Convergence of reinforcement learning algorithms and acceleration of learning

A. Potapov* and M. K. Ali†
Department of Physics, The University of Lethbridge, 4401 University Drive W, Lethbridge, Alberta, Canada T1K 3M4
(Received 11 May 2002; published 26 February 2003)

The techniques of reinforcement learning have been gaining increasing popularity recently. However, the question of their convergence rate is still open. We consider the problem of choosing the learning steps α_n, and their relation with discount γ and exploration degree ϵ. Appropriate choices of these parameters may drastically influence the convergence rate of the techniques. From analytical examples, we conjecture optimal values of α_n and then use numerical examples to verify our conjectures.

DOI: 10.1103/PhysRevE.67.026706 PACS number(s): 07.05.Mh, 02.30.Yy

I. INTRODUCTION

Reinforcement learning (RL) or, as it is sometimes called, ‘‘learning with a critic’’ is a kind of supervised learning when the learning agent (a controller such as a neural network), receives only evaluation signals concerning the system to be controlled. When the agent performs an action, the evaluative response (reward) does not contain information about the correct action that should have been taken. Instead, the agent gets evaluation of the present situation based on which it should learn to take right actions. This situation is typical when the desired result, but not the way to arrive at the desired result, is known. Examples of such cases include playing games such as chess or learning of living organisms and autonomous agents how to behave in a new environment, see Refs. [1,2] or [3] for more details.

Modern methods of reinforcement learning appeared about 15–20 years ago. In this relatively short period, these methods have gained popularity in solving problems of optimal control when the detailed description of the controlled system is not available or when other methods of optimal control are hard to apply. The theoretical basis of these methods includes the theory of dynamical programming (see, e.g., Ref. [4] or almost any book on optimal control) and a number of convergence theorems [3]. Most of these theorems establish only the facts of the convergence, but they do not give the estimates of the convergence rate or the choices for the parameters of the methods. From the theoretical point of view, the methods of RL estimate value functions of dynamical programming with the help of stochastic approximation. It is well known that estimates of averages often have a rather poor convergence, and since many of them need to be estimated simultaneously from the same data, the convergence must be even worse. Nonetheless, numerous experiments in both model and applied problems show that RL methods produce the best or close to the best control policies much quicker than it can be expected, provided the convergence rate is not too slow—see applications of reinforcement learning in Refs. [1–3,5–7].

Convergence or the RL methods depend on a number of parameters. Nonetheless, the recommendations concerning their choice which can be found in the literature are sometimes cotradictory. For example, for the choice of the step of learning at time t, α_t (see below), one can find recommendations to take it equal to t^{-1} [3], $k_i^{-1}(t)$ which is the number of times of being in a state i [3], 1 for some problems [1], or a small constant independent of t [8–10]. Most of these recommendations are supported by convergence theorems, which makes the actual choice slightly difficult. At the same time it follows from our own experience that the number of algorithm steps required for finding a good policy may differ by orders of magnitude for different parameters of learning, e.g., Ref. [6].

In this paper, we present recommendations for choices of the parameters of RL to increase the convergence rate. Our approach is based on approximate estimates and numerical experiments. First, we briefly mention some of the concepts of dynamical programming (DP) and describe a few methods of RL. Then we consider choices of the parameters for sample cases and then compare our conclusions with the results of numerical experiments.

II. CONTROLLED MARKOV PROCESSES AND DYNAMICAL PROGRAMMING

Dynamical programming has been developed for the so-called Markov decision processes, that is, Markov processes subjected to the controlling actions of a subsystem called a controller or ‘‘an agent.’’ The model consists of an agent that performs the actions a, and the ‘‘environment’’ that can be in certain states s. After each action, the state of the environment can change and the agent receives a reward signal r and an information about the next state s'. The value of r may depend on s, s', and a, and the goal of the agent is to maximize the total or average reward or some other functional of r. Some problems, such as control of mechanical or chemical systems, chaotic dynamical system, navigation in a maze, can be formulated in this way if a proper reward signal is defined (see Refs. [1,2,5–7] for the examples). For example, the agent in the maze navigation is punished in each state except when it is outside the maze. Here, the minimal punishment (maximal reward) corresponds to the shortest way out of the maze.

In what follows, we will consider only simple cases in which s, a, and the time t are discrete and the sets of possible values for $\{s_k\}$ and $\{a_i\}$ are finite. Some comments about

*Email address: apotapov@math.ualberta.ca
†Email address: ali@uleth.ca

more complex cases will also be made.

The basic concepts of the theory involve the terms *policy* and *value of states and actions*. A policy π is a rule for choosing action in a given state. Deterministic policy is a function $a=a(s)$, random policy is the set of probabilities $\pi_i(s)$ of taking action a_i in this state with $\Sigma_i \pi_i(s)=1$. According to the Markov property, the probability $P(s,a,s')$ of the next state s' after action a in a state s does not depend on the previous history or time. Usually, it is assumed that the value of the reward is completely determined by the set s, a, s', that is, there is a function $R(s,a,s')$, and r is always equal to one of the R values. It is assumed that the values of R are bounded, $|R(s,a,s')|<R_{\max}$. The total reward $\Sigma_{t=0}^{\infty} r_t$ during an infinite number of steps can be infinite. In order to define an optimization task, dynamic programming uses a discounted reward $\Sigma_{t=0}^{\infty}\gamma^t r_t<R_{\max}/(1-\gamma)$, $0<\gamma<1$.

Value of a state s for the given policy π, $V^{\pi}(s)$, is defined as an average total discounted reward when the process starts from the state s and the policy π is used for action selection. Averaging is used because (i) the policy may be random and (ii) the next state may vary for the same s and a. So, by definition

$$\begin{aligned} V(s)=&\sum_a \pi_a(s)\sum_{s'} P(s,a,s')R(s,a,s') \\ &+\gamma\sum_{a'} \pi_{a'}(s')\sum_{s''} P(s',a',s'') \\ &\times\left(R(s',a',s'')+\gamma\sum_{a''} \cdots\right) \\ =&\sum_a \pi_a(s)\sum_{s'} P(s,a,s')[R(s,a,s')+\gamma V(s')]. \end{aligned} \tag{1}$$

This is a system of linear equations for V. These equations are called the Bellman equations. Since $\gamma<1$, the solution exists and it is unique.

Similarly, *action values* $Q(s,a)$ are defined as an average total discounted reward when the process starts from the state s and action a (irrespective of π) and subsequently the policy π is used for action selection. It is easy to get similar equation for Q,

$$\begin{aligned} Q(s,a)=&\sum_{s'} P(s,a,s') \\ &\times\left(R(s,a,s')+\gamma\sum_{a'} \pi_{a'}(s')Q(s',a')\right) \\ =&\sum_{s'} P(s,a,s')[R(s,a,s')+\gamma V(s')]. \end{aligned} \tag{2}$$

If the Markov chain is ergodic and if for every policy there exists a limiting invariant probability of each state $p(s)$, then the policy can be characterized by an average value of state $\langle V\rangle=\Sigma_s p(s)V(s)$. Since $p(s)$ describes an invariant distribution, $\Sigma_s p(s)\Sigma_a \pi_a(s)\Sigma_{s'}P(s,a,s')=p(s')$, and taking this into account Eq. (1) gives after averaging $\langle V\rangle=\langle R\rangle+\gamma\langle V\rangle$ or

$$\begin{aligned} \langle V\rangle=&\sum_s p(s)V(s) \\ =&\frac{1}{1-\gamma}\sum_{s,a,s'} p(s)\pi_a(s)P(s,a,s')R(s,a,s') \\ =&\frac{1}{1-\gamma}\langle R\rangle=\frac{1}{1-\gamma}\lim_{N\to\infty}\frac{1}{N}\sum_{t=0}^{N} r_t. \end{aligned} \tag{3}$$

The last equality holds due to the ergodicity assumption.

An optimal policy π^* provides the maximal values of states $V^*(s)$ for the chosen γ. Note that in spite of the possibility of characterizing the policy by the mean state values (3), an *optimal policy may not give the maximal* $\langle R\rangle$: for small γ it is not important what will happen too far ahead, and an optimal policy is selected based only on the nearest future rewards. To obtain the best mean reward, it may be necessary to look far ahead.

In the theory of DP, it has been proven that the optimal deterministic policy consists in choosing the action a with the greatest $Q^*(s,a)$ or

$$a=\arg\max_u Q^*(s,u). \tag{4}$$

Such a policy is called the *greedy policy*. The optimal values of states and actions satisfy

$$V^*(s)=\max_{a'}\sum_{s'} P(s,a',s')[R(s,a',s')+\gamma V(s')] \tag{5}$$

and

$$Q^*(s,a)=\sum_{s'} P(s,a,s')[R(s,a,s')+\gamma \max_{a'} Q^*(s',a')]. \tag{6}$$

The usual way of solving these equations is *value iterations* (VI): one sets an initial guess $Q^{(0)}(s,a)$ and then obtains the next values as

$$\begin{aligned} Q^{(k+1)}(s,a)=&\sum_{s'} P(s,a,s')[R(s,a,s') \\ &+\gamma \max_{a'} Q^{(k)}(s',a')]. \end{aligned}$$

It has been shown that VI converge for any initial guess [2,4,11]. These iterations can be rewritten in terms of V as well.

Value iterations may be implemented as *policy iteration*: we set a policy $\pi^{(0)}$, evaluate $Q^{(0)}$ for this policy, and then make a new policy by choosing at every state the action $a=\arg\max_u Q^{(0)}(s,u)$. This gives the new policy $\pi^{(1)}$, and it can be shown that $Q^{(1)}(s,a)\geqslant Q^{(0)}(s,a)$. Then using $Q^{(1)}$, one generates the policy $\pi^{(2)}$, and so on. The sequence of policies converges to an optimal policy π^*. Note that the optimal policy is not required to be unique although all of them have the same Q^* and V^* [2,4].

III. REINFORCEMENT LEARNING

Now let us suppose that it is necessary to find an optimal policy for a Markov decision process without knowing the values of $P(s,a,s')$ and $R(s,a,s')$. It is assumed that one knows the states of the process, the set of available actions, and the reward after performing the current action. Therefore the policy should be extracted from the sequence s_1, a_1, r_1, s_2, a_2, r_2, ... There are several basic approaches to solve this problem.

A. Value iterations and modeling

The simplest idea is to use the same value iteration with approximate transition probabilities and rewards, that is, with $P(s,a,s')$ and $R(s,a,s')$ replaced by experimentally measured frequencies of transition from s to s' and the corresponding observed reward r. This way, one builds a model of the environment and then uses it for searching a good policy. This approach works well only if the frequencies quickly converge to probabilities, which is not often the case. If the convergence is poor, the method may give wrong estimates of Q and hence a wrong policy. There are also more sophisticated methods in RL which combine estimates of probability with other techniques, see, e.g., Refs. [1,2].

B. *Q* learning or use of stochastic approximation to solve Eq. (6)

Equation (6) can be rewritten as $\langle R(s,a)\rangle + \gamma\langle \max_{a'} Q^*(s',a')\rangle - Q^*(s,a) = 0$. Current estimates of $Q(s,a)$ differ from Q^*, therefore observations do not provide mean values, and only the values

$$\Delta_t = r_t + \gamma \max_{a'} Q^{(t)}(s_{t+1},a') - Q^{(t)}(s_t,a_t)$$

are available. Nonetheless, they are adequate to organize an iterative process that converges to Q^* and hence leads to an optimal policy.

The idea is based upon the method of *stochastic approximation*. Note that the simplest averaging can be expressed in the iterative form. For example, if one needs to estimate the average from the first n terms $x_1, \ldots, x_n$, then $S_n = (1/n)\Sigma_1^n x_i$. With the next term, the next estimate is

$$S_{n+1} = \frac{1}{n+1}\sum_1^{n+1} x_i = \frac{1}{n+1}(nS_n + x_{n+1})$$
$$= S_n + \alpha_{n+1}(x_{n+1} - S_n), \quad \alpha_{n+1} = \frac{1}{n+1}. \tag{7}$$

In 1951, this approach was generalized by Robbins and Monro [12]. They considered the following problem: one has to find the solution x^* of an equation $f(x) = f_0$ where, instead of $f(x)$, random values $\xi(x)$ are available such that only $\langle \xi(x)\rangle = f(x)$. The solution has been given in the form of the sequence x_k, which converges to x^*,

$$x_{n+1} = x_n + \alpha_n[f_0 - \xi(x_n)]. \tag{8}$$

To ensure averaging, the sequence α_n should be of the "$1/n$ type,"

$$\sum_1^\infty \alpha_n = \infty, \quad \sum_1^\infty \alpha_n^2 < \infty. \tag{9}$$

This method has been called stochastic approximation. Afterwards, the method was generalized, see, e.g., Refs. [13,14].

Most of the methods of reinforcement learning implement stochastic approximation. Q learning uses the following iterations:

$$Q^{(t+1)}(s_t,a_t) = Q^{(t)}(s_t,a_t) + \alpha_t[r_t + \gamma \max_{a'} Q^{(t)}(s_{t+1},a') - Q^{(t)}(s_t,a_t)].$$

In 1993, the stochastic approximation theorems were generalized for the proof of convergence of Q learning [15].

C. Sarsa and other methods

If Q learning can be considered as a stochastic version of value iterations, then sarsa is a stochastic version of policy evaluation which can be associated with policy improvement. The method searches for the solution of Eq. (2):

$$Q^{(t+1)}(s,a) = Q^{(t)}(s,a) + \alpha_t[r_t + \gamma Q^{(t)}(s_{t+1},a_{t+1}) - Q^{(t)}(s_t,a_t)].$$

At present, there are no rigorous results concerning sarsa. The name comes from the sequence $s_t, a_t, r_t, s_{t+1}, a_{t+1}$ involved in the evaluation process.

There are also methods for estimating V^*, for example, TD learning [2]. Note that the estimates of V alone do not give an optimal policy, one also needs the estimates of mean reward after each action. Nonetheless, there are tasks for which nonzero rewards are very rare. For example, in a game of chess the reward comes only after a long sequence of moves. In such problems, the knowledge of V is enough for action planning. Other RL methods can be found, e.g., in Refs. [2,1].

D. Exploration and ϵ-greedy policy

In value iterations method of dynamical programming, all values of $Q(s,a)$ are updated simultaneously, while in Q learning, in sarsa, and in many other RL algorithms the values are updated one at a time. This means that convergence may depend on the way of selection of the sequence of states and actions: if the system never comes to a certain state, then the corresponding Q estimates cannot be made.

There is a theorem concerning convergence of asynchronous value iterations [11]. It states that the method converges as long as each state is chosen for the updating infinitely many times and each action is chosen infinitely many times. In reinforcement learning, states and actions are selected according to the current policy, and therefore this policy should satisfy the conditions of asynchronous value iterations that allow *exploration* of states and actions [2,16].

For the Q learning, the policy may be just random. In other algorithms, e.g., sarsa, the policy should gradually converge to the optimal policy. To ensure exploration, the so-called ϵ-greedy policies are used: with the probability $1-\epsilon$ the action with the largest $Q(s,a)$ is chosen, and with the probability ϵ the action is chosen randomly. If ϵ converges gradually to 0, then the policy will converge to the optimal one.

E. Eligibility traces

This is a way to accelerate convergence of RL methods when nonzero rewards are rare. Then each nonzero reward is very valuable and it would be desirable to use it to update as many states as possible. To do it, the so-called eligibility traces have been introduced. For each state-action pair s, a one more variable is added: $e(s,a)$. At $t=0$ all e are set to 0. When the system is in the state s and the action a is chosen, one sets $e(s,a)=1$. Then after each algorithm step e is multiplied by λ, $0<\lambda<1$. Updates in learning algorithms are made for all states and actions simultaneously, but the added value is proportional to e. For example, Q learning takes the form

$$Q^{(t+1)}(s,a)=Q^{(t)}(s,a)+\alpha_t e_t(s,a)\times[r_t+\gamma\max_{a'}Q^{(t)}(s_{t+1},a')-Q^{(t)}(s_t,a_t)].$$

For $s=s_t$, $a=a_t$, it remains the same, while for others also some updates are done. Similarly, the algorithms of V estimating can be endowed with eligibility traces $e(s)$.

It has been shown that eligibility traces algorithm can be expressed in the form of forward predictions [2]. For TD(λ) method of V^* estimation the convergence has been proven [15], and λ denotes the eligibility traces.

F. Complex problems

The methods described above are efficient when the number of states is of the order of several hundreds or thousands but not too large. If the number of states is too large, there is no possibility of visiting them all and trying them all or at least all the essential actions during a reasonable time. On the other hand, the state may be continuous.

For very complex tasks, special techniques are needed. Usually, it is assumed that the task has some properties or special structures that enable one to reduce the complexity. This complexity reduction sometimes is called ‘‘coding’’ [2]. There are two main approaches.

(a) Aggregate many states s into one global state S. The number of such global states can be made not too large. After aggregation, the methods of RL are applied to the states S. The main shortcomings of this approach are (i) creation of a good aggregation is an art rather than science unless it is obvious that for a group of states the optimal action should be the same, (ii) after aggregation it is very likely that the Markov property will be violated, and therefore convergence of RL becomes a matter of luck (though luck is not too rare). See Ref. [6] for an example of learning without Markov property.

(b) Approximate $V(s)$ or $Q(s,a)$ by a fitting function $f_a(s,w)$ depending on a reasonable number of parameters w. The most popular are linear functions and neural networks [2,3]. In the latter case, the algorithm of RL is transformed into a method of neural network training, i.e., updating the weights w. This method is successful sometimes, though it also has shortcomings. Here, the matter of luck is the choice of f. The most well-known case of success of this approach is the TD Gammon, a program playing backgammon, where the method TD(λ) has been combined with neural network approximating $V(s)$ (see review of applications in Ref. [2]).

There is one more approach called direct policy estimation in which the policy is searched as a function of state $\pi(s)$, see, e.g., Ref. [17], it is also related with fitting function approximations. We shall not consider it here.

G. Non-Markov processes and incomplete description

The theory of RL and dynamical programming is essentially based on the hypothesis that the controlled process is Markov. In many practical applications it is only a hypothesis. Experiments show that methods of RL often can find a good policy even for non-Markov processes.

For non-Markov processes, there is also a theoretical result. Suppose that the frequencies of transitions s, $a\rightarrow s'$ uniformly converge to the probabilities, and the rewards after s, a uniformly converge to a mean reward $\langle r(s,a)\rangle$. Then there exists a solution of the Bellman equations (6), and value iterations should converge to it. There is a proof that under these conditions, the Q learning will also converge to the solution of Eq. (6) [18]. Note that in this case some hidden problems may arise, since actual transition probabilities may depend on previous history, initial conditions, the current policy, and so on.

Another very common problem occurs when there is a Markov decision process, but the agent does not receive complete information about the system, that is, the knowledge of the current state is only partial. For this case, the theory of partially observed Markov decision processes (MDPs) has been developed. Analysis is much more complex, but it has been shown that instead of states it is possible to consider probabilities of being in a certain state, and in terms of the probabilities the problem can be formulated as an MDP [19,20].

H. Convergence of RL algorithms, choice of parameters, and the purpose of this paper

Convergence of RL algorithms may drastically depend on the choice of the parameters discount γ, exploration ϵ, eligibility traces λ, and on the sequence of averaging steps α_n. Convergence theorems give only the bounds for these values, but usually they do not recommend their optimal values. Nonetheless, one can find in the literature a number of recommendations.

The optimal exploration level ϵ strongly depends on the sense of the problem. If the purpose is learning only, the value of ϵ may be high. On the other hand, if the system must perform some task along with learning, it cannot afford to have too much of random actions, and a trade-off between learning and performance should be reached. It depends on the specifics of the task, and hence no general recommendations can be made.

The discount γ influences the convergence, but the primary goal of its choice is to determine what policy must be considered as an optimal one. It is well known that a change of this parameter can drastically change the optimal policy—see, e.g., examples in Refs. [1,2].

The choice of eligibility traces decay factor λ may influence the convergence very strongly. As we have already mentioned, they have been introduced to update action values of all previous actions responsible for the given reward. Another interpretation is that the introduction of λ is equivalent to making predictions of future rewards [2,3], and so the choice is related with the predictability time for the system. This choice is also very specific to the problem at hand. Examples of the problems where the rewards are very rare have been considered, e.g., in Ref. [2], and numerical experiments show that the optimal values are around 0.8–0.9. Some analytical estimates can be found, e.g., in Ref. [10]. For systems in which correlations decay very fast, there is no need to use eligibility traces at all.

So we see that for the choice of ϵ, γ, and λ there are no definite formulas, but there are a few ideas that can help to make a reasonable choice. For the sequence of α_n there is no single simple rule except for the relation (9). In the literature, one can find a number of contradictory recommendations, for example, (1) take $\alpha_n=(n+1)^{-1}$, because this gives averaging in Eq. (7) [3]; (2) for some problems the optimal choice is $\alpha_n=1$ [1]; (3) take $\alpha_n=\alpha$ small enough, this choice is supported by proofs of convergence theorems in Refs. [8,9]; (4) use different α_n for each state: if the state is updated kth time at the nth step, use $\alpha_n=1/k$ [3].

At the same time, the choice of the α_n sequence may be important for convergence similar to the choice of other parameters, see, e.g., Ref. [6]. So we have tried to work out a rule which can help in choosing α_n.

In one of the early papers on stochastic approximation the following formula was proposed [13]:

$$\alpha_n=\frac{\alpha\beta}{n+\beta}. \tag{10}$$

By varying α and β, it is possible to obtain the above cited recommendations for the choice of α_n as special cases of Eq. (10). For $\alpha=\beta=1$, one gets $\alpha_n=(n+1)^{-1}$, the averaging result in recommendation (1). If $\beta=\infty$, one gets $\alpha_n=\alpha$—recommendations (2) and (3). Finally, if each Q is updated approximately every n_0th step, then, according to recommendation (4), one can take $\alpha_n=1/(n/n_0+1)=n_0/(n+n_0)$, that is, once again Eq. (10) with $\alpha=1$, $\beta=n_0$.

In this work, we have tried to work out recommendations for choosing the constants α and β to achieve better convergence. We consider below application of stochastic approximation to two simple problems which allowed us to propose a hypothesis concerning the choice of parameters. These conjectures are verified numerically.

IV. EXAMPLES OF CONVERGENCE OF STOCHASTIC APPROXIMATION AND CHOICE OF α_n

The literature on RL contains numerous examples of applications of the RL techniques. All the tasks considered can be subdivided into two major categories, namely, deterministic and stochastic tasks. An example of the deterministic task is the navigation problem of an agent in a fixed maze. In this case, after each action, the next state is unique. In the deterministic case, the transition matrix $P(s,a,s')$ for each s, a is nonzero only for one s'. All other cases which are not deterministic we shall call stochastic. It turns out that the choices of learning parameters are different for the two categories mentioned above, and the convergence rates are also different.

A. Deterministic learning

We call a task deterministic when after each action the next state s' is unique (in other words, a controlled dynamical system with complete knowledge of states) and the value of reward is always the same. Then Eq. (6) becomes much simpler,

$$Q(s,a)=R(s,a,s')+\gamma \max_{a'} Q(s',a')$$

or

$$R(s,a,s')+\gamma \max_{a'} Q(s',a')-Q(s,a)\equiv f(\{Q\})=0.$$

Here the value R is available directly from the experiment, and it is possible to try to solve this system online. The simplest way is to apply iterations

$$Q_{n+1}(s,a)=Q_n(s,a)+\alpha f(\{Q_n\}).$$

Taking into account that for the optimal policy Q^* $f(\{Q^*\})=0$,

$$\begin{aligned}
Q_{n+1}(s,a)-Q^*(s,a)&\\
&=Q_n(s,a)-Q^*(s,a)+\alpha[f(\{Q_n\})-f(\{Q^*\})]\\
&=(1-\alpha)[Q_n(s,a)-Q^*(s,a)]\\
&\quad+\alpha\gamma[\max_{a'} Q(s',a')-\max_{a'} Q^*(s',a')]
\end{aligned}$$

or

$$\begin{aligned}
|Q_{n+1}(s,a)-Q^*(s,a)|&\\
&\leqslant|1-\alpha||Q_n(s,a)-Q^*(s,a)|\\
&\quad+|\alpha|\gamma|\max_{a'} Q(s',a')-\max_{a'} Q^*(s',a')|.
\end{aligned}$$

Let the maximum of $\max_{a'} Q(s',a')$ in the last term be achieved for a_1, and that of Q^* for a_2. Since $Q(s',a_1) \geqslant Q(s',a_2)$ and $Q^*(s',a_1) \leqslant Q^*(s',a_2)$,

$$\begin{aligned}
&\left|\max_{a'} Q(s',a') - \max_{a'} Q^*(s',a')\right| \\
&\quad = |Q(s',a_1) - Q^*(s',a_2)| \\
&\quad \leqslant \max_{a' \in \{a_1,a_2\}} |Q(s',a') - Q^*(s',a')| \\
&\quad \leqslant \max_{a'} |Q(s',a') - Q^*(s',a')|.
\end{aligned}$$

Denoting $\delta_n(s,a) = |Q(s,a) - Q^*(s,a)|$, we get

$$\delta_{n+1}(s,a) \leqslant (|1-\alpha| + |\alpha|\gamma) \max_{s,a} \delta_n(s,a).$$

If the current policy allows to visit all states and try all actions, there should be convergence provided $|1-\alpha| + |\alpha|\gamma < 1$. This gives $0 < \alpha < 2/(1+\gamma)$, and the minimal value (and hence the fastest convergence) is achieved at $\alpha = 1$. Therefore, in the deterministic case the optimal scheme of Q learning is

$$Q_{n+1}(s,a) = R(s,a,s') + \gamma \max_{a'} Q_n(s',a').$$

This is the case of many classical examples of RL for navigation in mazes, as it has been noted in Ref. [1].

B. Stochastic learning

Let us consider the simplest possible situation: a system with a single state, a single action, but a number of possible rewards R_k (this situation can be a model for a system with several states, from which it always returns to one of them). It can be described by the single-action value Q, and Eq. (6) gives $Q = \Sigma P_k R_k + \gamma Q$, that is, $Q = \langle R \rangle/(1-\gamma)$. Let us consider calculation of Q from experiments (online) for this case. Iterations will have the following form:

$$Q_{n+1} = Q_n + \alpha_{n+1}(r_n + \gamma Q_n - Q_n),$$

where r_n is equal to one of the rewards R_k.

Let us choose α_n such that this choice satisfies the condition of stochastic approximation and ensures that $0 < \alpha_n < 1$ to avoid instability and to leave some freedom for the search of optimality. One of the simplest choices is recommended in the well-known paper on stochastic approximation [13] $\alpha_n = \alpha\beta/(\beta+n)$, $\beta > 0$. We denote the initial guess for Q by Q_0. Then we obtain the following iterative scheme:

$$Q_{n+1} = Q_n + \frac{\alpha\beta}{\beta+n}(r_{n+1} + \gamma Q_n - Q_n) = C_n Q_n + \alpha_n r_{n+1},$$

$$C_n = 1 - \frac{a}{\beta+n}, \quad a = \alpha\beta(1-\gamma).$$

At each step n the estimate of action value must be a weighted sum of the initial guess and the obtained rewards,

$$Q_n = w_0^{(n)} Q_0 + \sum_{i=1}^{n} w_i^{(n)} r_i.$$

It is easy to show that

$$w_0^{(n)} = C_{n-1} C_{n-2} \cdots C_0,$$

$$w_i^{(n)} = C_{n-1} C_{n-2} \cdots C_i \alpha_{i-1} = \frac{C_{n-1} C_{n-2} \cdots C_1}{C_{i-1} C_{i-2} \cdots C_1} \alpha_{i-1}.$$

To simplify consideration, we can use the following estimate:

$$\begin{aligned}
\ln(C_{n-1} C_{n-2} \cdots C_1) &= \sum_{k=1}^{n-1} \ln\left(1 - \frac{a}{\beta+n}\right) \\
&\cong -a \sum_{k=1}^{n-1} \frac{1}{\beta+n} \cong -a \int_1^n \frac{dx}{\beta+x} \\
&= -a \ln \frac{\beta+n}{\beta+1},
\end{aligned}$$

then

$$w_0^{(n)} \cong \left(\frac{\beta+1}{\beta+n}\right)^a,$$

$$w_i^{(n)} \cong \left(\frac{\beta+i}{\beta+n}\right)^a \frac{\alpha\beta}{\beta+i} = \frac{\alpha\beta}{\beta+n}\left(\frac{\beta+i}{\beta+n}\right)^{a-1}.$$

By varying β, we get different estimates for Q_n. Let us assume that all r_k are independent and identically distributed, then

$$\langle Q_n \rangle = \sum_{i=1}^{n-1} w_i^{(n)} \langle r_i \rangle = \langle R \rangle \sum_{i=1}^{n-1} w_i^{(n)},$$

$$\begin{aligned}
\sum_{i=1}^{n-1} w_i^{(n)} &\cong \sum_{i=1}^{n-1} \frac{\alpha\beta}{\beta+n}\left(\frac{\beta+i}{\beta+n}\right)^{a-1} \\
&= \frac{\alpha\beta}{(\beta+n)^a} \sum_{i=1}^{n-1} (\beta+i)^{a-1} \\
&\cong \frac{\alpha\beta}{(\beta+n)^a} \int_1^n (\beta+x)^{a-1} dx \\
&= \frac{(\beta+n)^a - (\beta+1)^a}{(1-\gamma)(\beta+n)^a} \\
&= \frac{1}{1-\gamma}\left[1 - \left(\frac{\beta+1}{\beta+n}\right)^a\right] \to \frac{1}{1-\gamma}.
\end{aligned}$$

Let us estimate the variance of Q_n:

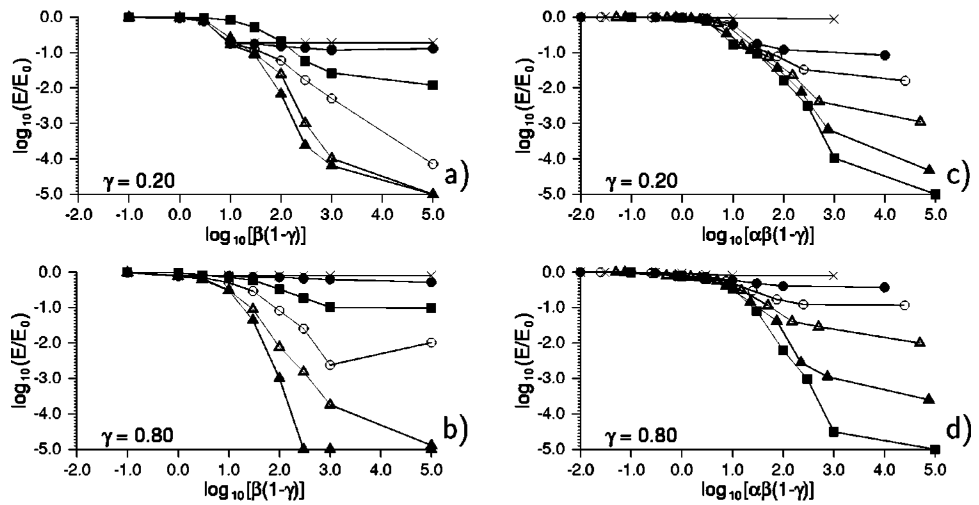

FIG. 1. Convergence of Q learning during 1000 iterations for deterministic process with $n_s=6$ states and $n_a=3$ actions. Panels (a),(b): $\alpha=1.0$, $\gamma=0.2$ and 0.8, exploration $\epsilon=0$ (×), 0.01 (●), 0.1 (○), 0.2 (△), 0.5 (filled triangle), 0.9 (■); panels (c),(d): $\epsilon=0.2$, $\gamma=0.2$, and 0.8, $\alpha=0.01$ (×), 0.1 (●), 0.25 (○), 0.5 (△), 0.75 (filled triangle), 1.0 (■). Panels (a),(b) show that the best convergence is achieved for $\epsilon=0.2$ and 0.5. Panels (c),(d) show that the convergence improves as α goes to 1 and β goes to infinity, which agrees with theoretical analysis of deterministic learning. Navigation in mazes fits into this category.

$$\langle (Q_n-\langle Q_n\rangle)^2\rangle=\sum_{i,j=1}^{n-1} w_i^{(n)}w_j^{(n)}\langle (r_i-\langle R\rangle)(r_j-\langle R\rangle)\rangle$$
$$=\sum_{i,j=1}^{n-1}(w_i^{(n)})^2(\langle r_i^2\rangle-\langle R\rangle^2)$$
$$=(\langle R^2\rangle-\langle R\rangle^2)\sum_{i,j=1}^{n-1}(w_i^{(n)})^2.$$

The last sum can be estimated as

$$D_n\sim\sum_{i,j=1}^{n-1}(w_i^{(n)})^2\cong\frac{\alpha^2\beta^2}{(\beta+n)^{2a}}\sum_{i=1}^{n-1}(\beta+i)^{2a-2}$$
$$\cong\frac{\alpha^2\beta^2}{(\beta+n)^{2a}}\int_{\beta+1}^{\beta+n}x^{2a-2}dx.$$

Now there are three different cases.

(1) $2a-2<-1$, $a<1/2$, $\beta<1/[2\alpha(1-\gamma)]$,

$$D_n\sim\frac{\alpha^2\beta^2}{(1-2a)(\beta+n)^{2a}}\left(\frac{1}{(\beta+1)^{1-2a}}-\frac{1}{(\beta+n)^{1-2a}}\right)$$
$$\sim(\beta+n)^{-2a}.$$

The influence of initial conditions decays slower than $n^{-1/2}$, and the variance decreases slower than n^{-1}.

(2) $2a-2=-1$, $a=1/2$, $\beta=1/[2\alpha(1-\gamma)]$,

$$D_n\sim\frac{\alpha^2\beta^2}{\beta+n}\ln\frac{\beta+n}{\beta+1}.$$

The influence of initial conditions decays as $n^{-1/2}$, and the variance decreases as $\ln n/n$.

(3) $2a-2>-1$, $a>1/2$, $\beta>1/[2\alpha(1-\gamma)]$,

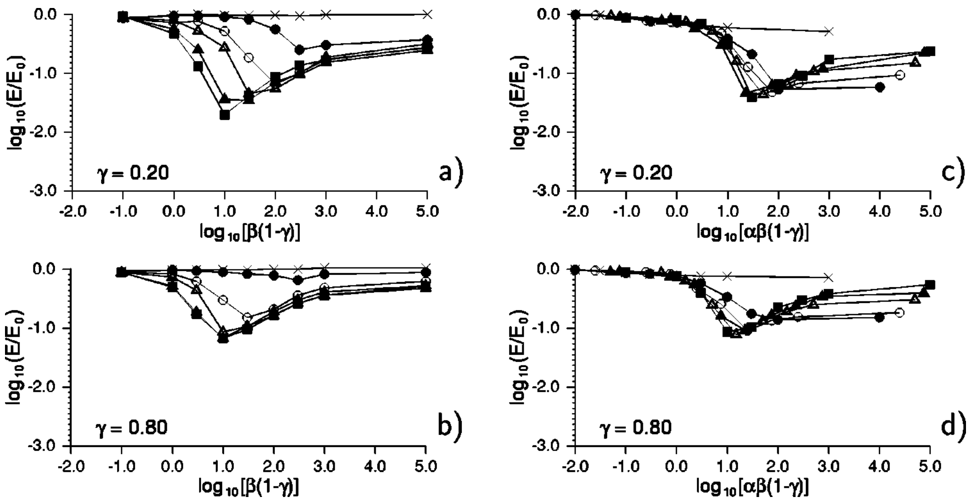

FIG. 2. Like Fig. 1, filled random process with $n_s=4$ states and $n_a=2$ actions. Panels (a),(b) show that learning is faster for higher exploration level. Panels (c),(d) show that the optimal convergence corresponds to $\alpha=1$ and $\alpha\beta(1-\gamma)$ close to 10, that is, $n_0\approx10$. Increase of γ [compare panels (a),(b) and (c),(d)] slows the convergence.

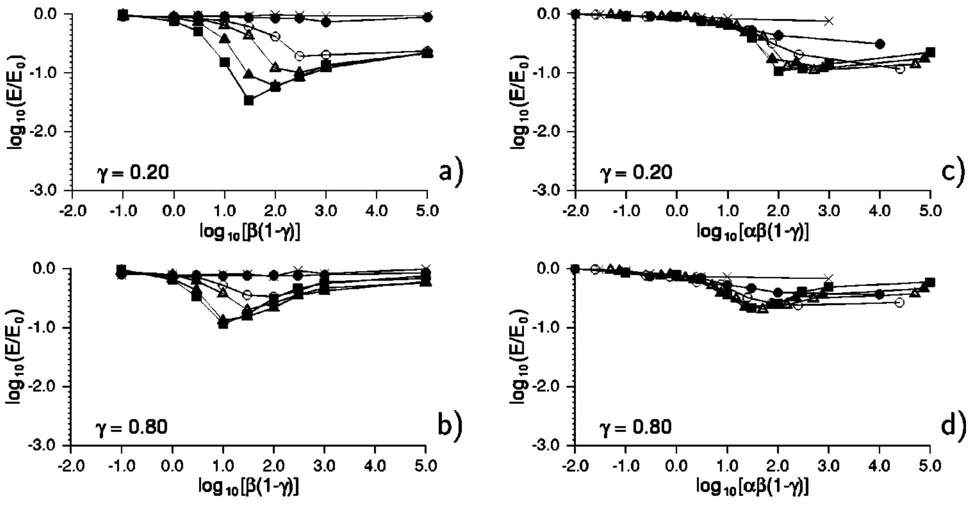

FIG. 3. Like Fig. 1, filled random process with $n_s=4$ states and $n_a=4$ actions. Panels (a),(b) show that learning is faster for higher exploration level. Panels (c),(d) show that the optimal convergence corresponds to $\alpha=1$ and $\alpha\beta(1-\gamma)=n_0$ close to 10^2. For greater γ convergence is slower.

$$D_n \sim \frac{\alpha^2\beta^2}{(2a-1)(\beta+n)^{2a}}[(\beta+n)^{2a-1}-(\beta+1)^{2a-1}]$$

$$\sim \frac{\alpha^2\beta^2}{2a-1}(\beta+n)^{-1}.$$

The influence of initial conditions decays as n^{-a}, and the variance decreases as n^{-1}.

It is obvious that the third case is preferential compared to the first two. It is impossible to change the exponent for the decay rate by varying β, but it is possible to vary the factor

$$\frac{\alpha^2\beta^2}{2a-1}=\frac{\alpha^2\beta^2}{2\alpha\beta(1-\gamma)-1}.$$

It reaches minimum for $a=1$ or $\alpha\beta=(1-\gamma)^{-1}$, and therefore for this choice

$$\alpha_n=\frac{\alpha}{1+\alpha(1-\gamma)n}, \quad D_n \sim \frac{\alpha}{(1-\gamma)[1+\alpha(1-\gamma)n]}, \tag{11}$$

and the influence of initial guess decays as

$$w_0^{(n)} \cong \frac{1+\alpha(1-\gamma)}{1+\alpha(1-\gamma)n}.$$

It is interesting that in this case $w_i^{(n)}$ is almost independent of i for i large enough, and hence the estimate of Q_n is almost exactly proportional to the estimate of $\langle r\rangle =(1/n)\Sigma_1^n r_k$.

Therefore our example gives Eq. (11) as the recommended choice of α_n. In real RL learning tasks averaging goes on slower because at every time step we get a term relevant to only one s, a pair. Possibly this can be accounted for by replacing n with n/n_0, the effective averaging rate. This gives the estimate in the form

$$\alpha_n=\frac{\alpha n_0}{n_0+\alpha(1-\gamma)n}, \quad \beta=\frac{n_0}{\alpha(1-\gamma)}. \tag{12}$$

To verify our hypothesis we performed calculations for a number of tasks, where $Q(s,a)$ can be calculated by means of dynamical programming.

V. NUMERICAL EXPERIMENTS

For numerical experiments we used a number of Markov decision processes without terminal states. The transition and

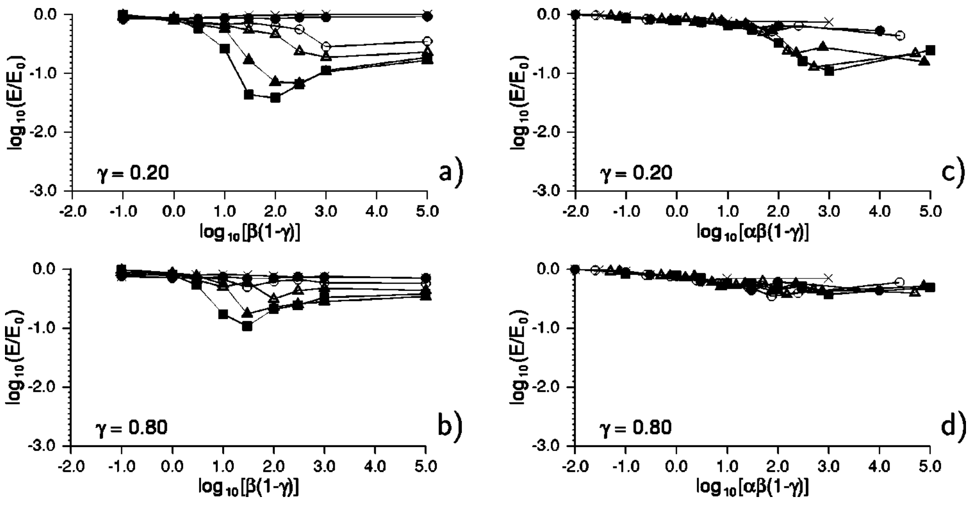

FIG. 4. Like Fig. 1, sparse random process with $n_s=4$ states and $n_a=4$ actions. Panels (a),(b) show that learning is faster for higher exploration level. Panel (c) shows that the optimal convergence corresponds to $\alpha=1$ and $\alpha\beta(1-\gamma)=n_0$ close to 10^3. For greater γ in panel (d) there is almost no convergence.

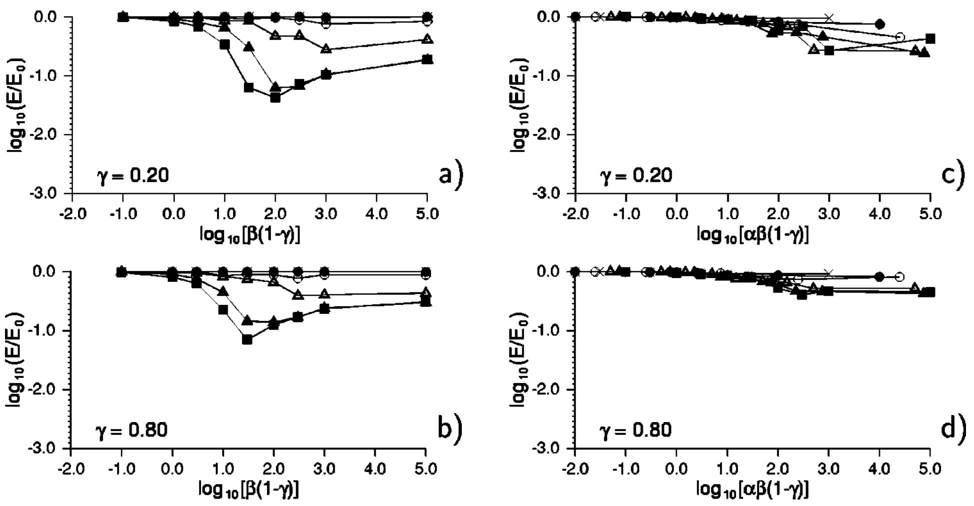

FIG. 5. Like Fig. 1, sparse random process with $n_s = 6$ states and $n_a = 3$ actions. Panels (a),(b) show that learning is faster for higher exploration level. Panel (c) shows that the optimal convergence corresponds to $\alpha \geqslant 0.5$ and $\alpha\beta(1-\gamma) = n_0 \geqslant 10^3$. For greater γ in panel (d) there is almost no convergence.

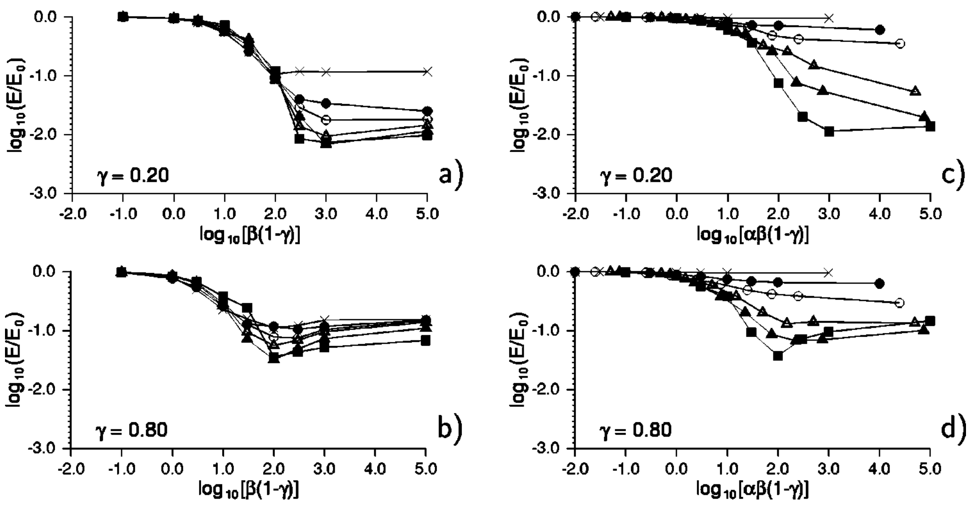

FIG. 6. Like Fig. 11, sparse random process with $n_s = 6$ states and $n_a = 3$ actions, obtained from the problem of navigation of an agent in a grid world. Panels (a),(b) show that learning is faster for higher exploration level. Panel (c) shows that the optimal convergence corresponds to $\alpha = 1$ and $\alpha\beta(1-\gamma) = n_0 \geqslant 10^3$. For greater γ in panel (d) the convergence is slower.

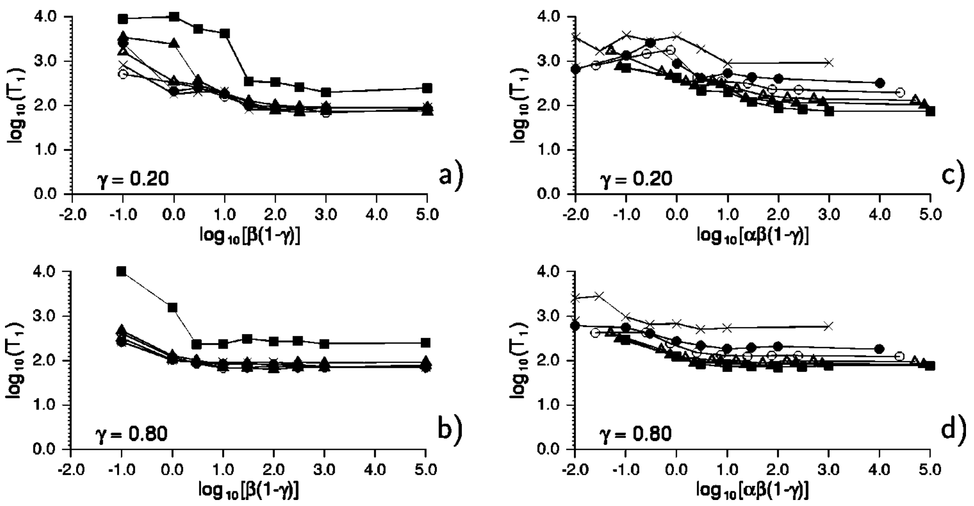

FIG. 7. Time needed for finding an optimal policy T_1 for the same process as in Fig. 1. If $T_1 = 10^4$, the optimal policy has not been found. In most cases the optimal policy has been found when the accuracy of Q estimates were 10% or worse. The controlled process has six states, three actions, and optimal behavior involves all six states.

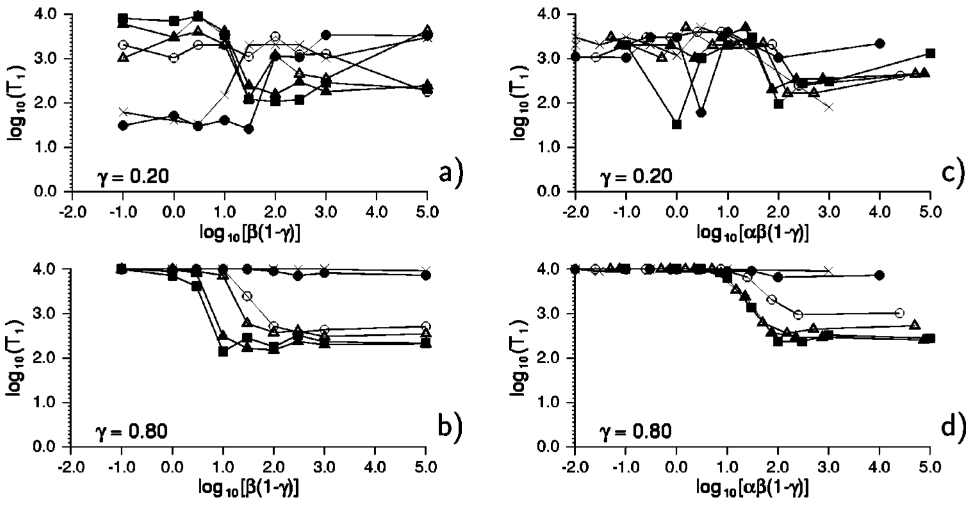

FIG. 8. Time needed for finding an optimal policy T_1 for the same process as in Fig. 3. The controlled process has four states, four actions, and optimal behavior is mainly concentrated on three states.

reward matrices P and R were generated randomly. Then we calculated the optimal policy for them and used only those for which optimal policy for $\gamma=0$ (estimated from the matrix R only) differs from that for $\gamma>0$, that is, the possibility to find the optimal policy occasionally only from the reward matrix was excluded. We performed calculations for various values of γ, α, β, and ϵ. Initial values were always $Q=0$.

Since fast convergence is important in practical applications, we made 2000 steps of RL algorithm, at every step we calculated the current error $E(t)=\max_{s,a}|Q_t(s,a)-Q^*(s,a)|$, and considered the ratio of mean error at the steps 10–20 (E_0) and 1000–2000 (E_1). This ratio has been averaged over ten independent runs.

Because of numerous repetitions, the calculations are rather time consuming, and for this reason we used Markov processes with a few states n_s and actions n_a only. We used three types of Markov processes: (i) deterministic, when for all s, a only one entry $P(s,a,s')=1$ and all other are zero, (ii) "sparse" random—only one to two nonzero entries for each s, a, (iii) "filled" random, when almost all entries are nonzero. The presented figures were obtained for the following parameters. Panels (a),(b): $\alpha=1.0$, $\gamma=0.2$ and 0.8, various β, and $\epsilon=0$ (×), 0.01 (●), 0.1 (○), 0.2 (△), 0.5 (filled triangle), 0.9 (■); panels (c),(d): $\epsilon=0.2$, $\gamma=0.2$ and 0.8, various β, and $\alpha=0.01$ (×), 0.1 (●), 0.25 (○), 0.5 (△), 0.75 (filled triangle), 1.0 (■).

The results for deterministic case (Fig. 1) show that the best convergence rate is obtained for $\beta\cong10^5$, which means that during 2000 steps $\alpha_n=\alpha\beta/(\beta+n)$ remained practically constant. The closer α is to 1, the better is the convergence. This is in a good agreement with convergence analysis for deterministic systems in Sec. IV.

For filled random processes (Figs. 2 and 3), again in a good agreement with the examples above, there is an optimal value of β, which depends on α, γ, and ϵ. The dependence on α is close to the theoretical one, $\beta\sim\alpha^{-1}$ (optimal convergence corresponds to approximately the same value of $\alpha\beta$). The value of n_0 estimated from the figures as the value of $\alpha\beta(1-\gamma)$ at the point of the plot minimum in panels (c) and (d) is about 10–100.

For sparse random processes (Figs. 4–6) situation is intermediate, that is, usually there is also an optimal β, but n_0 is greater than in the previous case—the minimum of the error plot in panels (c) and (d) corresponds to $n_0=\alpha\beta(1-\gamma)$ equal to 10^2, 10^3, or more.

The main conclusions are the following.

(1) Convergence rate is better when $\alpha=1$.

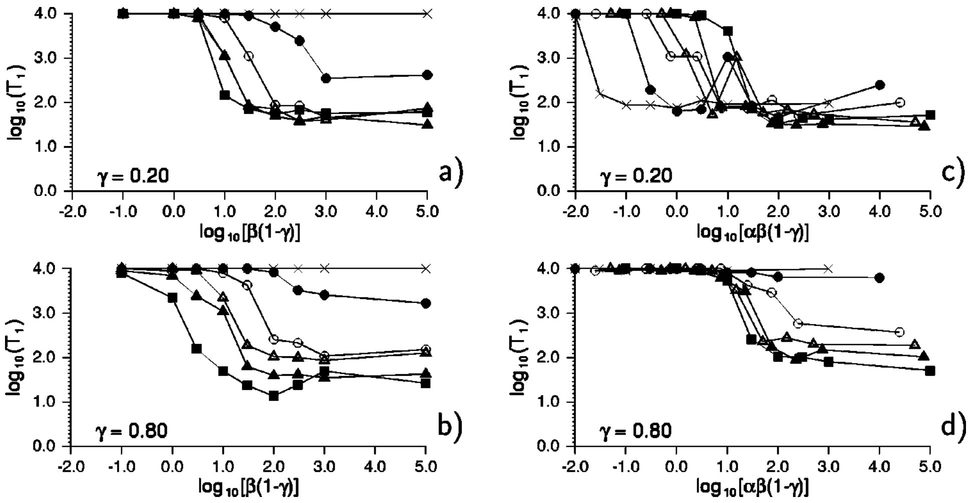

FIG. 9. Time needed for finding an optimal policy T_1 for the same process as in Fig. 4. The controlled process has four states, four actions, and optimal behavior is concentrated with 60 on one state, and the rest of probability is spread onto remaining three states.

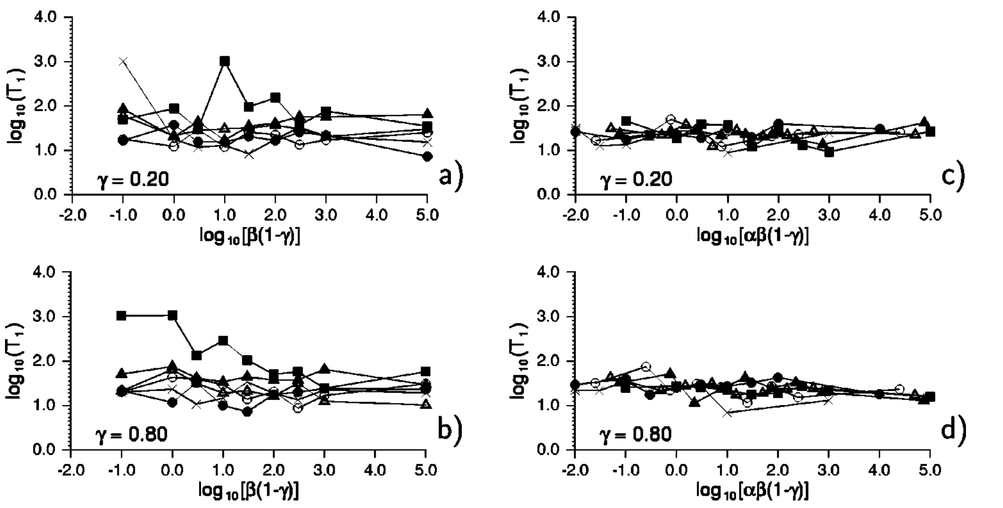

FIG. 10. Time needed for finding an optimal policy T_1 for the same process as in Fig. 6. The controlled process has six states, three actions, and optimal behavior involves only two states. An optimal policy is found much faster than in other cases.

(2) The best convergence is achieved with $\beta = n_0/\alpha(1-\gamma)$, where n_0 depends on the controlled system and may vary from ~ 10 to ∞. There is no good interpretation for the parameter n_0 in terms of dynamical programming or stochastic approximation for both stochastic and deterministic systems—for the latter it cannot be considered as a mean updating interval for Q. Nonetheless, for practical purposes this parameter appears convenient. As it follows from experiments, the more deterministic the controlled process is (that is, the more definite the result of actions is), the greater is the optimal n_0.

(3) For greater γ the convergence may be slower, especially for large ϵ.

(4) The greater the ϵ, the faster is the convergence, especially for optimal β.

Nonetheless, the convergence is rather weak except for the case of deterministic system with $\alpha_n \cong 1$: the best result for random system is the factor 10^{-1}–10^{-2} during 1000 iterations, that is, the best rate is close to $\sim n^{-0.5}$ (in agreement with theoretical result for variance as $\sim n^{-1}$). In spite of this slow convergence, usually the method finds an optimal policy during 2000 iterations.

Figures 7–10 show the mean number of iterations needed for the algorithm to find an optimal policy. Comparison with the Figs. 1–6 shows that the error in Q estimates when the optimal policy has been found is 10% or worse. Hence the methods of reinforcement learning do not require very accurate estimates of action values to provide a good policy. Most probably, this is the main reason of their success. In many examples only a few hundred or thousand time steps are necessary to get a good policy, and, as it follows from the figures, sometimes a good policy has been found under a very poor convergence. The reason of this phenomenon is not clear yet. Possibly this is related with the fact that the optimal policy often generates rather simple system's dynamics involving only a few states. This means that the reward from this short chain is greater than that from other short chains. Methods of reinforcement learning always start with evaluating rewards from short chains, especially if $\alpha_n \cong 1$. Then, after a certain number of steps enough to evaluate the short chains, the exact optimal policy can be found. If this hypothesis is true, it may explain the observed facts. In favor of this conjecture is also the fact that the longest transient period corresponds to optimal sequence of six states (Fig. 7), and the shortest transient to optimal sequence of two states (Fig. 10). However, this conjecture is hard to put in rigorous form, though the mentioned effect appears to be the main reason of the success of the reinforcement learning techniques. Anyway, proper choice of learning parameters can accelerate learning significantly.

VI. CONCLUSIONS

We considered the problem of choosing parameters of Q learning. We used the form of the sequence $\alpha_n = \alpha\beta/(\beta+n)$ proposed in Ref. [13], and showed that the best results were obtained with $\alpha = 1$. The optimal value of β depends on the properties of the MDP. β is $\geqslant 1$, and often it is 10, 100, or more. As the randomness in a process decreases, the optimal β value increases. For a pure deterministic process, in which each action uniquely defines the next state, the optimal value is $\beta = \infty$, that is, $\alpha_n = 1$. The value of γ is not too critical for convergence (though for smaller γ the convergence is usually better). γ is more important for the appropriate evaluation of performance: local, only a few steps ahead ($\gamma \cong 0$) or global ($\gamma \cong 1$) optimization is necessary. Usually the convergence is better for large exploration level ϵ, but its optimal choice depends on the specific problem concerning what level of randomness can be allowed, see, e.g., Ref. [2].

We must note that our results were tested on a limited number of systems. Nonetheless, for a new RL task our results provide a possible way of achieving a better convergence. We recommend applying RL methods with $\alpha_n = n_0/[n_0 + (1-\gamma)n]$ and n_0 equal to 10, 100, 1000, and 10^5 (which is close enough to ∞), and compare the resulting policies to find the optimal β value.

ACKNOWLEDGMENTS

This work has been supported through a grant to M.K.A. from the Defense Research Establishment Suffield under Contract No. W7702-8-R745/001/EDM. The work was carried out using computing facilities provided by Multimedia Advanced Computational Infrastructure (MACI).

[1] L.P. Kaebling, M.L. Littman, and A.W. Moore, J. Artif. Intell. Res. **4**, 237 (1996).
[2] R.S. Sutton and A.G. Barto, *Reinforcement Learning* (Bradford Book/MIT Press, Cambridge, MA, 1998).
[3] D.P. Bertsekas and J.N. Tsitsiklis, *Neurodynamic Programming* (Athena Scientific, Belmont, MA, 1996).
[4] R.E. Bellman and S.E. Dreyfus, *Applied Dynamic Programming* (RAND Corporation, Santa Monica, CA, 1962).
[5] S. Gadaleta and G. Dangelmayr, Chaos **9**, 775 (1999).
[6] A.B. Potapov and M.K. Ali, Phys. Rev. E **63**, 046215 (2001).
[7] *Neural Networks for Control*, W.T. Miller, R.S. Sutton, and P.J. Werbos (MIT Press, Cambridge, London, 1990).
[8] P. Dayan, Mach. Learn. **8**, 341 (1992).
[9] P. Dayan and T.J. Sejnowski, Mach. Learn. **14**, 295 (1994).
[10] S. Singh and P. Dayan, Mach. Learn. **32**, 5 (1997).
[11] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
[12] H. Robbins and S. Monro, Ann. Math. Stat. **22**, 400 (1951).
[13] A. Dvoretzky, in *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability* (University of California Press, Berkeley, CA, 1956), p. 39.
[14] H.J. Kushner and G.G. Yin, *Stochastic Approximation Algorithms and Applications* (Springer, New York, 1997).
[15] T. Jaakkola, M.I. Jordan, and S.P. Singh, Neural Comput. **6**, 1185 (1994).
[16] S.B. Thrun, in *Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches*, edited by D.A. White and D.A. Sofge (Van Nostrand Reinhold, New York, 1992).
[17] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour, *Advances in Neural Information Processing Systems 12* (MIT Press, Cambridge, MA, 2000), p. 1057.
[18] S.P. Singh, T. Jaakkola, and M.I. Jordan, in *Advances in Neural Information Processing Systems 7*, edited by G. Tesauro, D. Touretzky, and T. K. Leen (MIT Press, Cambridge, MA, 1995), p. 361.
[19] T. Jaakkola, S.P. Singh, and M.I. Jordan, in *Advances in Neural Information Processing Systems 7*, edited by G. Tesauro, D. Touretzky, and T. K. Leen (Morgan Kaufmann, San Mateo, CA, 1995), p. 345.
[20] G. Monahan, Manage. Sci. **28**, 1 (1982).