Temporal Difference Learning and Simulated Annealing for Optimal Control: A Case Study

Jinsong Leng, Beulah M. Sathyaraj, and Lakhmi Jain

School of Electrical and Information Engineering,
Knowledge Based Intelligent Engineering Systems Centre,
University of South Australia, Mawson Lakes SA 5095, Australia
Jinsong.Leng@postgrads.unisa.edu.au,
Beulah.Moses@postgrads.unisa.edu.au,
Lakhmi.Jain@unisa.edu.au

Abstract. The trade-off between exploration and exploitation has an important impact on the performance of temporal difference learning. There are several action selection strategies, however, it is unclear which strategy is better. The impact of action selection strategies may depend on the application domains and human factors. This paper presents a modified Sarsa(λ) control algorithm by sampling actions in conjunction with simulated annealing technique. A game of soccer is utilised as the simulation environment, which has a large, dynamic and continuous state space. The empirical results demonstrate that the quality of convergence has been significantly improved by using the simulated annealing approach.

Keywords: temporal difference learning, agent, convergence, simulated annealing.

1 Introduction

An intelligent agent is expected to be capable of adapting and learning in an uncertain environment. The ability of learning can be built up by interacting with the environment in which it is situated. Reinforcement learning is a proper approach for an agent to learn from experience.

Reinforcement learning is the learning of a mapping from situations to actions so as to maximise a scalar reward or reinforcement signal [19], which mainly includes Dynamic Programming (DP), Monte Carlo Method, and Temporal Difference Learning technique (TD). DP [4,8] has been well investigated under the formalism of Markov Decision Processes (MDPs). TD [18] is a form of asynchronous DP, and the value function is estimated by sample episodes. TD method addresses the problem of approximating the optimal action strategy through and while interacting with the environment.

Trial–and–error, delayed rewards, and trade-off between exploration and exploitation are three important features in TD algorithm. As a kind of unsupervised learning, TD has to balance exploration and exploitation during the learning period. To effectively build up the reward structure, the strategy dealing with the trade-off between exploration and exploitation has to be considered. Some strategies have been used for balancing exploration and exploitation, e.g., greedy strategies, randonmised strategies, annealing-like strategies. It is unclear which policy is better and the performance may depend on the task and on human factors. Only few careful comparative studies are available [19].

The simulated annealing (SA) [15] is a technique for solving combinatorial optimisation problems. SA starts with a high temperature and gradually decreases the temperature over time. As a consequence, the agent starts with the high possibility of exploration and turns towards exploitation by incorporating a probability function in accepting or rejecting new solutions. SA approach does not require large computer memory, which can speed up the computation.

In this paper, the simulated annealing technique is utilised for sampling possibility distribution over actions at a given state, so as to improve the quality of performance. A novel algorithm Sarsa(λ) is proposed by combining on-policy learning algorithm with SA method. A comparative study is conducted in a real-time, stochastic, and dynamic testbed called SoccerBots [1]. The simulation results are compared by using ϵ-greedy policy and ϵ-greedy policy with simulated annealing, in conjunction with a linear approximation function called Tile Coding [2]. The experimental results demonstrate that the quality of the performance can be enhanced by using the simulated annealing technique.

The rest of the paper is organised as follows: Section 2 introduces the TD technique and simulated annealing method. The simulation environment and algorithm are detailed in Section 3. The empirical results are analysed and discussed in Section 4. Section 5 presents the related work. Finally, we discuss future work and conclude the paper.

2 Background

2.1 Temporal Difference Learning

TD is a form of model-free approach and is a combination of DP and Monte Carlo Method [18,19]. The idea of TD is to learn how to take action through trial-and-error interactions in a dynamic environment. The value function is accumulated by an incremental learning model and infinite horizon discount model.

The incremental learning model accumulates the value function based on temporal difference errors, as given in Equation (1):

$$V_{t+1} = V_t + \alpha \left[R_{t+1} - V_t\right] \tag{1}$$

where the parameter α is learning rate, $0 \leq \alpha \leq 1$, R_{t+1} is the accumulated reward.

Based on the infinite horizon discount model, the one-step update at a given time t and state s is shown in equation (2).

$$V_{t+1}(s) = V_t(s) + \alpha \left[r_{t+1}(s) + \gamma V_{t+1}(s') - V_t(s)\right] \tag{2}$$

where the parameter γ is the discount rate, $0 \leq \gamma \leq 1$. The rewards in the future are geometrically discounted by the parameter γ, r_{t+1} is the immediate reward.

The use of eligibility traces is another mechanism to improve the convergence of value function [11,18]. The aim of eligibility traces is to assign credit or blame to the eligible states or actions. The ways for evaluating the traces include accumulating traces and replacing traces.

The accumulating traces are given by:

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a), & \text{if } s \neq s_t \\ \gamma\lambda e_{t-1}(s,a) + 1, & \text{if } s = s_t \end{cases} \tag{3}$$

whereas replacing traces use $e_t(s,a) = 1$ for the second update [19].

The trade-off between exploration and exploitation is a problem the learning agent has to deal with. On one hand, the agent must explore the environment explicitly, in case to fall into local optimum. On the other hand, the agent need to exploit the existing knowledge to make the value function converge quickly. However, how can the agent know that the sequence of actions that has been found is the best? It may be helpful to integrate some parametric optimisation techniques with reinforcement learning.

Normally, there are some action selection strategies [19]: e.g., ϵ-greedy policy is to select a random action with probability ϵ and the optimal action with probability 1-ϵ. The drawback is that unlucky sampling may cause the rewards obtained from optimal action to fall into a local optimum. On the another hand, randomised strategy is to select the action according to probability p, which is sampled by Boltzmann distribution. Yet, another strategy is the annealing-like approach, which is used in this paper.

2.2 Simulated Annealing

SA [10] is based on the theory of Markov processes and motivated by the physical process of annealing [15] in solids. The solids are heated to a temperature above the melting point and then allowed to cool. When the solid is cooled to a lower temperature, the solid reaches the stable crystalline structure. If the cooling is done slowly then the crystalline structure is stable. However, if the cooling is done fast then the obtained solid has crystalline imperfection. Therefore, for the system to reach equilibrium, the cooling has to be done slowly. Slower does not mean it can be done forever to obtain stable state. Simulated Annealing is analogous to this process.

The SA algorithm searches the search space similar to the thermodynamic change of energy-to-energy state. First random position in the search space is chosen to be the initial state or current state and another random position is chosen as the next state. The value functions at both the current state and the next state are evaluated. These value functions are compared. If the next state yields a better solution then it is chosen as the best solution. Even otherwise, the next state is chosen with a probability. This probability of acceptance of worse solution is P (accept) which is expressed as given in Equation 4.

$$P(accept) = exp(-c/T) < r \tag{4}$$

Where c = $\triangle$E is the change in the evaluation function, T is the current temperature and r is a random number between 0 and 1. It is not required to stick on to a particular solution, which could lead to locking up in local optimisation rather than global optimisation. The problem of getting locked to local optimum is overcome in SA by letting the possibility of the choice of accepting the worse solution with a certain probability of P(accept).

The cooling schedule or temperature schedule T of SA is how the temperature is decremented and is the mapping of the temperature to time. It depends on starting temperature, final temperature, decrement in temperature (linear or nonlinear) and number of iterations at each temperature. There are various types of cooling schedules available that are used in practice.

SA Algorithm [16] has been used to solve many combinatorial optimisation problems and some continuous variable problems. The choice of the temperature schedule or the cooling schedule, can be either linear or non-linear.

The algorithm is given in Fig. 1:

```
1.    Current node = MAKE-NODE (INITIAL-RANDOM
        STATE [Problem, Schedule[t]]).
2.    For t = 1 to $\infty$ do:
3.       If T = 0 then return Current;
4.        Next node = a randomly selected successor of Current;
5.        $\triangle$E = VALUE[Next] - VALUE[Current]
6.       If $\triangle E \geq 0$ then Current = Next;
7.       Else Current = Next only with probability exp(-$\triangle$E/T) ;
```

Fig. 1. Basic Algorithm of Simulated Annealing

The inner loop of Simulated Annealing algorithm need to be run as long as the search space; where the nodes are present are explored rather than exploring the entire search space. The value of the current and the next solution is evaluated and the difference is $\triangle$E. When the difference is greater than 0, then the next solution is accepted, else the next solution is accepted with a probability of P(accept) = exp(-$\triangle E/T$), where T is the current temperature.

3 Details of Simulation Environment and Algorithms

TeamBots is a Java-based collection of application programs and Java packages for multiagent mobile robotics research [1]. Each soccer team can have no more than 5 players. Two teams are built and the individual and team strategies for each team are defined.

The scenario is defined to learn a ball interception skill for the soccer player in SoccerBots. For the ball interception, the ball is kicked at a certain angle and speed. The player is away from the ball at a certain distance and angle to ball, in order to intercept the ball with the highest speed and the fewest steps. The intercepting ball problem is that a soccer agent is trained to find the optimal interception direction at each step toward the ball in order to catch an approaching ball in the shortest time.

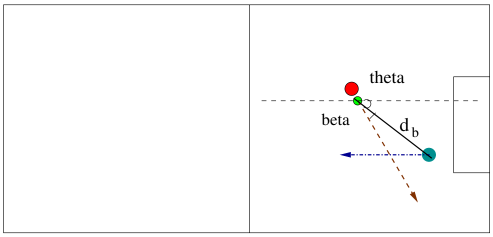

Fig. 2. Ball Interception Problem

In addition, the state space reduction technique is necessary for a large, stochastic, and dynamic environment. In this paper, a linear approximation function known as tile coding [2] is utilised to avoid state space from growing exponentially [12].

The details of modified Sarsa(λ) with replacing traces is given in Fig. 3.

In Fig. 3: α is a learning rate, γ is a discount rate. The action selection policy is the exploration strategy, which can be ϵ-greedy policy, i.e. the agent takes a random action with probability ϵ and takes the best action with probability (1 - ϵ). Q(s,a) is the state-action value, δ and θ are matrices to represent action, feature and related feature weights, and $\mathcal{F}$ is the feature to trace the eligible state process. The parameter T is a temperature, and ψ is the temperature-decay factor. Lower temperatures cause a greater difference in selection probability for actions. In the limit as $T = 0$, the action selection becomes the same as greedy action selection.

4 Empirical Study

SA based exploration works similarly to neighborhood search based exploration by searching the set of all possible actions, but reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior actions to be controlled by a randomised scheme [20]. The parameter T is initially high, allowing many inferior actions to be accepted, and is slowly reduced to a value where inferior actions are nearly always rejected. For the annealing-like approach, the temperature T is set to 120 and temperature-decay factor ψ is set to 0.99. The annealing process is illustrated in Fig. 4.

```
1.     Initialise Q(s,a) arbitrarily and e(s, a) = 0, for all s, a.
2.     Repeat (for each episode):
3.         Initialise s, a;
4.         Initialise T ← $\psi$ T;
5.         Repeat ( for each step of episode):
6.1.           Choose $a_p$ in s based on action selection policy (e.g., $\epsilon$-greedy);
6.2.           Choose $a_r$ in s at random;
6.3.           Generate an random possibility $P_{random}$;
6.4.           Calculate $Q(s, a_r)$ and $Q(s, a_p)$;
6.5.           Calculate the possibility of accept $P_{accept}$ using $exp((Q(s, a_r) - Q(s, a_p))/T)$;
6.6.           If $Q(s, a_r) \geq Q(s, a_p)$;
6.7.            a ← $a_r$;
6.8.           Else If $P_{random} \leq P_{accept}$;
6.9.            a ← $a_r$;
6.10.          Else
6.11.            a ← $a_p$;
7.             Take action a, observe r, s';
8.             Choose a' from s' using action selection policy derived from Q ;
9.             $\delta$ ← r + $\gamma$Q(s', a') - Q(s, a);
10.            e(s, a) ← 1;
11.            For all s, a:
12.                Q(s, a) ← Q(s, a) + $\alpha\delta$e(s, a);
13.                e(s, a) ← $\gamma\lambda$e(s, a);
14.            s ← s';
15.        until s is terminal.
```

Fig. 3. Sarsa(λ) Control Algorithm with Replacing Traces

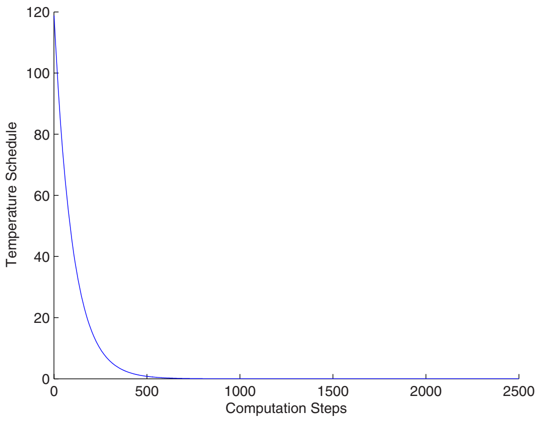

Fig. 4. Temperature Schedule and Computation Steps

The initial start temperature T, the temperature-decay factor ψ, and final temperature T' have to be scaled properly, otherwise, the poor results may occur. In [21], the choice process of their values includes to estimate the mean of the distribution of state values to define a maximum energy scale of the system, its standard deviation to define the maximum temperature scale, and the minimum change in energy to define the minimum-temperature scale.

To compare the realistic performance, the simulation is conducted with different set of parameter values with a number of experiments [13,14]. By setting ϵ to 0.1, the set of optimal parameter values can be found: $\alpha = 0.005$, $\gamma = 0.93$, and $\lambda = 0.9$. The performance may be heavily influenced by the action selection strategy. By running the episodes 2500 times, the average rewards can be obtained using algorithm of Fig. 3 with ϵ-greedy policy. The average rewards can be generated using algorithm in Fig. 3 with annealing-like policy. By comparing the average reward at different episodes, it is clear that the convergence with annealing-like policy is much quicker than that with ϵ-greedy policy.

Table 1. Performance Comparison: Accumulated Average Rewards

Episodes	400	600	800	1000	1200	1400	1600	1800	2000	2200	2400
ϵ-greedy	-28.79	-25.96	-24.25	-23.38	-23.2	-22.6	-22.36	-22.22	-22.07	-21.8	-21.72
SA	-25.98	-23.92	-22.79	-21.96	-21.47	-21.36	-21.07	-20.84	-20.67	-20.59	-20.57

The curves in Fig. 5 illustrate that annealing-like policy can significantly improve the performance, both in speed of convergence and eventually convergence.

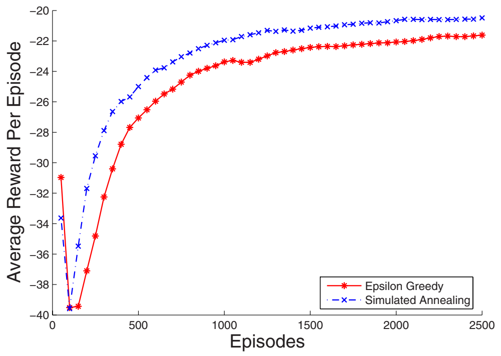

Fig. 5. Convergence Comparison of *epsilon*-Greedy and Simulated Annealing(SA)

5 Related Work

SA is a technique to find the good solution to an optimisation problem by trying random variation of the current solution. SA has been widely applied to optimisation problems [6] by searching for possible solutions and converge at optimal solution. The applications include but are not limited to Travelling

Salesman Problem (TSP), Capacited Vehicle Routing Problems (CVRP), Job Scheduling Problems, Timetabling Problems, and Selection of Communication Protocols/Standards and various other applications [5].

A version of simulated annealing called Adaptive Simulated Annealing is employed with the reinforcement learning algorithm, which shows further improvements in algorithmic convergence properties [3]. A Q-learning algorithm with simulated annealing is introduced to balance exploration and exploitation [7]. In [20], three action selection methods, i.e., neighborhood search based exploration, simulated annealing based exploration, and tabu search based exploration, are evaluated and compared on a discrete reinforcement learning task (robot navigation).

Softmax or Boltzmann distribution Action Selection. A combined use of reinforcement learning and simulated annealing is proposed in [17]. A theoretically established approach tailored to reinforcement learning following Softmax action selection policy are discussed. It has been proven that Boltzmanns formula converges to uniform distribution as T goes to infinity and to the greedy distribution as T goes to 0. In addition, an application example of agent-based routing will also be illustrated.

Another work [3] on adaptive simulated annealing(ASA) based reinforcement learning method is proposed. Here ASA [9] allows far-reaching access of the state space, and permits much faster annealing and hence faster convergence. The action can be generated according to a Boltzmann probability, which provides some kind of ”annealing” that is the spirit of the other annealing performed in value function maximisation.

The ϵ-greedy Action Selection with SA. SA based exploration is discussed in [20]. The ϵ-greedy algorithm is a method using near-greedy action selection rule. It behaves greedily (exploitation) most of the time, but every once in a while, say with small probability ϵ (exploration), instead select an action at random. This paper evaluates the role of heuristic based exploration in reinforcement learning. Three methods are compared: neighborhood search based exploration, simulated annealing based exploration, and tabu search based exploration. SA based exploration works by searching the set of all possible actions, but reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior actions. When a non-greedy action is selected, this action is evaluated by SA based exploration approach.

A similar work done in [7] is to explore the possibility of improving the simple ϵ-greedy approach by appropriately reducing ϵ during the learning process. The SA approach is combined with Q(λ). The task of finding the optimal policy in Q-learning is transformed into search for an optimal solution in a combinatorial optimisation problem. Then the Metropolis criterion from SA algorithm is applied to the search procedure in order to control the balance between exploration and exploitation. The improved algorithm is tested in the puzzle simulation domain.

6 Conclusion and Future Work

The parametric optimisation techniques can be combined with TD algorithms to improve the overall performance. This paper provides a comparative study by using the simulated annealing technique to balance between exploration and exploitation. The experimental results demonstrate that the algorithm in Fig. 3 with the annealing-like approach converges much quicker. For a large, stochastic, and dynamic system, utilising the annealing-like technique can reduce computational cost and learn quickly.

Future work includes the comparison of softmax action selection and simulated annealing technique. The ultimate goal is to develop a methodology for adaptively selecting the parameter values in the learning algorithm. In addition, the algorithm will be extended to soccer agents teaming to solve the cooperative learning problems.

References

1. Teambots (2000), `http://www.cs.cmu.edu/~trb/Teambots/Domains/SoccerBots`
2. Albus, J.S.: A Theory of Cerebellar Function. Mathematical Biosciences 10, 25–61 (1971)
3. Atiya, A.F., Parlos, A.G., Ingber, L.: A Reinforcement Learning Method Based on Adaptive Simulated Annealing. In: Proceedings of the 46th IEEE International Midwest Symposium on, pp. 121–124 (2003)
4. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
5. Chaharsooghi, S.K., Jafari, N.: A Simulated Annealing Approach for Product Mix Decisions. Scientia Iranica 14(3), 230–235 (2007)
6. Dowsland, K.A.: Simulated Annealing. In: Modern Heuristic Techniques for Combinatorial Problems (1995)
7. Guo, M., Liu, Y., Malec, J.: A New Q-learning Algorithm Based on the Metropolis Criterion. Systems, Man and Cybernetics, Part B, IEEE Transactions on 34(5), 2140–2143 (2004)
8. Howard, R.A.: Dynamic Programming and Markov Processes. MIT Press, Cambridge (1960)
9. Ingber, L.: Very Fast Simulated Re-annealing. Mathematical Computer Modelling 12(8), 967–973 (1989)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)
11. Klopf, A.H.: Brain Function and Adaptive Systems–A Heterostatic Theory. Technical report, AFCRL–72–0164, Air Force Cambridge Research Laboratories, Bedford, MA (1972)
12. Leng, J., Fyfe, C., Jain, L.: Reinforcement Learning of Competitive Skills with Soccer Agents. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, Springer, Heidelberg (2007)
13. Leng, J., Jain, L., Fyfe, C.: Simulation and Reinforcement Learning with Soccer Agents. Journal of Multiagent and Grid systems, IOS Press, The Netherlands 4(4) (to be published, 2008)
14. Leng, J., Jain, L., Fyfe, C.: Convergence Analysis on Approximate Reinforcement Learning. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, pp. 85–91. Springer, Heidelberg (2007)

15. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of State Calculations by Fast Computing Machines. J. Chem. Phys. 21, 1087–1092 (1953)
16. Russel, S., Norwig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (2003)
17. Stefán, P., Monostori, L.: On the relationship between learning capability and the boltzmann-formula. In: Monostori, L., Váncza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS (LNAI), vol. 2070, pp. 227–236. Springer, Heidelberg (2001)
18. Sutton, R.S.: Learning to Predict by the Method of Temporal Differences. Machine Learning 3, 9–44 (1988)
19. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
20. Vien, N.A., Viet, N.H., Lee, S., Chung, T.: Heuristic Search Based Exploration in Reinforcement Learning. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 110–118. Springer, Heidelberg (2007)
21. White, S.R.: Concepts of scale in simulated annealing. In: AIP Conference Proceedings, vol. 122, pp. 261–270 (1984)