# A New Q-Learning Algorithm Based on the Metropolis Criterion

Maozu Guo, Yang Liu, Jacek Malec

**Abstract**

The balance between exploration and exploitation is one of the key problems of action selection in Q-learning. Pure exploitation causes the agent to reach the locally optimal policies quickly, whereas excessive exploration degrades the performance of the Q-learning algorithm even if it may accelerate the learning process and allow avoiding the locally optimal policies. In this paper, finding the optimum policy in Q-learning is described as search for the optimum solution in combinatorial optimization. The Metropolis criterion of simulated annealing algorithm is introduced in order to balance exploration and exploitation of Q-learning, and the modified Q-learning algorithm based on this criterion, SA-Q-learning, is presented. Experiments show that SA-Q-learning converges more quickly than Q-learning or Boltzmann exploration, and that the search does not suffer of performance degradation due to excessive exploration.

*Keywords*

Reinforcement learning; Q-learning; The Metropolis criterion; Exploration; Exploitation

## I. Introduction

Reinforcement learning (RL) is one of the most rapidly developing machine learning methods in recent years [1], [2]. It includes the temporal difference algorithm proposed by Sutton [3] and the Q-learning of Watkins [4], [5]. Those algorithms have been extensively used in many applications such as industry control, time sequence prediction [6], robot soccer competition [7], and many more.

However, finding the proper balance between exploration and exploitation in Q-learning is one of the major issues requiring further attention. Exploitation occurs if the action selection strategy is based purely on current values of the state-action pairs (SAPs), i.e., when the selection is greedy. In case of most of the optimization problems this will lead to locally optimal policies, possibly differing from a globally optimal one. In contrast, exploration is the strategy based on the assumption that the agent

Maozu Guo and Yang Liu are with the Department of Computer Science and Engineering, Harbin Institute of Technology, Harbin 150001, People's Republic of China. E-mail: maozuguo@hit.edu.cn.

Jacek Malec is with the Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden. E-mail: jacek@cs.lth.se.

selects a non-optimal action in the current situation and obtains more knowledge about the problem. This knowledge allows it to neglect the locally optimal policies, and to reach the globally optimal one instead. On the other hand, excessive exploration will drastically decrease the performance of a learning algorithm, and in some cases might be even harmful with respect to the learning results themselves [8].

The balance between exploitation and exploration has been investigated previously. A simple strategy proposed to deal with this problem is the $\epsilon$-*greedy* [8] (with $0 \leq \epsilon < 1$), with larger $\epsilon$ corresponding to larger probability of exploration. Here, the value of $\epsilon$ has obviously a great impact on the algorithm. Sutton and Barto compared the performance of learning for different $\epsilon$ values and concluded that the result for a non-zero $\epsilon$ is usually better than that for $\epsilon$ equal 0 (i.e., the blindly greedy case), which means that $\epsilon$-greediness is effective. However, excessive exploration becomes unnecessary after a period of an initial interaction between the agent and the environment (assuming, of course, that the state-action-pair values are constant). As reinforcement learning is mainly used as an online learning method, excessive exploration will unavoidably result in the decreasing performance of a RL algorithm.

An overview of strategies addressing the balance problem has been presented by Kaelbling, Littman and Moore [9]. The first three, formally justified techniques are dynamic-programming, Gittins allocation indices and learning automata. However, the expense of dynamic-programming approach is exponential, Gittins allocation indices method applies only under the discounted expected reward criterion with immediate reward, and learning automata approach does not necessarily converge to the correct action. Besides, all three methods provide formal guarantees only for the case of immediate reward and are not extensible in an obvious way to the delayed reinforcement case. The other three strategies, labeled *ad hoc*, and not formally justified, are greedy approaches, randomized strategies (including Boltzmann exploration), and interval-based techniques. However, unmodified greedy strategies will necessarily pick suboptimal actions, randomized strategies get no direction in exploration, Boltzmann exploration, while ameliorating this problem, suffers when the values of the actions are not separated enough, moreover it converges unnecessarily slowly if not properly tuned, and finally, the interval-based techniques require the algorithm to store statistics for each action.

Therefore, we have decided to explore the possibility of improving the simple $\epsilon$-greedy approach by appropriately reducing $\epsilon$ during the learning process. This will not only improve the ability of the agent to acquire new knowledge, but will also allow the algorithm to avoid performance decrease due to the constant value of $\epsilon$ (and thus constant probability of exploration). A solution to a similarly posed problem may be found in the simulated annealing algorithm, where a local change of a solution to a combinatorial optimization problem is based on Metropolis criterion [10]. In this paper, the task of finding the optimal policy in Q-learning is transformed into search for an optimal solution in a combinatorial optimization problem. Then the Metropolis criterion from simulated annealing algorithm is applied to the search procedure in order to control the balance between exploration and exploitation. Hence, in the execution process of the proposed algorithm, the exploration will gradually decay, leading to convergence towards the optimum. After introducing the proposed method we present some experimental results confirming the hypothesis. The paper ends with conclusions and suggestions for future work.

## II. Simulated Annealing and Q-learning

### A. Simulated Annealing Algorithm

Simulating the annealing process of solids, the *simulated annealing* algorithm (SA) is one kind of the computational processes resembling nature [11] and has been shown to be an effective approximate algorithm to solve combinatorial optimization problems. The Metropolis criterion, the core of the simulated annealing algorithm, derives from the importance sampling method proposed by Metropolis and others in 1953 [10]. Observing the similarity between the annealing process of solids and combinatorial optimization problems, Kirkpatrick introduced the Metropolis criterion into optimization to solve the problem of getting stuck in the local optima [12]. In SA algorithm, the transition probability $P(i \Rightarrow j)$ of the Metropolis criterion is used to decide whether the transition from the current state (solution) $i$ to the new state $j$ occurs. $P(i \Rightarrow j)$ can be defined as follows (in case of a maximization problem):

$$P(i \Rightarrow j) = \begin{cases} 1 & \text{if f(j)} \geq \text{f(i)} \\ \exp\left(\frac{f(j)-f(i)}{t}\right) & \text{otherwise,} \end{cases}$$

where $f(i)$ and $f(j)$ are the values of the cost function of the optimization problem. $f(i)$ and $f(j)$ can be compared to the energy of the states $i$ and $j$, respectively, in an annealing solid. Obviously, the SA algorithm doesn't greedily reject all the suboptimal solutions, and the optimal state can eventually be reached.

### *B. Q-learning Algorithm*

Q-learning, one of the most important reinforcement learning algorithms, was presented by Watkins in 1989 [4]. Another case of reinforcement learning is temporal difference learning (TD) algorithm [5], and some researchers classify Q-learning as a special case of TD learning [13], [8].

One-step Q-learning is a simple algorithm, in which the key formula to modify the Q value is as follows

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

where $Q(s_t, a_t)$ is the value function of the state-action pair $(s_t, a_t)$ at moment $t$. $\alpha$ and $\gamma$ are the learning rate and discount factor, respectively, and $r_t$ is the reward value received as the result of taking action $a_t$ in state $s_t$.

---

*Algorithm 1:* ONE-STEP Q-LEARNING ALGORITHM

1. Initiate arbitrarily all $Q(s, a)$ values;
2. Repeat (for each episode):

(a) Choose a random (initial) state $s$;

(b) Repeat (for each step in the episode):

i. Select an action $a \in A(s)$ according to the policy;

ii. Execute the action $a$, receive immediate reward $r$, then observe the new state $s'$;

iii.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left( r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

iv. $s \leftarrow s'$

Until $s$ is one of the goal states;

Until the desired number of episodes have been investigated.

---

In the original Q-learning algorithm, the greedy strategy with pure exploitation (i.e., to select the optimal action according to the current state-action pair value, the step 2(b)i in One-step Q-learning algorithm) is used. But by employing this strategy it is generally difficult to obtain satisfactory results as the system usually falls into a locally optimal solution even if the learning time is extended. In case of the exploration strategy, the agent is allowed to adopt a non-optimal action in the current situation. For instance, in the $\epsilon$-greedy strategy, in order to balance the contradiction between exploration and exploitation, it is proposed to explore non-optimal actions in the current situation with a fixed probability $\epsilon$. However, the longer the learning process, the more accurate the knowledge of the agent becomes (i.e., the $Q$ values converge to the optimal $Q^*$), and hence continuing exploration with fixed probability must result in the decrease of performance of the system. Therefore, it is necessary to decay the exploration appropriately after some adequate amount of interactions between the agent and the environment.

## III. Q-learning and the Metropolis Criterion

As the accuracy of the agent's knowledge about the environment increases, the proportion of exploration should decrease. Towards this effect, the Metropolis criterion from simulated annealing algorithm is introduced into the action-selection strategy of Q-learning. The resulting algorithm, called *SA-Q-learning*, is presented below.

### A. The Main Principle

One can distinguish two kinds of search for optimum solutions. One is to search for the optimum in the solution space of combinatorial optimization problems (like, e.g., the traveling salesman problem). Another is to find the optimal policy (or path) given an initial state and a goal state in some appropriate state space (e.g., game problems). Reinforcement learning, which belongs to the latter kind, is based on observing the environment changes from a state to another as a result of the actions of an agent; in this process, the value function of the state (or the state-action pair) is calculated so that the optimal policy can be found.

The formal definition of the policy space is given below to describe learning definite policies in the discrete Markov decision process environment. Then the latter kind of problems mentioned above can be compared to the former one so that Q-learning can be regarded as a combinatorial optimization problem and thus it becomes meaningful to introduce the Metropolis criterion into the Q-learning algorithm.

*Definition 1:* In a problem with a discrete finite state space, the *policy space* is the set $P = A(s_1) \times A(s_2) \times \ldots \times A(s_n)$, where $s_i (i = 1, 2, \ldots, n)$ are the possible environment states, $A(s_i)$ are the sets of actions allowed in each of the state $s_i$, respectively. A *problem policy* will be represented as a $n$-tuple $(a_1, a_2, \ldots, a_n)$, with $a_i \in A(s_i)$.

The Definition 1 describes the policy space in case of problems with finite state spaces. It can obviously be easily extended to the case of problems with infinite state spaces. Moreover, the finiteness of the state space does not affect the considerations that follow below.

*Observation 1:* Let $P$ be a policy space for a problem. The change of any element in a $n$-tuple $(a_1, a_2, \ldots, a_n)$ corresponds to the transition from one solution to another in the policy space $P$.

*Proof* Assume that the $i$-th element is changed to be $b_i, b_i \in A(s_i)$. Then the new corresponding $n$-tuple becomes $(a_1, a_2, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_n)$. Hence, $(a_1, a_2, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots, a_n) \in P$ due to Definition 1. □

The *value function* of a policy is used to evaluate the policy. As it is based on Q-values, let us remind the definition here.

*Definition 2:* Let $p \in P$ be a policy. The Q-value for a state-action pair is defined as the expected discounted reward with infinite horizon, i.e.,

$$Q^p(s, a) \stackrel{df}{=} E(R_t | s_t = s, a_t = a) = E(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a),$$

where the actions are chosen according to the policy $p$.

*Definition 3:* Let $p \in P$. The value function $V(p)$ of the policy $p$ is

$$V(p) \stackrel{df}{=} V((a_1, a_2, \ldots, a_n)) = \sum_{k=1}^{n} Q^p(s_k, a_k),$$

i.e., it is the sum of all the Q-values obtained for this policy.

Regarding the reward maximization problem, the policy $p_1$ is considered superior to the policy $p_2$ if $V(p_1) > V(p_2)$.

We will compare the value of a policy to the energy of the micro-cosmic state in solid annealing, and it will be used in the algorithm to decide the probability of the action-selecting, in combination with other parameters, such as the temperature.

*Observation 2:* If the policy $p_1 = (a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$ transits to the policy $p_2 = (a_1, \ldots, a_{i-1}, b_i, a_{i+1}, \ldots$ then the difference between their values is $V(p_2) - V(p_1) = Q(s_i, b_i) - Q(s_i, a_i)$.

*Proof* Directly from Definition 3. □

### *B. SA-Q-learning algorithm*

Unlike in the One-step Q-learning (Algorithm 1), in the SA-Q-learning algorithm, when the agent selects actions, it not only obeys the policy learned so far, but also attempts to explore (according to the parameters such as temperature) by increasing chances of selecting actions other than those adopted by the current (possibly sub-)optimal policy.

---

*Algorithm 2:* SA-Q-LEARNING ALGORITHM

1. Initiate arbitrarily all $Q(s,a)$ values;

2. Repeat (for each episode):

(a) Choose a random (initial) state $s$;

(b) Repeat (for each step in the episode):

i. Select an action $a_r$ in $A(s)$ arbitrarily;

ii. Select an action $a_p$ in $A(s)$ according to the policy;

iii. $a \leftarrow a_p$

iv. Generate random value $\xi \in (0,1)$

v. If $\xi < \exp(\frac{(Q(s,a_r)-Q(s,a_p))}{\text{Temperature}})$ then $a \leftarrow a_r$

vi. Execute the action $a$, receive immediate reward $r$, then observe the new state $s'$

vii.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left( r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

viii. $s \leftarrow s'$

Until $s$ is one of the goal states

(c) Recalculate Temperature by the temperature-dropping criterion.

Until the desired number of episodes has been investigated.

---

Although the temperature-dropping criterion can be in general arbitrary, we use the geometric scaling factor criterion, i.e., $t_{k+1} = \lambda t_k,\ k = 0, 1, 2, \ldots,\ \lambda \in (0.5, 1)$. Usually $\lambda$ is a constant close to 1, in order to guarantee a slow decay of the temperature factor in the algorithm.

In Algorithm 2, no more than one action of the policy is modified in steps 2(b)i–2(b)vi, and this corresponds to the transition from one policy to another according to Observation 1. Therefore, Q-learning algorithm can be regarded as the combinatorial optimization in the policy space. Hence the introduction of the Metropolis criterion is rational.

By Observation 2, the statement 2(b)v in Algorithm 2 replaces the difference of the value functions between the policies with $Q(s, a_r) - Q(s, a_p)$ in order to reduce the amount of computation.

Comparing the description of Algorithm 1, there are only two additional steps in Algorithm 2: the randomized selection of action and the evaluation of the Metropolis criterion. Therefore there is no substantial increase of complexity between them, as the two steps take the constant time to evaluate.

Although [8] concludes that the results for $\epsilon > 0$ in Q-learning based on the $\epsilon$-greedy strategy are often better than those of $\epsilon = 0$, a great number of explorations in case of fixed probability becomes unnecessary and may even be harmful to the system performance as the learning process proceeds. With the introduction of the Metropolis criterion, SA-Q-learning algorithm eliminates the disadvantage of the probability $\epsilon$ remaining constant, and the exploration will gradually be reduced in par with the dropping temperature.

## IV. Experiments

The SA-Q-learning algorithm has been tested on the 22×17 puzzle problem taken from [14] and compared with the results of Q-learning algorithm equipped with the $\epsilon$-greedy strategy and of the Boltzmann exploration.

### A. The Puzzle Problem

In the puzzle problem (Figure 1), the bold lines represent the walls and the squares marked 'G' are the goal states. The agent is located in one of the squares of the grid (the current state) and has the choice among eight possible actions. The four one-step actions are: North, South, West and East. As their effect the agent moves one square forward in the corresponding direction. Once encountering the wall, the agent has no possibility of moving in the given direction - the resulting state after such action is the original one. Each operation has cost 1 and the received immediate reward can be assumed to be the negative cost value, i.e., -1. In case of the four compound operations: North-to-Wall, South-to-Wall, West-to-Wall and East-to-Wall, the agent moves in the stated direction until it reaches a wall. Each of those actions has cost 3. When the agent enters the goal region (i.e., one of the goal states), it receives reward 100.

Throughout the learning process the agent tries to find the optimal path to the goal and to maximize

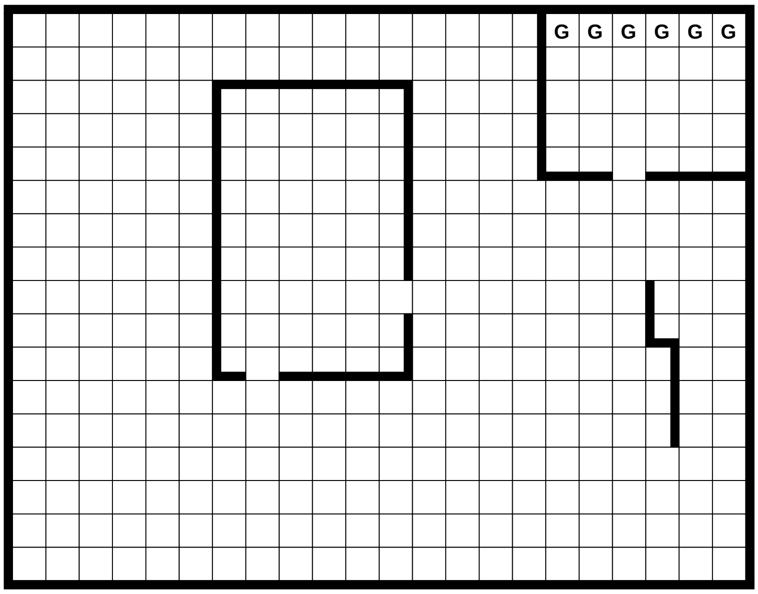


Fig. 1. The puzzle problem.

the reward it receives. Assuming no fixed initial position, all the optimal paths (i.e., the optimal policy) should be found.

*B. Results*

Performance of the SA-Q-learning algorithm has been evaluated with respect to two criteria:

1. The quality of the solution found, in terms of differences between the paths proposed by the policy learned by the algorithm, and the optimal paths for this problem, found by dynamic programming, as a function of the episode.

2. The maximum value of the value function of the learned policy, reflecting the distance to the optimal policy.

The comparison has been done for the Boltzmann exploration algorithm, the basic Q-learning algorithm (i.e., the 0-greedy strategy), the Q-learning equipped with the $\epsilon$-greedy strategy for $\epsilon > 0$, and the SA-Q-learning algorithm, as described above.

The graph in Fig. 2 presents the growth of the quality of solutions found by the four algorithms. The quality is measured by taking the ratio between the number of steps on the optimal path from the initial state to a goal state, and the number of steps on the path proposed by the policy learned so far. First of all, it can be noticed that this ratio is higher for the SA-Q algorithm than for the Boltzmann exploration. This may be attributed to the poor separability of the Q-values for the problem at hand

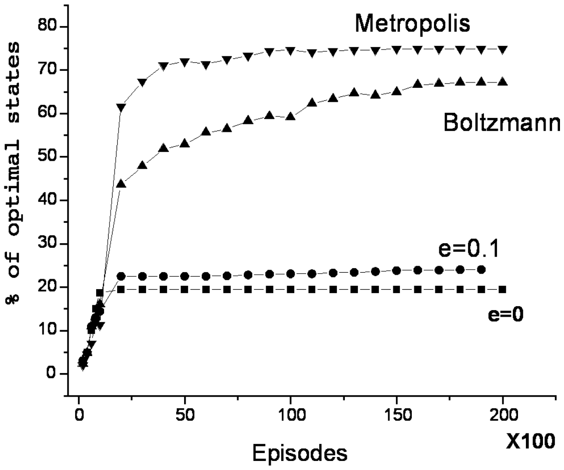


Fig. 2. The quality of the solutions found by the algorithms.

and to the fixed schedule of temperature decay. In case of SA-Q-learning the random value $\xi$ enforces additional, beneficial exploration, in particular in the beginning of the learning process. In turn, the ratio is much higher for the Boltzmann exploration, which decreases exploration with the temperature decayed over time, than for the basic Q-learning and the $\epsilon$-greedy strategy which provide much worse on-line learning performance due to the unchanged exploration probability. Moreover, while the pure reinforcement learning leads to discovery of interactions among actions and eventually to some locally optimal solution, the $\epsilon$-greedy case with larger $\epsilon$ decreases the probability of executing the (locally) optimal actions by the agent. For the studied case, $\epsilon = 0.1$ this improved the quality of the solution, but only marginally.

For the SA-Q-learning algorithm, however, the chances of converging to non-optimal policies are much smaller due to initial exploration and to its subsequent decrease according to the Metropolis criterion.

TABLE I

THE RECEIVED POLICY VALUES FOR THE DIFFERENT ALGORITHMS.

| | Number of Episodes | | | | |
|---|---|---|---|---|---|
| | 1000 | 10000 | 20000 | 30000 | 40000 |
| Boltzmann exploration | 606 | 19853 | 21742 | 22385 | 22628 |
| basic Q-learning | 609 | 22651 | 24142 | 24245 | 24253 |
| $\epsilon$-greedy ($\epsilon = 0.1$) | 609 | 22611 | 24029 | 24129 | 24611 |
| SA-Q-learning | 687 | 28086 | 28099 | 28110 | 28115 |

Table I presents the values of the $V$ function for the policies found after specific number of learning episodes. Although a more detailed analysis could be done, we have chosen to represent the result of learning by the total sum of all the $V$ values in the grid. Such an illustration is sufficient to draw at least approximate conclusions about the quality of learning of the four algorithms. The values found by SA-Q-learning algorithm exceed substantially those found by the Boltzmann exploration, the basic Q-learning and the 0.1-greedy Q-learning. Moreover, it should be noted that the values of the Boltzmann exploration algorithm are smallest, although the algorithm converges faster than the basic Q-learning and the 0.1-greedy Q-learning. This would suggest that fewer policies reach the optimum in case of the Boltzmann exploration than in the other approaches. Actually, the table I is another way of presenting the same results as shown in Figure 2, where the figure focuses on local quality of a policy (exemplified by a path from a randomly chosen start state), while the table presents a global, compound evaluation of the complete policy.

## V. Conclusions

The paper presents learning the optimal policy in Q-learning as a combinatorial optimization problem. The relation between exploration and exploitation in search for an optimal solution is presented and a version of the Q-learning algorithm based on the Metropolis criterion — the SA-Q-learning algorithm is proposed to balance between exploration and exploitation. Preliminary results show that the SA-Q-learning is superior to the standard Q-learning algorithm with respect to convergence speed. SA-Q-learning exhibits also improved performance in comparison with some random learning algorithms, namely the $\epsilon$-greedy strategy and the Boltzmann exploration.

Our future investigations will analyze the convergence speed of the SA-Q-learning algorithm in relation to the discount factor $\lambda$. Moreover, similarly to the Simulated Annealing algorithm, the influence of the dropping criterion of the temperature needs to be further analyzed.

## Acknowledgment

This paper has been completed while Maozu Guo has been visiting the Department of Computer Science, Lund University, Sweden. His visit has been supported by the Swedish Institute.

## References

[1] R.S. Sutton, D. Precup, S. Singh. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112 (1999), pp. 181–211.

[2] M.Z. Guo, B. Chen, X.L. Wang, J.R. Hong. A summary on reinforcement learning. Computer Science, 25 (3) (1998), pp. 13–15 (in Chinese).

[3] R.S. Sutton. Learning to predict by the method of temporal difference. Machine Learning, 3 (1) (1988), pp. 9–44.

[4] C.J.C.H. Watkins. Learning from delayed rewards. Ph.D. dissertation, Psychology Department, Cambridge University, 1989.

[5] C.J.C.H. Watkins, P. Dayan. Q-learning. Machine Learning, 8 (3) (1992), pp. 279–292.

[6] L. Yang, J.R. Hong, T.Y. Huang. Combining the methods of temporal differences with neural network for real-time modeling and prediction of time series. Chinese Journal of Computers, 19 (9) (1996), pp. 695–700 (in Chinese).

[7] M. Asada, E. Uchibe, K. Hosoda. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. Artificial Intelligence, 110 (1999), pp. 275–292.

[8] R.S. Sutton, A.G. Barto. Reinforcement Learning: an Introduction. MIT Press, 1998

[9] L. P. Kaelbling, M. L. Littman, A. W. Moore. Reinforcement Learning. A Survey. Journal of AI Research, 4 (1996), pp. 237–285.

[10] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller. Equation of calculations by fast computing machines. J. Chem. Physics, 21 (1953), pp. 1087–1092.

[11] S. Boettcher, A. Percus. Nature's way of optimizing. Artificial Intelligence, 119 (2000), pp. 275–286.

[12] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimisation by simulated annealing. Science, 220 (1983), pp. 671–680.

[13] T. Mitchell. Machine Learning. McGraw-Hill, 1997.

[14] T.G. Dietterich, N.S. Flann. Explanation-based learning and reinforcement learning: a unified view. Machine Learning, 28 (1997), pp. 169–210.