

Tabu Search Exploration for On-Policy Reinforcement Learning

Myriam Abramson
George Mason University
Fairfax, VA 22030
mabramso@gmu.edu

Harry Wechsler
George Mason University
Department of Computer Science
Fairfax, VA 22030
wechsler@cs.gmu.edu

Abstract—On-policy reinforcement learning provides online adaptation, a characteristic of intelligent systems and lifelong learning. Unlike dynamic programming, an exhaustive sweep of the search space is not necessary for convergence in reinforcement learning with an efficient exploration strategy. For efficient and “believable” online performance, an exploration strategy also has to avoid cycling through previous solutions and know when to stop without getting stuck in a local optimum. This paper addresses the above problem with tabu search (TS) exploration. Several strategies for reinforcement learning are introduced. Experimental results are presented in the game of Go, a deterministic, perfect-information two-player game, and Sarsa Learning Vector Quantization (SLVQ), an on-policy reinforcement learning algorithm.

I. INTRODUCTION

The convergence of reinforcement learning based on stochastics depends on visiting every state infinitely often. The Bellman optimality equation is predicated on the value function representing a good approximation of its expected return¹. This is not obviously feasible for large state space and the role of exploration is to bias the search while providing an accurate representative sample of the state space. Exploration helps with convergence not only in guiding what to explore but also in deciding what not to explore. Tabu search[6] explicitly addresses this aspect. Tabu search is a memory-based exploration method based on inhibition and restraint. Although no formal proof of its admissibility exists, it has been applied successfully to combinatorial optimization problems and combined with learning algorithms[7]. For problems where the state space itself is exponential in nature, i.e. P-space problems, the speed-up of memory-based methods over randomization as noted in [17] does not reduce the search to a polynomial case. By reducing the neighborhood of candidate solutions, TS will identify an optimal or near-optimal solution after examining only a small subset of the state space. As shown in [5], [9], action penalties can make the Q-values

¹Recall in dynamic programming, the Bellman optimality equation for V^* is

$$V^*(s) = \max_a \sum_{s'} p_{ss'}^a [R_{ss'}^a + \gamma V^*(s')]$$

Reinforcement learning replaces prior knowledge of the transition probabilities by an estimation based on sampling next-state returns.

“admissible” by non-overestimating their expected return for Q-learning. The intuition of this paper is that action inhibition can also provide a similar focusing mechanism to exploration for on-policy reinforcement learning.

The trade-off dilemma between exploration and exploitation, as illustrated by the two-armed bandit problem, has been shown[8] to have an optimal strategy in the allocation of an exponential number of trials to the observed better arm with respect to the observed worse arm. With respect to sampling in reinforcement learning, this means that the observed best action should progressively receive more trials. In this context, two types of exploration methods are distinguished:

- *Diffusion* of the target policy towards other candidate actions. As training progresses, this diffusion lessens. The metaphor here is that of looking at something from far away and then getting closer. Bonus/penalty approaches that augment the evaluation of a move entail a diffusion of the target policy.
- *Permutation* of the behavior policy toward the target policy. The metaphor here is to turn each stone systematically before moving forward. Tabu search and simulated annealing belong to this type.

Go is a perfect information, deterministic, 2-player game that has replaced chess as the “drosophila” of AI. It has been shown that Go is P-space hard[11] and therefore “solving” the game with an algorithm computing an exact evaluation function is probably not practical. Go is hard for machine learning because there are no clear identifiable features and the input-output function is not smooth in all cases. Exploration is a way to combine heuristic search in a reinforcement learning framework.

This paper is organized as follows. Section II highlights the differences between on and off policies in reinforcement learning and motivates the need for efficient exploration. Section III present how tabu search principles can be applied to reinforcement learning. Section IV introduces SLVQ and Section V shows experimental results.

II. REINFORCEMENT LEARNING POLICIES

In reinforcement learning, a learning agent interacts with its environment by taking actions and accepting input from

the environment. Input from the environment constitutes the state of the environment followed by an immediate reward. State information passed to the agent summarizes all currently relevant information about the environment. In contrast to a purely reactive agent, a learning agent is endowed with an internal state that summarizes past history of its interactions with the environment. The environmental state and the internal state of the agent together are the state of the system upon which the learning agent bases its actions. An internal state enables an agent to generalize from previous experience which is missing from purely reactive architecture. The reward passed to the learning agent is a scalar reinforcement that serves to evaluate current and past actions. While interacting with the environment, the agent follows a policy to determine what actions to take. A policy is a function, denoted as π , that maps the system state to an action to be taken by the agent. Through interaction with the environment, the agent learns either a value function $V^\pi(s)$, which represent the “desirability” of a state s given a fixed policy π , or an action-value function, denoted as $Q(s, a)$, which maps state s and action a to a “long-term” reward $E[\sum_{t=0}^{\infty} \gamma^t r^{t+1}]$ where $0 < \gamma \leq 1$ and r^t is the reward at time t . This form of expected long-term reward is called “discounted future” reward over an infinite horizon and has a finite value.

There are two basic ways of using experience in reinforcement learning: off-policy and on-policy[16]. They differ only by the update rule used to arrive at an optimal policy. Figure 1 illustrates the basic differences between on and off policies regarding the action selection and update step. In an on-policy, the policy being updated (target policy) affects the selection of the next move. In an off-policy, the move evaluation of the next “best” move affects the update of the current move but the move selection does not depend on the policy being updated and can come from a completely different policy (behavior policy). Both policies reflect the bootstrapping strategy of dynamic programming to update the prediction for s_t from the next prediction s_{t+1} . The off-policy, embodied in the Q-learning algorithm[18], uses the estimate of the optimal policy for update of the existing policy and consequently separates exploration from control. The on-policy, embodied in the Sarsa algorithm(1) [12], [16], uses the current estimate of an existing non-optimal policy for refinement towards a *better* existing policy and combines exploration and control. The only guarantee to arrive at an optimal policy with Sarsa is possible only if the control policy progressively inches itself toward the optimal policy as the exploration tapers off during training[14]. On-policy RL algorithms are dependent on the exploration for the accuracy of the action values. In both policies, convergence has been proved in the discrete, tabular case if each action is selected infinitely often[19], [3]. Convergence has also been proven for TD(λ) in the linear representation case[4].

III. TABU SEARCH

Meta-level search addresses the question of how to search in order to learn. Genetic algorithms, simulated annealing and

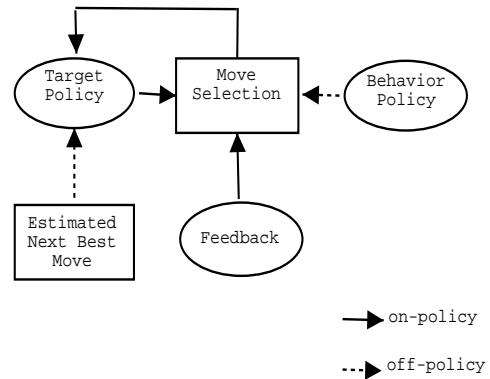


Fig. 1. On and off policies

Algorithm 1 Sarsa

Initialize $Q(s, a)$, the learning rate α , and the discount factor γ .

Observe the current state s

While stopping condition is false

- 1) Select an action a to execute according to π
 - 2) Receive immediate reward r
 - 3) Observe the new state s'
 - 4) Update $Q(s, a)$
 $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a))$
 - 5) Update π towards \hat{Q}
 - 6) Set s to s'
-

tabu search are considered metaheuristics, or general search strategies, built on heuristics for the application domain. Tabu search is a meta-heuristic that, unlike simulated annealing and genetic algorithms, uses knowledge of the history of the search instead of blind randomization to navigate through the search space and escape local optimum solutions. It uses knowledge by remembering characteristics of state-action pairs. It generalizes memory-based methods such as the counter-based method[17]. We must be able to recognize a solution (aspiration criteria) and we must be able to focus our attention on promising regions of the search space while avoiding cycling through previous solutions. Technically, a tabu search approach involves picking the best candidate that is not marked *tabu* from a candidate list, i.e. that is not on the tabu list, and override the tabu restriction if the candidate satisfies the *aspiration criteria* (see Algorithm 2). Theoretically, a tabu search approach involves recognizing what is right, what is new, and what is similar in the candidate solutions. In contrast, randomization is considered a weak diversification method. The tabu search approach has a principled way of choosing exploitation over exploration.

The “least” tabu move is selected if no moves satisfy the aspiration criteria and all moves are tabu. A tabu search entails the definition of the aspiration criteria and the specification of the tabu list structure which are discussed below.

Algorithm 2 Tabu search

Choose an initial feasible solution i

- 1) Generate candidate solutions in the neighborhood of i
 - 2) Filter out solutions that do not meet the aspiration criteria or that are marked *tabu*
 - 3) Select the best candidate solution j .
 - 4) If candidate solution j improves on solution i , set $i = j$
 - 5) Update tabu status of candidate moves and aspiration criteria
 - 6) If stopping criteria is met, stop, else go to 2
-

A. Aspiration Criteria

What could be an aspiration criterion in a reinforcement learning approach? In other words, how do we recognize such a compelling move in an online learning approach versus a *best* move according to current estimate? The consistency of a move with a positive outcome can be measured after a certain number of trials if a non-overlapping confidence interval with other candidate moves can be found:

$$\left[\bar{X} - t_{n-1} \frac{s}{\sqrt{n}}, \bar{X} + t_{n-1} \frac{s}{\sqrt{n}} \right]$$

where \bar{X} is the moving average of the action value Q , s the sample standard deviation of Q and t_{n-1} the *critical value* of the t distribution with $n - 1$ degrees of freedom for a specified confidence level. For a 95% confidence level and $n = 30$, $t_{n-1} = 2.0452$.

An aspiration criterion defined by the predictability of a positive outcome models the preferential treatment of attentive behavior to states with high outcome reliability[15]. Selective attention, the capability to discriminate between input states, has been shown to increase with the consistency of the outcome and decrease with variability. An aspiration criterion is necessary to focus the search for convergence.

B. Tabu List Strategies

Restrictions on the candidate moves structure the search space to explore variations in a local neighborhood. This is equivalent to systematically turning each stone in a region of the search space before looking elsewhere. When restrictions are temporal restrictions, a tabu list stands for a short-term memory. Otherwise, a tabu list is similar to a nearest-neighbor type of approach where the neighborhood can be defined in different ways. The tabu list structure is the source of diversification as well as intensification modulated by the tabu tenure parameter. Some different approaches to structuring a tabu list for exploration in a reinforcement learning algorithm are introduced below.

1) *Simple Tabu Search(STS)*: Similarly to the temperature in simulated annealing, a tabu tenure is defined over the moves on the tabu list. The policy becomes greedier as the tabu tenure declines to 0. Candidate moves are ordered according

to their evaluation and the best move that is not tabu, i.e. not on the tabu list, is picked and thereafter becomes tabu for the length of time specified by the tabu tenure parameter by setting its tabu status. All other moves on the tabu list see their tabu status decline. To avoid cycling through previous sequences of moves, the tabu tenure of each state-action pair is set with some random variation of each other. Algorithm 3 illustrates STS applied to reinforcement learning. As opposed to the bonus/penalty approach, the rate of exploration can be controlled to match the training time as in simulated annealing by adjusting the length of the tabu tenure as an initial global parameter. The length of the tabu tenure determines the degree of intensification or diversification. The tabu tenure parameter decays locally in proportion of the number of times an action has been used:

$$TabuStatus = TabuTenure * 0.99^{hits}$$

Algorithm 3 STS move selection

s is the current state and Q is the set of all state-action pairs
PICKMOVE(s , Q)

- 1) matches \leftarrow all pairs matching s
 - 2) bestMatch $\leftarrow \arg \max_{Q(s,a)}(matches)$
 - 3) tabuMatches \leftarrow pairs with tabu moves (tabu status > 0)
 - 4) permissibleMatches \leftarrow non-tabu move pairs
 - 5) if bestMatch satisfies aspiration criteria
pair \leftarrow bestMatch
jump to 7
 - 6) pair $\leftarrow \arg \max_{Q(s,a)}(permissibleMatches)$
 - 7) pair.tabuStatus \leftarrow tabuTenure
 - 8) for all pairs \in tabuMatches
pair.tabuStatus $\leftarrow \max(pair.tabuStatus - 1, 0)$
 - 9) return pair.move
-

2) *Elite List Strategy*: Aspiration status and tabu status are often mirror images of each other. Exclusively selecting one type of move renders the others “tabu”. This strategy lends itself well to a reinforcement learning approach. The best moves are selected and updates are stored in a buffer until the aspiration tenure of the moves expires. Once updated, their aspiration status might change and other moves get a chance to be selected. This buffering of the backups have the added benefit to stem out noise from subsequent moves.

3) *Concept-based Tabu search*: The tabu status can be computed not only on search attributes but also on the move attributes themselves. This is where the opportunity to integrate domain knowledge arises and the capability to select truly “different” moves based on content arises. This strategy is illustrated in the game of Go. The operational characteristics of moves in Go can express different purposes or strategies:

Claim	Increase the player’s moyo ²
Defend	Increase the player’s sum of liberties

²Sphere of influence or potential territory.

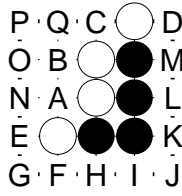


Fig. 2. Candidate moves in life-and-death pattern

a	{F,3} {B,3} {C,3} {D,3} {E,3}, {A,3}
b	{O,3} {N,3} {P,3} {Q,3} {F,2} {B,2} {C,2} {D,2} {E,2}

TABLE I

TABU LISTS FOR THE FIRST 2 MOVES IN FIGURE 2

Connect Join two groups
 Invade Decrease the opponent's moyo
 Attack Decrease the opponent's sum of liberties

More abstract features, such as safe or expert move, could also be determined offline with a supervised learning approach[2]. Here, we do not try to evaluate the moves themselves but use their categorization to guide the exploration of the search space. The categorization of the candidate moves in Figure 2 is as follows:

Attack A,B,C,D,E,F
 Invade N,O,P,G,Q

M, L, K, I, J and H do not accomplish any purpose and are therefore removed from the candidate list if included thereby pruning the neighborhood of candidate solutions further. If move F is selected, its tabu status is set to the tabu tenure parameter. The tabu status of moves A, B, C, and D are set as well since they belong to the same move class. Table Ia shows the tabu list after the first move with a global tabu tenure parameter of 3. Let's assume White plays A afterwards. Black will then play an invading move if any are on the candidate list since all attacking moves are tabu. Let's assume Black plays O . Table Ib shows the tabu list after this second Black move. Moves with a tabu status of 0 are removed from the tabu list. The tabu status of move F won't be reconsidered until it becomes a candidate move again.

IV. SLVQ

Tabu search exploration methods are illustrated in conjunction with SLVQ[1], an on-policy reinforcement learning algorithm combining Sarsa[12], [16] and LVQ[10] as a function approximator (see Figure 3). The distributed LVQ representation of the policy function generates a piecewise constant tessellation of the state space that significantly reduces the state space requirements and has a close correspondence to a tabular representation of state-action pairs. SLVQ maintains a set of *codebook* tuples $\{m, a, Q, \alpha, T\}$ where m is the weight vector, a the action associated with m , Q the action value, α the local

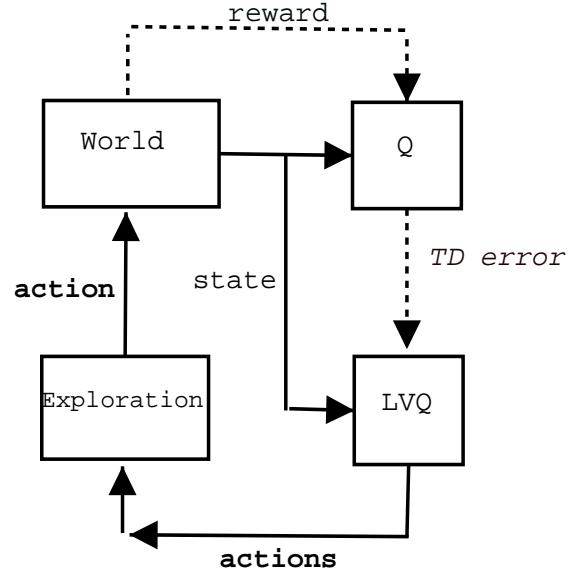


Fig. 3. SLVQ architecture

learning rate, and T , the tabu status of a . Its update rule with s_t as the current state and γ as the discount parameter is as follows:

$$m_t = \arg \max_m \text{similarity}(s_t, m)$$

$$\Delta Q(m_t, a) = \alpha_{m_t} [r_t + \gamma Q(m', a') - Q(m_t, a)]$$

$$m(t+1) = m(t) + \Delta Q(m_t, a) [s_t - m_t]$$

V. EXPERIMENTAL RESULTS

SLVQ initializes its weight vectors from random games and plays black. The similarity function used in the game of Go is based on Tversky's contrast model adapted to continuous features[13]. Let a pattern P be represented by the set of influence values p_{ij} (after normalization) at each of its points. The similarity S between patterns P and R can be computed as a function of their commonality and their reflexive differences where α and β represents the saliency of P and R respectively:

$$\text{Closeness}(P, R) = \sum_i \sum_j \min \{p_{ij}, r_{ij}\}$$

$$\text{Contrast}(P, R) = \sum_i \sum_j \max \{p_{ij} - r_{ij}, 0\}$$

$$\text{Contrast} = \alpha \text{Contrast}(P, R) - \beta \text{Contrast}(R, P)$$

$$\text{Similarity}(P, R) = \text{Closeness}(P, R) - \text{Contrast}$$

-0.72	-0.73	-0.49	-1.00	0.13
-0.7	-0.74	-1.00	1.00	0.4
-0.73	-0.6	-1.00	1.00	0.87
-0.62	-1.00	1.00	1.00	0.93
-0.59	-0.17	0.34	0.84	0.92

Fig. 4. Influence value representation of the pattern in Figure 2

TS methods are compared with the softmax exploration method[16] using SLVQ against 2-ply Minimax using the sum of influence values (Figure 4) as the evaluation of the board. The influence value function[20] propagates the value of a stone, +1 for black and -1 for white, to nearby empty intersection points. The softmax exploration method adds a small random number decaying with time to the Q-value of a move. Results shown are averaged over 3 runs. Data points were collected every 10 games. Smoothing of the graphs (10%) ensures an easier readability of the performance trend in this complex game where one move difference can drastically alter the outcome of the game. Winning corresponds to a positive score (territory) difference over the opponent. Please note that Go is not a zero-sum game and that winning rarely involves wiping out the opponent’s territory.

Figure 5 shows offline results using the greedy policy for the different exploration methods for the pattern in Figure 2. In this life-and-death pattern, SLVQ, augmented by different TS strategies, plays Black against minimax. Even that Black starts at a disadvantage, it wins against its minimax opponent. The following parameters were used: α the learning rate remains constant at 0.1, the number of codebook tuples was set to 300, the buffering for the Elite strategy was set to 5 updates, the initial tabu tenure parameter was set to 5, the eligibility trace λ was set to 0.2 and the task was undiscounted. The exploration rate for the softmax method was set to 0.07 and decayed with time as a function of the number of times a codebook tuple was used. There is a statistical difference in learning performance between the Elite and STS methods and the softmax method using the Wilcoxon’s matched pair signed rank test. All the TS methods converge toward the same result until the Bellmann error is minimized and win over Minimax playing white by a small margin. The softmax method does not win for this set of parameters and number of training games. The Elite strategy focuses the fastest in this problem.

Increasing the dimensions of the board to 7x7 makes the game noticeably more difficult. Data points were collected every 20 games. The number of codebook tuples was increased to 1000. The learning rate α was set initially to 0.3 and decayed to 0.1. The eligibility trace λ was set to 0.5. Figure 6 shows offline results using the greedy policy in 7x7 games against Minimax for the different TS exploration methods and the softmax method. Although all the exploration methods win against Minimax for this set of parameters, the TS methods

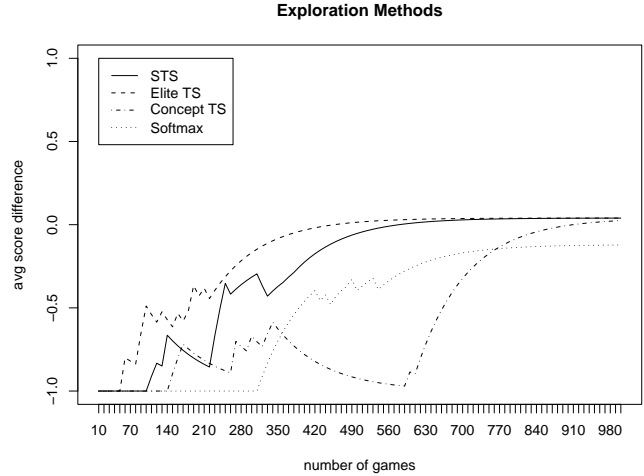


Fig. 5. Exploration methods for the pattern in Figure 2 and playing against Minimax

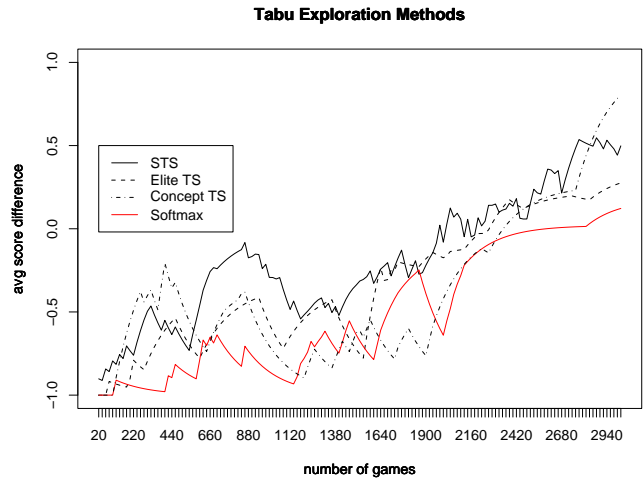


Fig. 6. Exploration methods in 7x7 games vs. Minimax

dominate in terms of learning power.

VI. CONCLUSIONS

An intelligent exploration of the search space as found in tabu search (TS) gives a mechanism for escaping local minimum states, a problem of gradient search algorithms, while avoiding cycling through previous sequences of solutions. As a directed search method based on content rather than search statistics, TS provides an opportunity to integrate domain knowledge and concept learning in reinforcement learning algorithms. In addition, TS serves also as a training method by suggesting non-random variations around the best candidate move. By focusing on promising regions of the search space quickly, TS

methods speed up the convergence of on-policy RL algorithms. Future work includes a principled determination of the tabu tenure parameter. A proof of the admissibility of TS as a metaheuristic for on-policy reinforcement learning would be very useful.

REFERENCES

- [1] M. Abramson and H. Wechsler. Competitive reinforcement learning for combinatorial problems. In *International Joint Conference on Neural Networks*, 2001.
- [2] F. A. Dahl. Honte, a go-playing program using neural nets. In *Workshop on Machine learning in Game Playing*, 1999.
- [3] P. Dayan. The convergence of $td(\lambda)$ for general λ . *Machine Learning*, (8):341–362, 1992.
- [4] P. Dayan and T. J. Sejnowski. $Td(\lambda)$ converges with probability 1. *Machine Learning*, (14):295–301, 1994.
- [5] G. DeJong. Hidden strengths and limitations: an empirical investigation of reinforcement learning. In *International Conference on Machine Learning*. Morgan Kaufmann, 2000.
- [6] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [7] A. Hertz and D. de Werra. The tabu search metaheuristic: how we use it. *Annals of Mathematics and Artificial Intelligence*, 1:111–121, 1990.
- [8] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 2nd edition, 1992.
- [9] S. Koenig and R. G. Simmons. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithm. *Machine Learning*, 22:227–250, 1996.
- [10] T. Kohonen. *Self-Organizing Maps*. Springer, 2nd edition, 1997.
- [11] D. Lichtenstein and M. Sipser. Go is polynomial-space hard. *Journal of the Association for Computing Machinery*, 27(2):393–401, 1980.
- [12] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, Cambridge University Engineering Dept., 1994.
- [13] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9), 1999.
- [14] S. Singh, T. Jaakkola, M. L. Littman, and Csaba Szepesvari. Convergence results for single-step on-policy reinforcement learning algorithms. *Machine Learning*, 38(3):287–308, 2000.
- [15] J. E. R. Staddon and R. H. Ettinger. *An Introduction to the Principles of Adaptive Behavior*. Harcourt Brace Jovanovich, 1989.
- [16] R. S. Sutton and A. Barto. *Reinforcement Learning: an Introduction*. MIT Press, Cambridge, MA, 1998.
- [17] S. B. Thrun. Efficient exploration in reinforcement learning. Technical Report TR CMU-CS-92-102, Carnegie Mellon University, 1992.
- [18] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.
- [19] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, (8):279–292, 1992.
- [20] L. Zobrist. *Feature Extraction and Representation for Pattern Recognition and the Game of Go*. PhD thesis, University of Wisconsin, 1970.