in our previous work (Ghavamzadeh & Mahadevan, 2001), we defined a separate gain for each subtask in the hierarchy to guarantee the local optimality for every subtask in the hierarchy and the recursive optimality for the entire hierarchy.

However, the HO-MAXQ decomposition described in the last section converges to a stronger form of optimality called hierarchical optimality, which is global optimality consistent with the given hierarchy. In this form of optimality, the learned policy for each individual subtask is not necessarily optimal, but the learned policy for the entire hierarchy is optimal. In this paper, we extend hierarchical optimality to the average reward model and propose one algorithm for the discrete-time average reward model and one algorithm for the continuous-time average reward model. In order to guarantee hierarchical optimality, we define only one global gain for the entire hierarchy in both the average reward HRL algorithms proposed in this paper.

To simplify exposition in the average reward algorithms proposed in our previous work, we assumed that for every possible stationary policy of each subtask in the hierarchy, the embedded Markov chain has a unichain transition probability matrix. Under this assumption every subtask in the hierarchy is a unichain SMDP. This means the gain of every subtask in the hierarchy does not vary with initial state. In the average reward algorithms described in this paper, we focus on the global optimality of the hierarchical policy instead of local optimality of each individual subtask in the hierarchy and we use a global gain for the hierarchy instead of separate gain for each individual subtask in the hierarchy. Therefore, we can relax this assumption here to assuming only that for every possible stationary policy consistent with the overall hierarchy, the embedded Markov chain has a unichain transition probability matrix and as a result the whole task can be modeled as a unichain SMDP. This means the expected average reward of every stationary policy for the hierarchy of the problem does not vary with initial state.

We now describe new average reward HRL algorithms based on the HO-MAXQ framework. At the center of the MAXQ method for HRL is the MAXQ value function decomposition. We show how the overall adjusted value function for a hierarchical policy h is decomposed into a collection of adjusted value functions for individual subtasks in the discrete-time average reward MAXQ method. The projected h function of hierarchical policy π on subtask M_i, denoted $h^\pi(i,s)$, is the average adjusted sum of rewards earned of following policy π_i (and the policies of all descendants of M_i) starting in state s until M_i terminates plus the expected average adjusted reward outside the current subtask M_i:

$$h^\pi(i,s) = \lim_{N\to\infty} E_s^\pi\{\sum_{u=0}^{N-1}(r(s_u,a_u) - g^\pi)\} \qquad (2)$$

where g^π is the global gain of the hierarchy. Now let us suppose that the first action chosen by π is invoked and executes for a number of steps and terminates in state s' according to $P_i^\pi(s'|s,a)$ and after that subtask M_i itself executes for number of steps N_1 and terminates in state s'' according to transition probability $P_i^{term}(s'',N_1|s',\pi_i(s))$. We can write Equation 2 in the form of a Bellman equation:

$$\begin{aligned} h^\pi(i,s) = r(s,\pi_i(s)) - g^\pi y_i(s,\pi_i(s)) + \sum_{s'\in S_i} P_i(s'|s,\pi_i(s)) h^\pi(i,s') \\ + \sum_{s''\in S_i, N_1} P_{EX(i)}(s'',N_1|s',\pi_i(s)) h^\pi(parent(i),s'') \end{aligned} \qquad (3)$$

Since $r(s,\pi_i(s))$ is the expected total reward between two decision epochs of subtask i, given that the system occupies state s at the first decision epoch and decision maker chooses action $\pi_i(s)$, we have

$$r(s,\pi_i(s)) = V^\pi_{y_i(s,\pi_i(s))}(\pi_i(s),s) = h^\pi(\pi_i(s),s) + g^\pi y_i(s,\pi_i(s))$$

By replacing $r(s,\pi_i(s))$ from the above expression, Equation 3 can be written as

$$\begin{aligned} h^\pi(i,s) = h^\pi(\pi_i(s),s) + \sum_{s'\in S_i} P_i(s'|s,\pi_i(s)) h^\pi(i,s') \\ + \sum_{s''\in S_i, N_1} P_{EX(i)}(s'',N_1|s',\pi_i(s)) h^\pi(parent(i),s'') \end{aligned} \qquad (4)$$

We can re-state Equation 4 for action-value function decomposition as follows:

$$R^\pi(i,s,a) = h^\pi(a,s) + \sum_{s'\in S_i} P_i(s'|s,a) R^\pi(i,s',\pi_i(s')) + \sum_{s''\in S_i,N_1} P_{EX(i)}(s'',N_1|s',\pi_i(s)) R^\pi(parent(i),s'',\pi_{parent(i)}(s''))$$

In the above equation, the second term is called the completion function and is denoted by $C^\pi(i,s,a)$ and the third term is called the external completion function and is denoted by $CE^\pi(i,s,a)$. With this definition, we can express the R function recursively as

$$R^\pi(i,s,a) = h^\pi(a,s) + C^\pi(i,s,a) + CE^\pi(i,s,a)$$

The above formulas can be used to obtain update equations for h function, completion function C and external completion function CE in discrete-time average reward framework. Pseudo-code for the resulting algorithm is shown in Algorithm 1. As mentioned above the overall task is modeled by an unichain SMDP. Therefore, after running for appropriate time, this algorithm generates a gain-optimal policy that maximizes average reward for the overall task. It does not necessarily produce bias-optimal (or T-optimal) policies that also maximize the finite reward to absorbing goal states (Mahadevan, 1996).

In update formula for CE in line 18 of the algorithm, action a'' is the subtask of action a that is taken in state s.

This algorithm can be easily extended to continuous-time by changing the update formulas for h and g in lines 5 and 7 as

$$h_{t+1}(i,s) \xleftarrow{\alpha} [k(s,i) + r(s',s,i)\tau - g_t\tau]$$

$$g_{t+1} = \frac{r_{t+1}}{t_{t+1}} = \frac{r_t + k(s,i) + r(s',s,i)\tau}{t_t + \tau}$$

where τ is the time elapsing between states s and s'.

6. Experimental Results

In this section, we first apply the discrete-time average reward HO-MAXQ algorithm proposed in this paper to a modified version of the well-known taxi problem (Dietterich, 2000), and then we will turn to the more complex AGV domain and apply both discrete-time and continuous-time average reward HO-MAXQ algorithms proposed in this paper to an AGV scheduling task.

6.1 Modified Taxi Problem

Unlike the original taxi problem (Dietterich, 2000), the version used in this paper is a continuing task. A 5-by-5 grid world inhabited by a taxi agent is shown in figure 2. There are four stations, marked as B(lue), G(reen), R(ed) and Y(ellow). The taxi starts in a randomly chosen location and passengers randomly appear at these four stations. The passenger at each station wishes to be transported to one of the three

Algorithm 1 A discrete-time hierarchically optimal average reward RL algorithm. A continuous-time version requires minor changes in steps 5 and 7.

1: Function AR-HO-MAXQ(Task i, State s)
2: let Seq= $\{\}$ be the sequence of states visited while executing i
3: **if** i is a primitive MaxNode **then**
4: execute action i in state s, observe state s' and reward $r(s',s,i)$
5: $h_{t+1}(i,s) \xleftarrow{\alpha} [r(s',s,i) - g_t]$
6: **if** i is a non-random action **then**
7: update the global average reward $g_{t+1} = \frac{r_{t+1}}{n_{t+1}} = \frac{r_t + r(s',s,i)}{n_t+1}$
8: **end if**
9: push state s into the beginning of Seq
10: **else**
11: **while** i has not terminated **do**
12: choose action a according to the current exploration policy $\pi_i(s)$
13: let ChildSeq=AR-HO-MAXQ(a,s), where ChildSeq is the sequence of states visited while executing action a
14: observe result state s'
15: let $a^* = argmax_{a' \in A_i(s')}[C_t(i,s',a') + h_t(a',s')]$
16: **for** each s in ChildSeq from the beginning **do**
17: $C_{t+1}(i,s,a) \xleftarrow{\alpha} [C_t(i,s',a^*) + h_t(a^*,s')]$
18: $CE_{t+1}(a,s,a'') \xleftarrow{\alpha} argmax_{a' \in A_i(s')} Q_t(i,s',a')$
19: **end for**
20: append ChildSeq onto the front of Seq
21: $s = s'$
22: **end while**
23: **end if**
24: **return** Seq
25: **end AR-HO-MAXQ**

other stations (also chosen randomly). The taxi must go to one of the passenger's location, pick up the passenger, go to its destination location and put down the passenger there. Then passengers once again randomly appear in four stations and the task continues. Each navigation action with probability 0.7 causes the taxi to move one cell in the corresponding direction, and with probability 0.3 moves the agent in one of the other three directions, each with probability 0.1. The system performance is measured in terms of the number of passengers deposited at their destinations per a fixed number of time steps. The state variables in this task are *taxi location*, *taxi status*, *status of each station* (whether there is a passenger waiting at that station or not), and *destination of passenger at each station*, which is the total number of 512000 states.

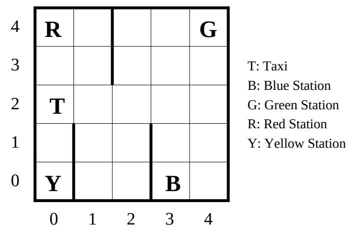


Figure 2. The Taxi Domain

Figure 3 compares the proposed discrete-time average reward HO-MAXQ algorithm with the discrete-time discounted reward HO-MAXQ algorithm (Andre & Russell, 2002) showing the better performance of the proposed average reward algorithm.

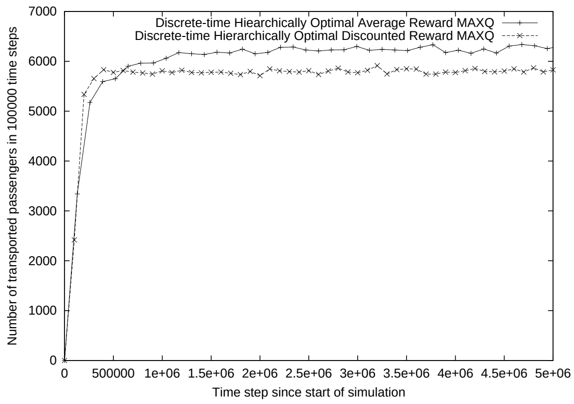


Figure 3. This plot shows that the average reward HO-MAXQ algorithm proposed in this paper works better than the discounted reward HO-MAXQ on the modified taxi problem.

6.2 AGV Scheduling Problem

We now apply both discrete-time and continuous-time average reward algorithms proposed in this paper to an AGV scheduling task. Automated Guided Vehicles (AGVs) are used in flexible manufacturing systems (FMS) for material handling. Any FMS system using AGVs faces the problem of optimal scheduling the paths of AGVs in the system. Also, when a vehicle becomes available, and multiple move requests are queued, a decision needs to be made as to which request should be serviced by that vehicle. Hence, AGV scheduling requires dynamic dispatching rules, which are dependent on the state of the system like the number of parts in each buffer, the state of the AGV, the location of the AGV, and the processing going on at the workstations. The system performance is usually measured in terms of the *throughput*, which is the number of finished assemblies deposited at the unloading deck per a fixed number of time steps.

Figure 4 shows the layout of a factory environment with three machines. Parts of type i have to be carried to drop off station at machine i (D_i) and the assembled parts brought back into the warehouse. The state variables in this task are *number of parts/assemblies in each buffer*, *part availability in the warehouse*, *AGV's status* (what each AGV is carrying), and *AGV's location*, which is the total number of 1347192 states.

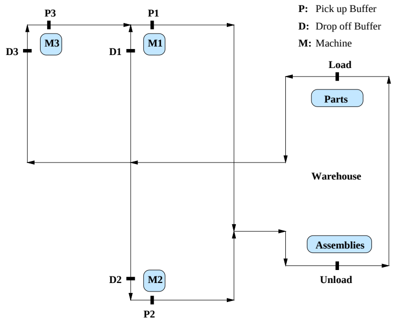


Figure 4. An AGV optimization task with an AGV agent (not shown) which carries raw materials and finished assemblies between the machines and the warehouse.

In this paper, we conduct two sets of experiments on the above AGV scheduling task and compare the performance and speed of the algorithms proposed in this paper with other related algorithms. We model the AGV scheduling task using both discrete-time and continuous-time model and compare three HRL algorithms: *average reward HO-MAXQ*, *discounted reward HO-MAXQ* and *average reward recursively optimal MAXQ*. The discrete-time and continuous-time average reward hierarchically optimal algorithms were described in the previous section, and use a single gain for the entire hierarchy. The discrete-time and continuous-time recursively optimal average reward MAXQ algorithms are algorithms were described in our previous work (Ghavamzadeh & Mahadevan, 2001) and use separate gains for every subtask in the hierarchy. In both sets of experiments, we use the task graph for the AGV scheduling task shown in figure 5.

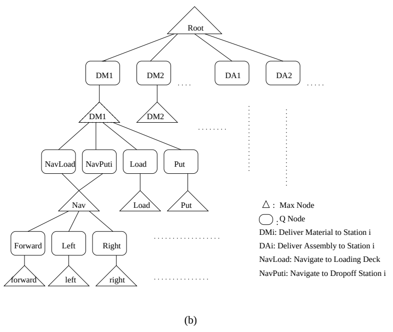


Figure 5. A hierarchical decomposition of the value function for an AGV scheduling task.

The discrete-time experimental results were generated with the following model parameters. The inter-arrival time for parts at the warehouse is uniformly distributed with a mean of 12 time steps and variance of 2 time steps. The percentage of Part1, Part2 and Part3 in the part arrival process are 40, 35 and 25 respectively. The time required for assembling the various parts are Poisson random variables with means 6, 10 and 12 time steps for Part1, Part2 and Part3 respectively.

The continuous-time experimental results were generated with the following model parameters. The time required for execution of each primitive action is a normal random variable with mean 10 seconds and variance 2 seconds. The inter-arrival time for parts at the warehouse is uniformly distributed with a mean of 100 seconds and variance of 20 seconds. The percentage of Part1, Part2 and Part3 in the part arrival process are 40, 35 and 25 respectively. The time required for assembling the various parts are normal random variables with means 100, 120 and 180 seconds for Part1, Part2 and Part3 respectively, and variance 20 seconds. In both cases, each experiment was conducted five times and the results averaged.

Figure 6 compares the proposed discrete-time average reward HO-MAXQ algorithm with the discrete-time discounted reward HO-MAXQ algorithm (Andre & Russell, 2002) and the discrete-time average reward recursively optimal algorithm described in our previous work (Ghavamzadeh & Mahadevan, 2001) showing clearly the improved performance of the proposed discrete-time average reward algorithm. This figure also shows that the average reward HO-MAXQ algorithm converges faster to the same throughput as the non-hierarchical average reward algorithm. The non-hierarchical average reward algorithm used in this experiment is relative value iteration (RVI) Q-learning (Abounadi et al., 2001). The difference in convergence speed becomes more significant as we increase the number of states.

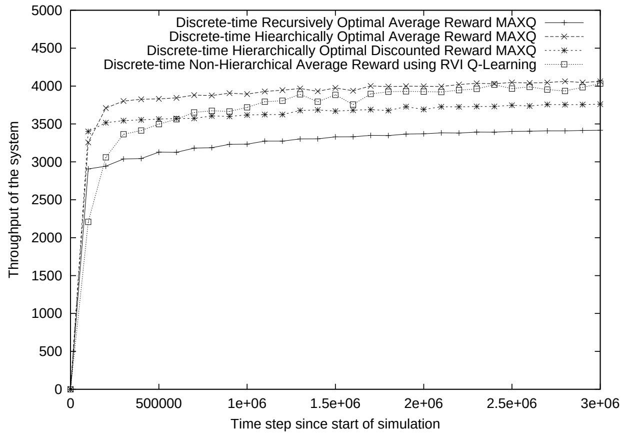


Figure 6. This plot shows that in the discrete-time case, the average reward HO-MAXQ algorithm proposed in this paper outperforms both the discounted reward HO-MAXQ HRL algorithm and our earlier average reward recursively optimal algorithm on the AGV scheduling task. It also demonstrates the faster convergence of the average reward HO-MAXQ algorithm comparing to the non-hierarchical average reward algorithm (RVI Q-learning).

Figure 7 compares the continuous-time average reward HO-MAXQ algorithm proposed in this paper with the continuous-time discounted reward HO-MAXQ algorithm and the continuous-time average reward recursively optimal algorithm from our previous work, showing that the average reward HO-MAXQ converges to the same performance as the discounted reward HO-MAXQ algorithm. Both clearly have better performance than the average reward recursively optimal algorithm. This figure also shows that the average reward HO-MAXQ algorithm converges faster to the same throughput as the non-hierarchical average reward algorithm. The non-hierarchical average reward algorithm used in this experiment is a continuous-time version of the relative value iteration (RVI) Q-learning (Abounadi et al., 2001). The difference in convergence speed becomes more significant as we increase the number of states.

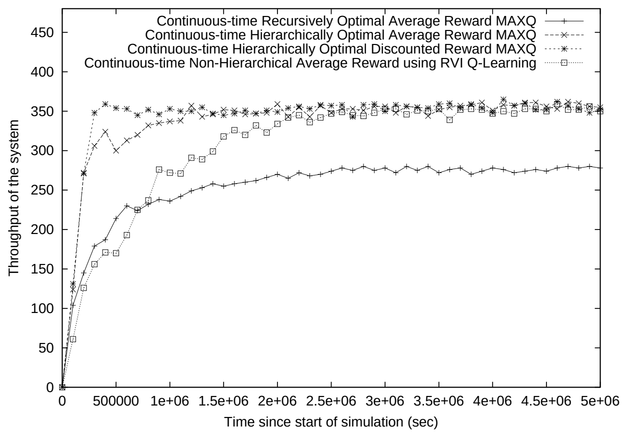


Figure 7. This plot shows that in the continuous-time case, the average reward HO-MAXQ algorithm proposed in this paper converges to the same performance as the discounted reward HO-MAXQ algorithm, and both outperform the recursively optimal average reward algorithm on the AGV scheduling task. It also demonstrates the faster convergence of the average reward HO-MAXQ algorithm comparing to the flat average reward algorithm (RVI Q-learning).

These results are both consistent with the view that the undiscounted optimality paradigm is more appropriate for cyclical tasks, such as AGV scheduling, than the more commonly studied discounted framework.

7. Conclusions and Future Work

This paper describes two new hierarchically optimal average reward HRL algorithms based on the hierarchically optimal MAXQ (HO-MAXQ) decomposition introduced in (Andre & Russell, 2002). The effectiveness of both algorithms was demonstrated by applying them to a modified version of the well-known taxi problem (Dietterich, 2000) as well as a real-world AGV scheduling problem. The proposed algorithms converge to a hierarchically optimal policy instead of a recursively optimal policy as in our previous work. Both proposed algorithms in this paper use a single gain for the entire hierarchy.

Hierarchical average reward reinforcement learning deserves further investigation. Almost all HRL approaches, such as HAMs (Parr, 1998), options (Sutton et al., 1999) and MAXQ (Dietterich, 2000) assume that subtasks always terminate. However, another approach in building a HRL framework based on the average reward SMDP model is to assume subtasks are continuing and cyclical, just as the overall task in the AGV problem is, and policies switch among these nonterminating subtasks using *interruptions*. We could also imagine a framework based on a mixture of terminating and continuing subtasks.

Many practical and theoretical issues remain to be studied in this research. We have not demonstrated a proof of convergence of the two proposed average reward algorithms. Average-reward RL convergence proofs are fairly intricate (Abounadi et al., 2001), but it would be instructive to theoretically analyze these average reward hierarchically optimal algorithms, which use one global gain for the entire hierarchy, and compare them with their recursively optimal counterparts, which use separate gains for every subtask in the hierarchy. It is obvious that these average reward HRL algorithms can be applied to many other cyclical tasks in manufacturing (Gershwin, 1994).

References

Abounadi, J., Bertsekas, D. P., & Borkar, V. S. (2001). Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, *40*, 681–698.

Andre, D., & Russell, S. (2002). State abstraction for programmable reinforcement learning agents. *To be appeared in the Proceedings of the Eighteenth AAAI.*

Bradtke, S. J., & Duff, M. O. (1995). Reinforcement learning methods for continuous-time markov decision problems. In S. Minton (Ed.), *Advances in neural information processing systems*, vol. 7, 393–400. Cambridge, MA.: MIT Press.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Artificial Intelligence Research*, *13*, 227–303.

Gershwin, S. (1994). *Manufacturing systems engineering.* Prentice Hall.

Ghavamzadeh, M., & Mahadevan, S. (2001). Continuous-time hierarchical reinforcement learning. *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 186–193).

Howard, R. A. (1971). *Dynamic probabilistic systems: Semi-markov and decision processes.* John Wiley and Sons.

Mahadevan, S. (1996). Average reward reinforcement learning: foundations, algorithms, and empirical results. *Machine Learning*, *22*, 159–196.

Mahadevan, S., Marchalleck, N., Das, T., & Gosavi, A. (1997). Self-improving factory simulation using continuous-time average reward reinforcement learning. *Proceedings of the Fourteenth International Conference on Machine Learning.*

Parr, R. E. (1998). *Hierarchical control and learning for markov decision processes.* Doctoral dissertation, Department of Computer Science, University of California, Berkeley.

Puterman, M. L. (1994). *Markov decision processes.* New York, USA: Wiley Interscience.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. *Proceedings of the Tenth International Conference on Machine Learning.*

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 181–211.

Tadepalli, P., & Ok, D. (1996). Auto-exploratory average reward reinforcement learning. *Proceedings of the Thirteenth AAAI* (pp. 881–887).

Wang, G., & Mahadevan, S. (1999). Hierarchical optimization of policy-coupled semi-markov decision processes. *Proceedings of the Sixteenth International Conference on Machine Learning.*