Continuous-Time Hierarchical Reinforcement Learning

Mohammad Ghavamzadeh GHAVAMZA@CSE.MSU.EDU
Sridhar Mahadevan MAHADEVA@CSE.MSU.EDU
Department of Computer Science, Michigan State University, East Lansing, MI 48824-1226, USA

Abstract

Hierarchical reinforcement learning (RL) is a general framework which studies how to exploit the structure of actions and tasks to accelerate policy learning in large domains. Prior work in hierarchical RL, such as the MAXQ method, has been limited to the discrete-time discounted reward semi-Markov decision process (SMDP) model. This paper generalizes the MAXQ method to continuous-time discounted and average reward SMDP models. We describe two hierarchical reinforcement learning algorithms: *continuous-time discounted reward MAXQ* and *continuous-time average reward MAXQ*. We apply these algorithms to a complex multiagent AGV scheduling problem, and compare their performance and speed with each other, as well as several well-known AGV scheduling heuristics.

1. Introduction

Hierarchical methods provide a general framework for scaling reinforcement learning to problems with large state spaces by using the task (or action) structure to restrict the space of policies. Prior work in hierarchical RL, including HAMs (Parr, 1998), options (Sutton et al., 1999) and MAXQ (Dietterich, 2000), has been limited to the discrete-time discounted reward SMDP model. This paper extends the MAXQ hierarchical RL framework to the continuous-time SMDP model, and introduces two new versions of MAXQ: one for the continuous-time discounted reward model, and one for the continuous-time average reward model. Although average reward RL has been extensively studied, using both the discrete-time MDP model (Schwartz, 1993; Mahadevan, 1996; Tadepalli & Ok, 1996) as well as the continuous-time SMDP model (Mahadevan et al., 1997; Wang & Mahadevan, 1999), prior work has been limited to "flat" algorithms. Both the proposed algorithms are tested on a complex multiagent AGV scheduling task.

The rest of this paper is organized as follows. Section 2 briefly introduces the continuous-time SMDP framework under both discounted and average reward paradigms. Section 3 describes the MAXQ method using an automated guided vehicle (AGV) task. Section 4 and 5 illustrate the continuous-time discounted reward MAXQ and continuous-time average reward MAXQ algorithms, respectively. Section 6 presents experimental results of using proposed algorithms in a multiagent AGV scheduling problem. Finally, section 7 summarizes the paper and discusses some directions for future work.

2. Semi-Markov Decision Processes

Semi-Markov decision processes (SMDPs) are useful in modeling temporally extended actions. They extend the discrete-time MDP model in several aspects. Time is modeled as a continuous entity and decisions are only made at discrete points in time (or *events*). The state of the system may change continually between decisions, unlike MDPs where state changes are only due to actions.

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 a set of state and action dependent transition probabilities, R is the reward function, and F is a function giving probability of transition times for each state-action pair. $P(s'|s,a)$ denotes the probability that action a will cause the system to transition from state s to state s'. This transition is at decision epochs only. Basically, 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. From F and P, we can compute Φ by

$$\Phi(t,s'|s,a) = P(s'|s,a)F(t|s,a)$$

where Φ denotes the probability that the system will be in state s' for the next decision epoch, at or before t time units after choosing action a in state s, at the last decision epoch. The reward function for SMDPs is more complex than in the MDP model. In addition to the fixed reward of taking action a in state s, $k(s, a)$, an additional reward may be accumulated at rate $c(s', s, a)$ for the time the natural process remains in state s' between decision epochs. Formally, the expected reward between two decision epochs, given that the system is in state s and chooses action a in the first decision epoch, is expressed as

$$r(s, a) = k(s, a) + E_s^a\{\int_0^\tau c(W_t, s, a)dt\}$$

where τ is the transition time to the second decision epoch and W_t denotes the state of the natural process during this transition.

2.1 Discounted Models

We begin with a short overview of infinite-horizon discounted semi-Markov decision processes (Puterman, 1994; Bradtke & Duff, 1995). We assume continuous-time discounting at rate $\beta > 0$, which means that the present value of one reward unit received t time units in the future equals $e^{-\beta t}$. In this model, for policy π, $v^\pi(s)$ denotes the expected infinite-horizon discounted reward, given that the process occupies state s at the first decision epoch and is defined by

$$v^\pi(s) = E_s^\pi\{\sum_{n=0}^{\infty} e^{-\beta\sigma_n}[k(s_n, a_n) + \int_{\sigma_n}^{\sigma_{n+1}} e^{-\beta(t-\sigma_n)}c(W_t, s_n, a_n)dt]\} \quad (1)$$

In the above expression, $\sigma_0, \sigma_1, \ldots$ represents the times of successive decision epochs and $e^{-\beta\sigma_n}$ transforms the reward to values at the first decision epoch. In this model, the expected discounted reward between two decision epochs is defined as

$$r(s, a) = k(s, a) + \int_0^\infty \sum_{s'\in S}[\int_0^u e^{-\beta t}c(s', s, a)P(s'|s, a)dt]F(du|s, a) \quad (2)$$

Using Equation 2, we can re-express the value function in Equation 1 as

$$v^\pi(s) = r(s, \pi(s)) + \sum_{s'\in S}\int_0^\infty e^{-\beta t}v^\pi(s')\Phi(dt, s'|s, \pi(s))$$

The action value function $Q^\pi(s, a)$ represents the discounted cumulative reward of doing an action a in state s once, and then following policy π subsequently.

$$Q^\pi(s, a) = r(s, a) + \sum_{s'\in S} P(s'|s, a)\int_0^\infty e^{-\beta t}Q^\pi(s', \pi(s'))F(dt|s, a)$$

2.2 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. (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. For policy π, state $s \in S$ and time $t \geq 0$, $v_t^\pi(s)$ denotes the expected total reward generated by the process up to time t, given that the system occupies state s at time 0 and is defined as

$$v_t^\pi(s) = E_s^\pi\{\sum_{n=0}^{v_t-1} k(s_n, a_n) + \int_0^t c(W_u, s_{v_u}, a_{v_u})du\}$$

where v_u is the number of decisions made up to time t. In this model, the expected total reward between two decision epochs is defined as

$$r(s, a) = k(s, a) + \int_0^\infty \sum_{s'\in S}[\int_0^u c(s', s, a)P(s'|s, a)dt]F(du|s, a)$$

The average expected reward or gain $g^\pi(s)$ for a policy π at state s can be defined by taking the limit inferior of the ratio of the expected total reward up until the nth decision epoch to the expected total time until the nth epoch. So, the gain of a policy $g^\pi(s)$ can be expressed as the ratio

$$g^\pi(s) = \lim_{n\to\infty} \frac{E_s^\pi\{\sum_{i=0}^n[k(s_i, a_i) + \int_{\sigma_i}^{\sigma_{i+1}} c(W_t, s_i, a_i)dt]\}}{E_s^\pi\{\sum_{i=0}^n \tau_i\}}$$

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 transition time is defined as:

$$y(s, a) = E_s^a\{\tau\} = \int_0^\infty t \sum_{s'\in S}\Phi(dt, s'|s, a)$$

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

$$h^\pi(s) = V_t^\pi(s) - g^\pi t \quad (3)$$

where t is the time at which the decision epoch occurs. The Bellman equation for unichain average reward SMDPs is defined based on the h function in

Equation 3 and can be written 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 action value function $R^\pi(s,a)$ represents the average adjusted value of doing an action a in state s once, and then following policy π subsequently.

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

3. The MAXQ Framework

The continuous-time hierarchical reinforcement learning algorithms introduced in this paper are extensions of the MAXQ method for discrete-time hierarchical reinforcement learning (Dietterich, 2000). This approach involves the use of a graph to store a distributed value function. The overall task is first decomposed into subtasks up to the desired level of detail, and the task graph is constructed. We illustrate the idea using the AGV scheduling task used as the experimental testbed in this paper. Automated Guided Vehicles (AGVs) are used in flexible manufacturing systems (FMS) for material handling. Any FMS system using AGVs faces the problem of optimally 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, 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 unit time. Figure 1 shows the layout of a factory environment. 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. This is a task which can be parallelized, if we have more than one AGV working on it. Note the agents need to learn three skills here. First, how to do each subtask, such as deliver material to stations or navigation and when to perform *Pickup* or *Load* action. Second, agents also need to learn the order to do subtasks (e.g. go to load station and load part i before heading to the drop off station at machine i). Finally, AGVs need to learn how to coordinate with other AGVs (i.e. $AGV1$ can deliver part to machine i whereas $AGV2$ can deliver the finished assembly from machine j). The strength of the MAXQ framework is that it can serve as a substrate for learning all these three types of skills.

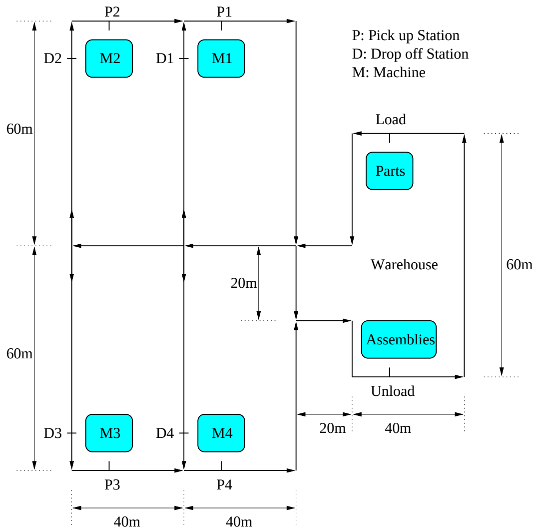

Figure 1. An AGV optimization task with four AGV agents (not shown) which carry raw materials and finished parts between the machines and the warehouse.

First, the AGV scheduling task is decomposed into subtasks and its task graph is built. Then its task graph is converted to the MAXQ graph, which is shown in figure 2. The MAXQ graph has two types of nodes: *MAX* nodes (triangles) and *Q* nodes (rectangles), which represent the different actions that can be done under their parents.

More formally, the MAXQ method decomposes an MDP M into a set of subtasks $M_0, M_1, ..., M_n$. Each subtask is a three tuple $(T_i, A_i, \tilde{R}_i)$ defined as:

- $T_i(s)$ is a termination predicate which partitions the state space S into a set of active states S_i, and a set of terminal states T_i. The policy for subtask M_i can only be executed if the current state $s \in S_i$.

- A_i is a set of actions that can be performed to achieve subtask M_i. These actions can either be primitive actions from A, the set of primitive actions for the MDP M, or they can be other subtasks.

- $\tilde{R}_i(s')$ is the pseudo reward function, which specifies a *pseudo-reward* for each transition to a terminal state $s' \in T_i$. This *pseudo-reward* tells how desirable each of the terminal states is for this particular subtask.

Each primitive action a is a primitive subtask in the MAXQ decomposition, such that a is always executable, it terminates immediately after execution,

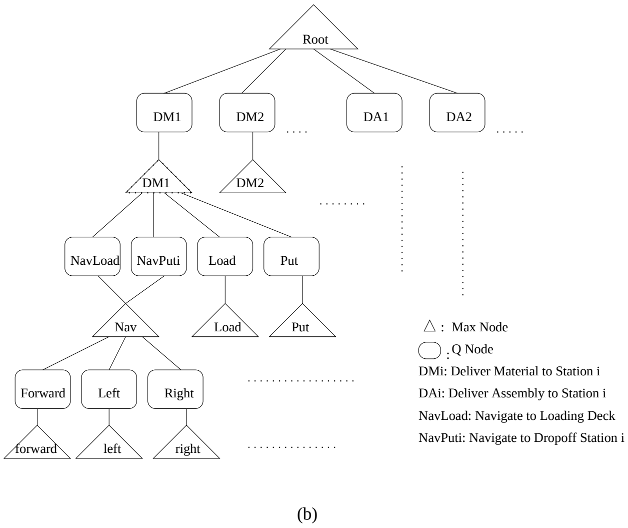

Figure 2. MAXQ graph for the AGV scheduling task.

and its *pseudo-reward* function is uniformly zero. The projected value function V^π is the value of executing hierarchical policy π starting in state s, and at the root of the hierarchy. The completion function $(C^\pi(i,s,a))$ is the expected cumulative discounted reward of completing subtask M_i after invoking the subroutine for subtask M_a in state s.

The value function $V(i,s)$ in the MAXQ method is calculated by decomposing it into two parts: the value of the subtask which is independent of the parent task, and the value of the completion of the task, which of course depends on the parent task.

$$V(i,s) = \begin{cases} max_a Q(i,s,a) & if\ i\ is\ composite \\ \sum_{s'} P(s'|s,i)R(s'|s,i) & if\ i\ is\ primitive \end{cases}$$

$$Q(i,s,a) = V(a,s) + C(i,s,a) \tag{4}$$

The Q values and the C values can be learned through a standard temporal-difference learning method, based on sample trajectories (see (Dietterich, 2000) for details). One important point to note here is that since subtasks are temporally extended in time, the Q-learning update rule used here is based on the SMDP model (Puterman, 1994).

Let us assume that an agent is at state s while doing task i, and chooses subtask j to execute. Let this subtask terminate after N steps and result in state s'. Then, the SMDP Q-learning rule used to update the completion function is given by

$$C_{t+1}(i,s,j) \leftarrow (1-\alpha)C_t(i,s,j) + \alpha\gamma^N(\max_{a'} V_t(a',s') + C_t(i,s',a'))$$

A *hierarchical* policy π is a set containing a policy for each of the subtasks in the problem: $\pi = \{\pi_0 \dots \pi_n\}$. The projected value function in the hierarchical case, denoted by $V^\pi(s)$, is the value of executing hierarchical policy π starting in state s and starting at the root of the task hierarchy. A *recursively optimal* policy for MDP M with MAXQ decomposition $\{M_0 \dots M_n\}$ is a hierarchical policy $\pi = \{\pi_0 \dots \pi_n\}$ such that for each subtask M_i the corresponding policy π_i is optimal for the SMDP defined by the set of states S_i, the set of actions A_i, the state transition probability function $P^\pi(s',N|s,a)$, and the reward function given by the sum of the original reward function $R(s'|s,a)$ and the pseudo-reward function $\tilde{R}_i(s')$. The MAXQ learning algorithm has been proven to converge to the unique recursively optimal policy for MDP M and MAXQ graph H, where M is a discounted infinite horizon MDP with discount factor γ, and H is a MAXQ graph defined over subtasks $\{M_0 \dots M_n\}$.

4. Continuous-Time Discounted Reward MAXQ Algorithm

At the center of the MAXQ method for hierarchical reinforcement learning is the MAXQ value function decomposition. We show how the overall value function for a policy is decomposed into a collection of value functions for individual subtasks for the continuous-time discounted reward model. The projected value function of hierarchical policy π on subtask M_i, denoted $V^\pi(i,s)$, is the expected cumulative discounted reward of executing π_i (and the policies of all descendents of M_i) starting in state s until M_i terminates. The value $V^\pi(i,s)$ has the following form in the continuous-time discounted reward framework:

$$V^\pi(i,s) = E_s^\pi\{\sum_{n=0}^{\infty} e^{-\beta\sigma_n} r(s_n,a_n)\} \tag{5}$$

where $r(s_n,a_n)$ is defined using Equation 2. Now let us suppose that the first action chosen by π_i is invoked and executes for a number of steps N and terminates in state s' according to $P_i^\pi(s'|s,a)$. We can rewrite Equation 5 as

$$V^\pi(i,s) = E_s^\pi\{\sum_{n=0}^{N-1} e^{-\beta\sigma_n} r(s_n,a_n) + e^{-\beta\sigma_N}\sum_{n=0}^{\infty} e^{-\beta\sigma_n} r(s_{N+n},a_{N+n})\} \tag{6}$$

The first summation on the right-hand side of Equation 6 is the discounted sum of rewards for executing subroutine $\pi_i(s)$ starting in state s until it terminates, in other words, it is $V^\pi(\pi_i(s), s)$, the projected value function for the child task $M_{\pi_i(s)}$. The second term on the right-hand side of the equation is the value of s' for the current task i, $V^\pi(i, s')$, discounted by $e^{-\beta t}$, where s' is the current state when subroutine $\pi_i(s)$ terminates and t is the sample transition time from state s to state s'. We can write Equation 6 in the form of a Bellman equation:

$$V^\pi(i,s) = V^\pi(\pi_i(s), s) + \sum_{s' \in S_i} P_i(s'|s, \pi_i(s)) \int_0^\infty e^{-\beta t} V^\pi(i, s') F_i(dt|s, \pi_i(s)) \tag{7}$$

Equation 7 can be re-stated for action-value function decomposition as follows:

$$Q^\pi(i,s,a) = V^\pi(a,s) + \sum_{s' \in S_i} P_i(s'|s,a) \int_0^\infty e^{-\beta t} Q^\pi(i, s', \pi_i(s')) F_i(dt|s,a)$$

The right-most term in this equation is the expected discounted cumulative reward of completing task M_i after executing action a in state s. This term is called the *completion function* and is denoted by $C^\pi(i,s,a)$. With this definition, we can express the Q function recursively as

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

and we can re-express the definition for V as

$$V^\pi(i,s) = \begin{cases} Q^\pi(i,s,\pi_i(s)) & if \;\; i \;\; is \; composite \\ k_i(s,i) + \int_0^\infty \sum_{s' \in S_i} P_i(s'|s,i) \left[\int_0^u e^{-\beta t} c_i(s',s,i) dt\right] F_i(du|s,i) & if \;\; i \;\; is \; primitive \end{cases}$$

We can use the above formulas to obtain update equations for value function V, outside completion function C and inside completion function $\tilde{C}$ in the continuous-time discounted reward model. Pseudo-code for the resulting algorithm is shown in Algorithm 1 [1].

[1] We use the notation $u \xleftarrow{\alpha} v$ in Algorithm 1 and Algorithm 2 as an abbreviation for the stochastic approximation update rule $u \leftarrow (1-\alpha)u + \alpha v$.

Algorithm 1 The continuous-time discounted reward MAXQ algorithm.

function MAXQ(MaxNode i, State s)
let Seq={} be the sequence of (states visited, transition times) while executing i
if i is a primitive MaxNode **then**
  execute action i in state s, observe state s' in τ time units, receive lump portion of reward $k(s,i)$ and continuous portion of reward with rate $r(s',s,i)$

$$V_{t+1}(i,s) \xleftarrow{\alpha} [k(s,i) + \frac{1-e^{-\beta\tau}}{\beta} r(s',s,i)]$$

  push (state s, transition time τ) into the beginning of Seq
else
  while i has not terminated **do**
    choose action a according to the current exploration policy $\pi_i(s)$
    let ChildSeq=MAXQ(a,s), where ChildSeq is the sequence of (states visited, transition times) while executing action a
    observe result state s'
    let $a^* = argmax_{a' \in A_i(s')}[\tilde{C}_t(i,s',a') + V_t(a',s')]$
    $T = 0$;
    for (s,τ) in ChildSeq from the beginning do
      $T = T + \tau$

$$\tilde{C}_{t+1}(i,s,a) \xleftarrow{\alpha} e^{-\beta T}[\tilde{R}_i(s') + \tilde{C}_t(i,s',a^*) + V_t(a^*,s')]$$

$$C_{t+1}(i,s,a) \xleftarrow{\alpha} e^{-\beta T}[C_t(i,s',a^*) + V_t(a^*,s')]$$

    end for
    append ChildSeq onto the front of Seq
    $s = s'$
  end while
end if
return Seq
end MAXQ

5. Continuous-Time Average Reward MAXQ Algorithm

We now describe a new average reward hierarchical reinforcement learning algorithm based on the MAXQ framework. To simplify exposition, we assume 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 expected average reward of every stationary policy for each subtask in the hierarchy does not vary with initial state. As we mentioned earlier, value function decomposition is the heart of the MAXQ method. We show how the overall h function for a policy is decomposed into a collection of h functions for individual subtasks in the continuous-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 descendents of M_i) starting in state s until M_i terminates:

$$h^\pi(i,s) = \lim_{N\to\infty} E_s^\pi\{\sum_{t=0}^{N-1}(r(s_t,a_t) - g^i\tau_t)\} \quad (8)$$

where τ's and g^i are the length of decision epochs and gain of subtask M_i respectively. 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 $P_i^\pi(s'|s,a)$. We can write Equation 8 in the form of a Bellman equation:

$$\begin{aligned} h^\pi(i,s) = {} & r(s,\pi_i(s)) - g^i y_i(s,\pi_i(s)) \\ & + \sum_{s'\in S_i} P_i(s'|s,\pi_i(s)) h^\pi(i,s') \end{aligned} \quad (9)$$

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 $\pi_i(s)$ and the expected length of time until next decision epoch is $y_i(s,\pi_i(s))$, we have

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

By replacing $r(s,\pi_i(s))$ from the above expression, Equation 9 can be written as

$$\begin{aligned} h^\pi(i,s) = {} & h^\pi(\pi_i(s),s) - (g^i - g^{\pi_i(s)}) y_i(s,\pi_i(s)) \\ & + \sum_{s'\in S_i} P_i(s'|s,\pi_i(s)) h^\pi(i,s') \end{aligned} \quad (10)$$

We can re-state Equation 10 for action-value function decomposition as follows:

$$\begin{aligned} R^\pi(i,s,a) = {} & h^\pi(a,s) - (g^i - g^a) y_i(s,a) \\ & + \sum_{s'\in S_i} P_i(s'|s,a) R^\pi(i,s',\pi_i(s')) \end{aligned}$$

In the above equation, the term

$$-(g^i - g^a) y_i(s,a) + \textstyle\sum_{s'\in S_i} P_i(s'|s,a) R^\pi(i,s',\pi_i(s'))$$

denotes the average adjusted reward of completing task M_i after executing action a in state s. This term is called the completion function and is denoted by $C^\pi(i,s,a)$. With this definition, we can express the R function recursively as

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

and we can re-express the definition for h as

$$h^\pi(i,s) = \begin{cases} R^\pi(i,s,\pi_i(s)) & \textit{if } i \textit{ is composite} \\ k(s,i) + \int_0^\infty \sum_{s'\in S_i} P_i(s'|s,i)[\int_0^u c(s',s,i)dt] F_i(du|s,i) \\ \quad - g^i \sum_{s'\in S_i} P_i(s'|s,i) \int_0^\infty t F_i(dt|s,i) & \textit{if } i \textit{ is primitive} \end{cases} \quad (11)$$

The above formulas can be used to obtain update equations for h function, outside completion function C and inside completion function $\tilde{C}$ in the continuous-time average reward model. Pseudo-code for the resulting algorithm is shown in Algorithm 2. As mentioned above, all subtasks in the hierarchy, even primitive actions, are modeled by a unichain SMDP.

6. Experimental Results

We now apply the two proposed continuous-time algorithms to the AGV scheduling task described in section 3 and compare their performance and speed with each other, as well as several well-known AGV scheduling heuristics.

The experimental results were generated with the following model parameters. There are four AGVs in the environment, the inter-arrival time for parts at the warehouse is uniformly distributed with a mean of 4 sec and variance of 1 sec. The percentage of Part1, Part2, Part3 and Part4 in the part arrival process are

Algorithm 2 The continuous-time average reward MAXQ algorithm.

function MAXQ(MaxNode i, State s)
let Seq={} be the sequence of (states visited, transition times, reward) while executing i
if i is a primitive MaxNode **then**
  execute action i in state s, observe state s' in τ time units, receive lump portion of reward $k(s,i)$ and continuous portion of reward with rate $r(s',s,i)$

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

  if i is a non-random action **then**
    update average reward or gain of subtask i

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

  end if
  push (state s, transition time τ, reward $\rho = k(s,i) + r(s',s,i)\tau$) into the beginning of Seq
else
  while i has not terminated **do**
    choose action a according to the current exploration policy $\pi_i(s)$
    let ChildSeq=MAXQ(a,s), where ChildSeq is the sequence of (states visited, transition times) while executing action a
    observe result state s'
    let $a^* = argmax_{a' \in A_i(s')}[\tilde{C}_t(i,s',a') + V_t(a',s')]$
    $T = 0; \quad R = 0;$
    for (s,τ,ρ) in ChildSeq from the beginning do
      $T = T + \tau; \quad R = R + \rho;$

$$\tilde{C}_{t+1}(i,s,a) \xleftarrow{\alpha} [\tilde{R}_i(s') - (g_t^i - g_t^a)T + \tilde{C}_t(i,s',a^*) + V_t(a^*,s')]$$

$$C_{t+1}(i,s,a) \xleftarrow{\alpha} [C_t(i,s',a^*) + V_t(a^*,s') - (g_t^i - g_t^a)T]$$

    if a is a non-random action **then**
      update average reward or gain of subtask i

$$g_{t+1}^i = \frac{r_{t+1}(i)}{t_{t+1}(i)} = \frac{r_t(i) + R}{t_t(i) + T}$$

    end if
    end for
    append ChildSeq onto the front of Seq
    $s = s'$
  end while
end if
return Seq
end MAXQ

20, 28, 22 and 30 respectively. The time required for assembling the various parts is normally distributed with means 15, 24, 24 and 30 sec for Part1, Part2, Part3 and Part4 respectively, and the variance 2 sec. The time required for primitive actions are also normally distributed. Each experiment was conducted five times and the results averaged. Since this is a multiagent task, we extend both proposed algorithms to the multiagent case using the approach introduced in (Makar et al., 2001).

In this approach (which we call *cooperative MAXQ*), each agent uses the same MAXQ hierarchy to decompose the task into subtasks. Learning is decentralized and coordination skills among agents are learned by using joint actions at the highest level of the hierarchy. The Q (or R) nodes at the highest level of the hierarchy are configured to represent the joint action space among multiple agents. In this approach, each agent only knows what other agents are doing at the level of high level subtasks, and is unaware of their lower level actions. This idea allows agents to learn coordination faster by sharing information at the level of subtasks, rather than attempting to learn coordination taking into account primitive joint state-action values.

Figure 3 compares the proposed MAXQ algorithms with several well-known AGV scheduling rules, showing clearly the improved performance of the reinforcement learning methods. As seen in this figure, the agents learn a little faster initially in the discounted MAXQ framework, but the final system throughput achieved using the average reward algorithm is higher than the discounted reward case. This result is consistent with the assumption that the undiscounted optimality framework is more appropriate for cyclical tasks same as AGV scheduling than the discounted framework.

7. Conclusions and Future Work

This paper describes two new continuous-time hierarchical RL algorithms based on the MAXQ framework. The effectiveness of both algorithms was demonstrated by applying them to a large scale multiagent AGV scheduling problem. The first algorithm extends the original discrete-time MAXQ algorithm to the continuous-time discounted SMDP model, whereas the second algorithm extends MAXQ to the continuous-time average reward SMDP model. As such, since the second algorithm is the first hierarchical average-reward algorithm proposed to our knowledge, it deserves further discussion. In particular, the MAXQ approach assumes that subtasks always terminate. How-

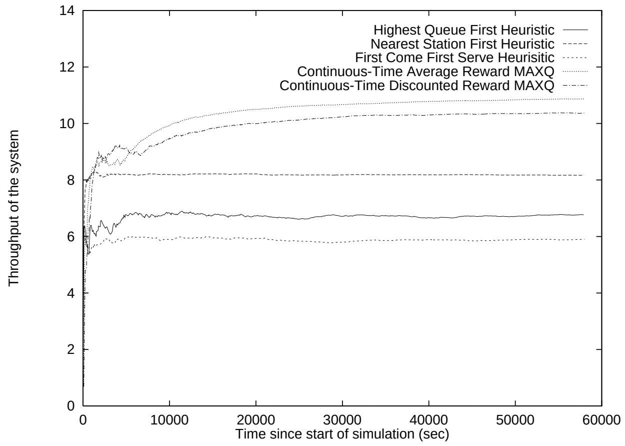

Figure 3. This plot shows both continuous-time average reward and discounted reward multiagent MAXQ algorithms outperform three well-known widely used (industrial) heuristics for AGV scheduling.

ever, for complete generality, we need to also consider the case where the subtasks are also cyclical and non-terminating, just as the overall task was in the AGV problem. One way to handle such non-terminating subtasks is to use *interruptions*, as implemented in the options framework (Sutton et al., 1999). Alternatively, one could imagine a mixed-mode framework where subtasks are optimized using an undiscounted total-reward criterion, and the parent task is formulated using an average reward criterion.

Many practical and theoretical issues remain unexplored in this research. We have not demonstrated a proof of convergence of the two algorithms. Analyzing the proof of average reward extension of MAXQ is particularly interesting. It is obvious that many other manufacturing and robotics problems can benefit from a MAXQ-like approach, particularly in the multiagent case where task-level coordination can greatly accelerate learning (Makar et al., 2001).

Acknowledgements

This work is supported by the Defense Advanced Research Projects Agency, DARPA contract No. DAANO2-98-C-4025.

References

Bradtke, S. J., & Duff, M. O. (1995). Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. In G. Tesauro, D. Touretzky and T. Leen (Eds.), *Advances in neural information processing systems*, vol. 7, 393–400. Cambridge, MA.: The MIT Press.

Dietterich, T. G. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

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* (pp. 202–210).

Makar, R., Mahadevan, S., & Ghavamzadeh, M. (2001). Hierarchical Multiagent Reinforcement Learning. *Proceedings of the Fifth International Conference on Autonomous Agents*.

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* (pp. 298–305).

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Journal of 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* (pp. 466–473).