A Reinforcement Learning Method Based on Adaptive Simulated Annealing

Amir F. Atiya
Dept Computer Engineering
Cairo University
Giza, Egypt
amiratiya@link.net

Alexander G. Parlos
Dept Mechanical Engineering
Texas A&M University
College Station, TX 77843
a-parlos@tamu.edu

Lester Ingber
Lester Ingber Research
ingber@ingber.com

September 13, 2003

Abstract

Reinforcement learning is a hard problem and the majority of the existing algorithms suffer from poor convergence properties for difficult problems. In this paper we propose a new reinforcement learning method, that utilizes the power of global optimization methods such as simulated annealing. Specifically, we use a particularly powerful version of simulated annealing called Adaptive Simulated Annealing (ASA) [3]. Towards this end we consider a batch formulation for the reinforcement learning problem, unlike the online formulation almost always used. The advantage of the batch formulation is that it allows state-of-the-art optimization procedures to be employed, and thus can lead to further improvements in algorithmic convergence properties. The proposed algorithm is applied to a decision making test problem, and it is shown to obtain better results than the conventional Q-learning algorithm.

1 Introduction

Reinforcement learning (RL) is the theory of learning that acts to maximize some measure of future payoff or reward [10]. Usually each reward is affected by all actions taken in the past, and that presents a fairly challenging problem involving aspects of Markov processes and dynamic programming. The RL problem is characterized by a state transition function that probabilistically specifies the next state as a function of the current state and the "action" taken. The actions represent a "control decision" input. It is required to find the control strategy that maximizes the expected future aggregate reward. Because the control decision affects the instantaneous reward, as well as the state transition that will in turn affect future rewards, the resulting optimization problem is by no means simple to solve.

Available RL algorithms are in no means adequate. Theoretical studies prove convergence in only a few narrow special cases (see [14], [8]). Practical experience indicates that they generally do not achieve the Bellman optimality condition (that is the globally optimal solution, see [1]). We are here proposing a new RL method which aims at improving the achieved solution, and at avoiding the susceptibility to local minima. We use the well-known Adaptive Simulated Annealing algorithm (ASA), which has been shown in many studies as very effective in reaching the global minimum.

The paper is organized as follows. In the next section we present an overview of RL theory and an overview of some of the main existing methods. Section III briefly reviews the Adaptive Simulated Annealing algorithm, and Section IV presents the proposed method. Section V gives a simulated test for the new method, and Section VI presents the conclusions of this paper.

2 Introduction to Reinforcement Learning

A measure of the aggregate future rewards used in RL is the so-called *value function*. It represents the main objective function for the RL problem. Typically, at each time instant the value function is estimated and the action that maximizes the value function is taken. In what follows we will here very briefly review the problem of value function estimation. We will be following a similar terminology as used in [15].

Consider a dynamic system given by

$$x_{t+1} = f(x_t, a_t, n_t), \tag{1}$$

where a_t is the control decision or "action", and n_t is the disturbance (assume successive independence). Every decision a_t leads to the instantaneous reward $r(x_t, a_t)$. Based on the current value of the state, it is required to find a control decision that maximizes some measure of aggregate future reward. Mathematically speaking, one needs to obtain a mapping $a_t = \phi(x_t)$ so as to maximize the *value function*:

$$V(x, \phi) = E\Big[\sum_{t=0}^{\infty} \beta^t r(x_t, \phi(x_t)) \Big| x_0 = x\Big], \tag{2}$$

where $\beta \in [0, 1)$ is a discount factor.

The optimal control strategy ϕ^* is the one that maximizes the value function. This means that

$$\phi^*(x) = \mathrm{argmax}_{\phi}(V(x, \phi)). \tag{3}$$

Let the associated optimal value function be $V^*(x)$. A result of Bellman's theory of dynamic programming states that

$$\phi^*(x) = \mathrm{argmax}_a E_n\Big(r(x, a) + \beta V^*(f(x, a, n))\Big), \tag{4}$$

$$V^*(x) = \max_a E_n\Big(r(x, a) + \beta V^*(f(x, a, n))\Big), \tag{5}$$

where E_n denotes the expectation with respect to the disturbance term n.

A strategy that successively refines the estimate of ϕ^* by iteratively applying Eq. 4 is called *policy iteration*. Similarly a strategy that iterates on Eq. 5 to estimate the value function is called *value iteration*. Using these methods to generate the decision strategy is prohibitive from a computational point of view, because of the exponential nature of evaluating $\phi^*(x)$ or $V^*(x)$ for every possible value of x in a look-up fashion. As a result, a new family of algorithms have been proposed that model the value function in a parametric fashion. The parameters are iterated so as to successively improve the value function estimates. Among this family of algorithms are the well-known temporal difference learning (TD) [13]. A related method is the Q-learning algorithm. It was proposed by Watkins [16]. A modified version was simultaneously developed by Lukes, Thompson and Werbos [5], [17]. In this algorithm a "Q-function" rather than a value function is approximated. The Q-function, $Q(x, a)$, is essentially the value function when taking any given control decision a at the current state x, then continuing by choosing the control decisions optimally. The parameters of the Q-function are updated so as to minimize the "temporal difference", or the difference between the estimated value function at time t and an improved version of it that incorporates the information available next time step, including the reward $r(x, a)$. Since the development of TD learning and Q-learning, several other methods were proposed in the literature, some focusing on optimizing with respect to the action parameters directly (e.g. [6], [7], [11], [18]), others using novel optimization formulations (e.g. [2]'s support vector machines' linear programming formulation), and some using concepts from backpropagation training [9].

3 The Adaptive Simulated Annealing Algorithm

Simulated annealing is a well-known optimization method for finding the global optimum, developed by Metropolis et al [12]. The basic idea of the method is to sample the space using a Gaussian distribution. The standard deviation of the sampling distribution is "annealed" with time, as we hone in on the optimum. Simulated annealing was very effective in discovering the global optimum, but its problem has been the slow convergence. To alleviate that Ingber [3] has developed a much faster method of simulated annealing called *Adaptive Simulated Annealing (ASA)*. It is based on sampling the space using a specific fat-tailed distribution. This allows far-reaching access of the state space, and allows much faster annealing and hence faster convergence. Since then, ASA has been the simulated annealing method of choice for many global optimization applications.

The default parameter-sampling distribution permits temperature schedules exponentially faster than a Cauchy distribution, which is in turn exponentially faster than a Gaussian/Boltzmann distribution, according to each distribution's proof for (weakly) ergodic sampling of the parameter space. Furthermore, since nonlinear systems typically are non-typical, the many OPTIONS broken out for user tuning makes ASA very powerful across many classes of nonlinear problems. Some of these ASA OPTIONS permit methodical "quenching" versus "annealing" to explore possible efficiencies for a given system. This is quite different from most other simulated-annealing algorithms, which at their inception resort to quenching – denying their own proof of proper annealing, e.g., with exponential temperature schedules – just to move their algorithm faster, of course at the expense of denying proper sampling to achieve global optimization.

4 Proposed Reinforcement Learning Method

Assume $V(x, w)$ is a parametrized approximation to the value function, with $w = (w_1, ..., w_K)^T$ being the parameter vector, and x being the state vector. Similar to the Q-learning approach, we use a linear model of some basis functions:

$$V(x, w) = \sum_{k=1}^{K} w_k g_k(x). \tag{6}$$

We assume that given a value function approximation, the optimal control can be obtained easily. This is usually the case, because typically the dimensionality of the control variables is much less than that of the state space. Required is to find the optimal parameters w_k that lead to the most accurate value function estimation, in other words the highest value function. We use ASA as follows:

1) Run the application forward, generating a "training set", or a sequence of states $x_t, t = 1, ..., T$. It is preferrable to have multiple sequences.

2) Sample the w_k 's according to ASA.

3) Evaluate the value function according to Eq. 5 sequentially with time, where at each time step, we employ the optimal control action according to Eq. 4.

4) Repeat the sampling and the evaluation steps 2), 3) many times according to the ASA sampling scheme, where the realized value or reward (RV) represents the objective function to be maximized, given by

$$RV = \sum_{t=1}^{T} \beta^t r(x_t, \phi(x_t)) \tag{7}$$

for each sequence. Upon termination the w_k's with its associated value function (Eq 6) represent the optimal solution.

As can be inferred, he proposed method is more of a value iteration method. It exhibits several advantages compared to the Q-learning method, as follows:

1) First and foremost, it is a method designed to reach the global minimum, and so it is expected to obtain superior solutions.

2) It has the flexibility to use fairly sophisticated or nonlinear value function models. We used here a linear model (Eq 6) for illustration only. We could use a nonlinear

model such as a neural network (we mean nonlinear in the parameters). With a nonlinear model, Q-learning will have a tough time producing meaningful results, but with ASA, it would not present a problem albeit possibly a little slower value function computation per iteration.

3) The algorithm also allows for *exploration.* Instead of choosing the maximum action as in Eq. 4, the action can be generated according to a Boltzmann probability, as follows:

$$P(a_\tau = \alpha_i) = \frac{e^{[r(\alpha_i, x_\tau) + \beta V(x_{\tau+1}, w)]/T_\tau}}{\sum_j e^{[r(\alpha_j, x_\tau) + \beta V(x_{\tau+1}, w)]/T_\tau}}, \tag{8}$$

where α_i represents the possible values that the control action a_τ can take, and $x_{\tau+1}$ is a state vector generated for the next time step, conditional on action α_j being taken (the dependence on j is skipped for brevity). The parameter T_τ represents the temperature which should be decreased gradually with iteration. This version of the algorithm allows exploring otherwise inaccessible portions of the state space In fact the exploration framework provides some kind of "annealing" that is the spirit of the other annealing performed in value function maximization.

5 Implementation example

We have tested the proposed algorithm on the following sequential decision making problem. A source generates a sequence of random numbers according to an unknown distribution. An agent receives these numbers and attempts to choose a number as high as possible. At each time step the agent decides to either stop and choose the currently received number or wait for the next time step in hope for a larger number. If the agent forgoes choosing a number, he cannot choose it again at a later time step (it's gone). The reward is exponentially discounted as in standard RL problems. The more the agent waits, the more accurate the statistics that are collected on the unknown distribution, and hence the agent can make a more informed decision. However, an offsetting factor is the punishing effect of the discounting as time goes by. This problem arises in the theory of decision making, where one has to compromise between waiting to get more information and losing value and opportunity as the decision gets delayed.

We consider a collection of sources which produce normally distributed numbers. The mean and standard deviation of the generated numbers from each source are unknown. However, the means and the standard deviations of the different sources are generated from a distribution uniform in $[0, 1]$ and in $[0, 0.5]$, respectively. We considered 100 sources, and for each source we generated a sequence of numbers of length 1000. This represents our training set.

Table 1: Comparison between the new method and the Q-learning method (for various learning rates)

Method	Achieved Obj Fn Val
New Method	0.7569
Q-Learning $\eta = 0.002$	0.6528
Q-Learning $\eta = 0.005$	0.6836
Q-Learning $\eta = 0.02$	0.6176

We considered a state vector with four components: the running mean (the mean up to the current time), the running standard deviation, the maximum (of the previous sequence up to current time), and the time, each properly scaled. We modelled the value function as a linear function using Gaussian basis functions whose centers are located at scattered parts of the state space. We took 10 Gaussian basis functions, so our optimization parameter is the 10-dimensional vector w. We used the ASA code downloadable at [4] and modified it for our purposes. Table 1 shows a comparison between the new method and the Q-learning method. The performance measure indicated is the discounted objective function as defined in Eq. 7. One can see that the proposed ASA method achieves a significantly better solution (10.72% improvement in objective function). However, if we factor out the fact that the most trivial decision method of choosing the first generated number achieves 0.5, then the improvement of the new method over Q-learning seems more significant relatively speaking. It should be noted that even though that there is a common belief that simulated annealing methods are slow, this is not the case here. Within a few seconds we got the solution. The advantage of ASA (over other simulated annealing methods) is the fast annealing that allows faster convergence.

6 Conclusion

A new algorithm for the RL problem is proposed. It is based on the adaptive simulated annealing algorithm. Implementation on a decision making test example shows that the method achieves superior performance compared to the well-known Q-learning method. The proposed approach is flexible enough to tackle sophisticated nonlinear value function models, as well as exploration models. These, however, are not yet tested, and this will be considered in future work.

Acknowledgment

The second author would like to thank the financial support provided by the State of Texas Advanced Technology Program, Grant $512-0225-2001$, and the National Science Foundation Grants No. $CMS-0100238$ and $CMS-0097719$.

References

[1] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[2] T. Dietterich and X. Wang, "Batch value function approximation via support vectors",

[3] L. Ingber "Very fast simulated re-annealing", *Mathematical Computer Modelling*, Vol. 12, No. 8, pp. 967-973, 1989.

[4] http://www.ingber.com/#ASA.

[5] G. Lukes, B. Thompson, and P. Werbos, "Expectation driven learning with an associative memory", in *Proceedings International Joint Conference on Neural Networks*, pp. I:521-524, 1990.

[6] J. Moody, L. Wu, Y. Liao and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios", *J. Forecasting*, Vol. 17, pp. 441-470, 1998.

[7] J. Moody and M. Saffell, "Learning to trade via direct reinforcement", *IEEE Trans. Neural Networks: Special Issue: Neural Networks in Financial Engineering*, Vol. 12, No. 4, pp.875-889, July 2001.

[8] A. Poznyak, K. Najim, and E. Gomez-Ramirez, *Self-Learning Control of Finite Markov Chains*, Marcel Dekker, New York, 2000.

[9] D. Prokhorov and D. Wunsch, "Adaptive critic designs", *IEEE Transactions Neural Networks*, Vol. 8, pp. 997-1007, Sep 1997.

[10] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[11] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation", *Advances in Neural Information Processing Systems 12*, pp. 1057-1063, MIT Press, 2000.

[12] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines", *J. Chem. Phys.*, Vol. 21, No. 6, pp. 1087-1092, 1953.

[13] R. Sutton, "Learning to predict by the method of temporal differences", *Machine Learning*, Vol. 3, No. 1, pp. 9-44, 1988.

[14] J. Tsitsiklis and B. Van Roy, "An analysis of temporal difference learning with function approximation", *IEEE Transactions Automatic Control*, Vol. 42, No. 5, pp. 674-690, 1997.

[15] B. Van Roy, "Neuro-dynamic programming: overview and recent trends", in *Handbook of Markov Decision Processes: Methods and Applications*, E. Feinberg and A. Schwartz, Eds., Kluwer, 2001.

[16] C. Watkins, *Learning from Delayed Rewards*, Ph.D. Thesis, Cambridge University, UK, 1989.

[17] P. Werbos, "Approximate dynamic programming for real-time control and neural modeling", in *Handbook of Intelligent Control*, D. White and D. Sofge, Eds., 1992.

[18] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning", *Machine Learning*, Vol. 8, pp. 229-256, 1992.