Heuristic Search Based Exploration in Reinforcement Learning

Ngo Anh Vien, Nguyen Hoang Viet, SeungGwan Lee, and TaeChoong Chung*

Artificial Intelligence Lab, Department of Computer Engineering,
School of Electronics and Information, Kyunghee University
1-Seocheon, Giheung, Yongin, Gyeonggi, 446-701, South Korea
{vienna,vietict,leesg,tcchung}@khu.ac.kr

Abstract. In this paper, we consider reinforcement learning in systems with unknown environment where the agent must trade off efficiently between: *exploration*(long-term optimization) and *exploitation* (short-term optimization). $\varepsilon - greedy$ 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. Many works already proved that random exploration drives the agent towards poorly modeled states. Therefore, this study evaluates the role of heuristic based exploration in reinforcement learning. We proposed three methods: neighborhood search based exploration, simulated annealing based exploration, and tabu search based exploration. All techniques follow the same rule "Explore the most unvisited state". In the simulation, these techniques are evaluated and compared on a discrete reinforcement learning task (robot navigation).

1 Introduction

In reinforcement learning [1], [2] it is necessary to introduce a process of trial and error designed to maximize rewards obtained from environment. This trial and error process is called an exploration. Exploration plays a fundamental role in any active learning system. Whenever a learning system learns to control an unknown environment, two opposing objectives have to be combined. On the one hand, in order to identify a suboptimal controller the environment must be sufficiently explored. For example a robot facing an unknown environment has to spend time moving around and acquire knowledge of its environment. On the other hand experiences made during learning must also be considered for action selection in order to minimize the costs of learning (e.g. in terms of negative reward). E.g. although a robot has to explore its environment, it should avoid collisions with obstacles, once it received some negative reward for collisions. Thus for efficient learning, actions should be generated such that the environment is explored and pain is avoided.

Because there is a trade-off between exploration and exploitation (avoiding bad rewards) [7], [8], balancing of them is very important. This is known as

* Corresponding author.

F. Sandoval et al. (Eds.): IWANN 2007, LNCS 4507, pp. 110–118, 2007.

the exploration-exploitation dilemma. The schema of the exploration is called a policy. There are many kinds of policies such as $\varepsilon - greedy$, Boltzmann distributions, softmax, weighted roulette and so on. In these existing policies, exploring is decided by using stochastic random numbers as its generator, according to the reference value and the provided criterions). They belong to the class of *undirected* exploration methods.

The most uninformed *undirected* exploration technique is the random walk [5], [6], which completely ignores costs and negative rewards from the environment. The term *undirected* is due to this observation: exploration is ensured only by randomness. Exploration by modified probability distributions is typically found in reinforcement learning literature. The probability distribution for action selection is drawn by the utility estimate of each action.

A different class of exploration methods is methods that use *directed* exploration. This class, introduced in [4] under the name of *active* exploration. *Directed* exploration techniques memorize exploration-specific knowledge which is used for guiding the exploration search. The exploration is thus not completely random, as in *undirected* exploration. Examples of directed exploration methods are *frequency-based*, *recency-based* and *error-based* exploration [3].

In order to efficiently utilize exploration for learning process, in this paper, we investigate the idea that exploration is a way to combine heuristic search in a reinforcement learning framework. Briefly, we may perform exploitation-exploration of reinforcement learning using three of most basic heuristic search algorithm: neighborhood search, simulated annealing and tabu search. These heuristic searches have been applied successfully to combinatorial optimization problems and combined with learning algorithms [9], [10]. It is for the reason that set of environment states is so large, and the agent could not explore each states in the environment with infinite visits. Therefore, we decided to focus on the use of heuristics based exploration. Unlike the $\varepsilon - greedy$ exploration techniques, heuristic based exploration uses heuristic and memorized knowledge about previous learning to direct the exploration. In [11], the authors investigated the idea that applied online search techniques to reinforcement learning. Search techniques were utilized to find a better trajectory rather than executing the greedy policy with respect to the approximated value function. An adaptive simulated annealing based reinforcement learning method [12] utilized the power of global optimization methods such as simulated annealing to cope with poor convergence properties for difficult problems. This method considered a batch formulation for the reinforcement learning problem, unlike almost the online formulation always used. In this paper, we proposed three heuristic based exploration techniques based on a) neighborhood search, b) simulated annealing search, and c) tabu search. All follow the same rule "Explore the most unvisited state". They are similar to $\varepsilon - greedy$ in that they still behave greedily with probability $1 - \varepsilon$, but every once in a while, with small probability ε, they not choose random action otherwise they choose the action by using heuristic.

The rest of this paper is organized as follow. The next section is reinforcement learning algorithm. In Section 3, heuristic based exploration methods are presented. Sections 4 reports simulations and experimental results respectively. Concluding remarks follow in Section 5.

2 Reinforcement Learning

Reinforcement learning [2] is a sub-area of machine learning concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states.

One of the most important breakthroughs in reinforcement learning was the development of an off-policy Temporal Difference control algorithm known as Q-learning. Its simplest form, one-step Q-learning, is defined by

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

In this case, the learned action-value function, Q, directly approximates Q^*, the optimal action-value function, independent of the policy being followed. This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs. The policy still has an effect in that it determines which state-action pairs are visited and updated. However, all that is required for correct convergence is that all pairs continue to be updated. Under this assumption and a variant of the usual stochastic approximation conditions on the sequence of step-size parameters, Q_t has been shown to converge with probability 1 to Q^*. The Q-learning algorithm is shown in procedural form in Fig. 1

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from $s$using policy derived from Q (e.g., $\varepsilon - greedy$)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  until s is terminal

Fig. 1. Q-learning: An off-policy TD control algorithm

$\varepsilon - greedy$ is method using near-greedy action selection rule. It behaves greedily most of the time to select the best action suggested by the Q function with a probability of ε, and it selects a random action with probability of $(1 - \varepsilon)$.

3 Heuristic Search Based Exploration

Unlike the $\varepsilon - greedy$ exploration techniques described so far, heuristic exploration uses heuristic and memorized knowledge about previous learning to direct the exploration. In this section, we will describe heuristic exploration techniques one by one based on a) neighborhood search, b) simulated annealing search, and c) tabu search. All follow the same rule "Explore the most unvisited state". Implementing this rule which is different from each heuristic techniques will be in turn illustrated.

3.1 Neighborhood Search Based Exploration

For each state, we use one variable to store the chosen action $action_{last}(state)$ of the previous visit. And $N(s, action_{last}(s))$ is the neighborhood action set in the next visit to state s. Using neighborhood search based exploration, we choose the action for the current state to be based on $N(s, action_{last}(s))$. And randomly choosing one action $a \in N(s, action_{last}(s))$ for the current state. Therefore, when the robot explores, the random action is chosen only in the set of neighborhood actions of the last visiting's action. This method is different from $\varepsilon - greedy$ at which exploration's action is chosen in a narrowed set and closer action to the previous action in comparison to $\varepsilon - greedy$ exploration's action set. Neighborhood search based exploration is illustrated as following:

1.(Initialization) $Q(s, a) = 0$;
1.1 Robot starts at state s_{now} = starting state.
1.2 (Action choosing)
+) With probability q_0, action is chosen greedily : $a = argmaxQ(s_{now}, a)$.
+) else with probability $1 - q_0$: Randomly select an action $a \in N(s_{now}, action_{last}(s_{now}))$.
2. (Update and termination)
2.1 Take action a, observe next state s_{next} and reward.
2.2 Update Q-value (s_{now}, a)
2.3 $action_{last}(s_{now}) \leftarrow a$; $s_{now} \leftarrow s_{next}$
2.4 If the next state is destination state and termination criteria not apply, then return to step 1.1, else return to 1.2

3.2 Simulated Annealing Based Exploration

Simulated annealing 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 (inferior action which has lower Q-value in terms of negative reward. Lower Q-value means worse future predicted result) under the control of a randomized scheme. Specifically, if a choice from one action $action_{last}(s)$ to another

neighboring but inferior action a results in a change in value ΔQ, the action a is still accepted if:

$$\exp(-\Delta Q/T) > R \quad \text{where } \Delta \mathrm{Q} = \mathrm{Q(s,a)} - \mathrm{Q(s, action_{last}(s))} \tag{2}$$

and T is a control parameter, and $R \in [0, 1]$ is a uniform random number. 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. There is the concept of simulated annealing approach. It is called the thermodynamic process of annealing in physics. Consequently, the simulated annealing based exploration is summarized as following:

1.(Initialization) $Q(s, a) = 0$;
1.1 Robot starts at state s_{now} = starting state.
2 (Action choosing)
2.0 Cooling schedule: $T \leftarrow \alpha T$
2.1 With probability q_0, action is chosen greedily : $a = argmaxQ(s_{now}, a)$.
2.2 else with probability $1 - q_0$: Randomly select an action $a \in N(s_{now}, action_{last}(s_{now}))$:
a) if $Q(s_{now}, a) > Q(s_{now}, action_{last}(s_{now}))$ then accept action a ;
b)otherwise: If action a satisfies Eq. (2) then accept a, else return to 2.2
3. (Update and termination)
3.1 Take action a, observe next state s_{next} and reward.
3.2 Update Q-value (s_{now}, a)
3.3 $action_{last}(s_{now}) \leftarrow a; s_{now} \leftarrow s_{next}$
3.4 If the next state is destination state and termination criteria not apply, then return to step 1.1, else return to 2

We choose the cooling schedule as following: after each episode the temperature parameter is reduced by a geometric schedule: $T \leftarrow \alpha T$ (α in the range 0.9 to 0.99).

3.3 Tabu Search Based Exploration

Tabu Search, also like simulated annealing, is based on neighborhood search, but in a deterministic way which tries to model human memory processes. Memory is implemented by the implicit recording of previously-seen state-action pair. These centre on the creation of a *tabu list* of moves which have been made in the recent past of the exploration, and which are *tabu* or forbidden for a certain number of iteration. This help agent to avoid re-revisiting, and serves also to promote a diversified search of the solutions.

One objective in Tabu Search is to encourage exploration of parts of solution space that have not been visited previously. This can be achieved in practice by prohibiting the reversal of previous state-action pairs. So we prohibit the reversal of the most recent state-action pairs only. Recency may be construed as a fixed parameter (the tabu tenure of a move), or it may be allowed to vary dynamically

during the search. We use the basic concept of tabu search that was derived from neighborhood search description. In the tabu search based exploration, a history record H is kept of the state-action pairs previously encountered during the search, so that the neighborhood $N(s_{now}, a)$ is modified to $N(H, s_{now}, a)$ as follows:

1.(Initialization) $Q(s, a) = 0$;
1.1 Robot starts at state s_{now} = starting state.
1.2 (Action choosing)
+) With probability q_0, action is chosen greedily : $a = argmaxQ(s_{now}, a)$.
+) else with probability $1 - q_0$: Randomly select an action $a \in N(H, s_{now}, action_{last}(s_{now}))$.
2. (Update and termination)
2.1 Take action a, observe next state s_{next} and reward.
2.2 Update Q-value (s_{now}, a)
2.3 $action_{last}(s_{now}) \leftarrow a$; $pair(s_{now}, a) \leftarrow tabu$; $H = H + \{pair(s_{now}, a)\}$
2.4 for all $pairs \in H$ $pair.tabuLength \leftarrow pair.tabuLength - 1$; remove pairs from H which have tabuLength == 0
2.5 $s_{now} \leftarrow s_{next}$. If the next state is destination state and termination criteria not apply, then return to step 1.1, else return to 1.2

The pair state-action is marked *tabu* when the agent at that state chooses that action. And the length of *tabu* time is decreased at each move of the agent.

4 Simulation and Experimental Results

We test all exploration methods on the two-dimensional robot navigation task as depicted in Fig. 2(a). The task is to navigate the robot (top-left) to its goal position (right margin right) with as few steps as possible. This navigation task is called random gridworld task. In this evaluation, we used 30×30 feasible gridworld in which the goal state can be reached from any free state. The blocked states are black. Therefore, the state space consists of roughly 900 states which are represented by its x-y-coordinates in the grid. Each time the robot has 4 valid actions (south, north, east, and west), each of which corresponds to one neighbor in the grid. There are three kinds of rewards. If the robot collides against the block or wall, it moves back to the previous state but receives the negative reward -1. Positive reward 1 is only received when entering the goal position. And the zero reward is received when moving to a new free state which is not the goal position. At the beginning, no a priori information is provided to guide the search of the goal, and the Q-values are initialized randomly.

We compared three proposed strategies with $\varepsilon - greedy$ strategy. In the simulation, we used following parameters setting: $q_0 = 0.8$ is also used for all techniques, learning rate $\alpha = 0.1$, discount rate $\gamma = 0.9$. With SA based method: cooling rate $\alpha = 0.9$, initial temperature $T = 800$. Table 1 shows the comparison of the complexity of the first trial. Complexity of the first trial is an important characteristic of the exploration techniques investigated is the number of

actions required for the first trial. In this first run there is no knowledge about the environment available, thus the Q-values are initialized randomly. The table illustrated the average number of steps required for the first run averaged over 20 experiments each. The third column of Table 1 shows the average steps over all trials so far (over 3000×20 trials, each experiments have 3000 trials). And the forth column shows the shortest path (number of steps from original position to destination).

Table 1. Complexity of the first trial

Exploration technique	Average steps per the first trial	Average steps per trial	Shortest path (steps)
$\varepsilon - greedy$	32.198	190	70
Neighborhood based exploration	23.804	186	68
Simulated annealing based	18.728	179	63
Tabu search based	19.274	184	66

These results describe that heuristic based explorations seem to be more efficient in terms of expected search steps for finding the goal in the beginning. And $\varepsilon - greedy$ exploration takes in this particular task much more steps.

Figure 2 shows convergence comparison of all techniques (for clear comparison at the convergent stage, only the trials after 400 were displayed). Neighborhood

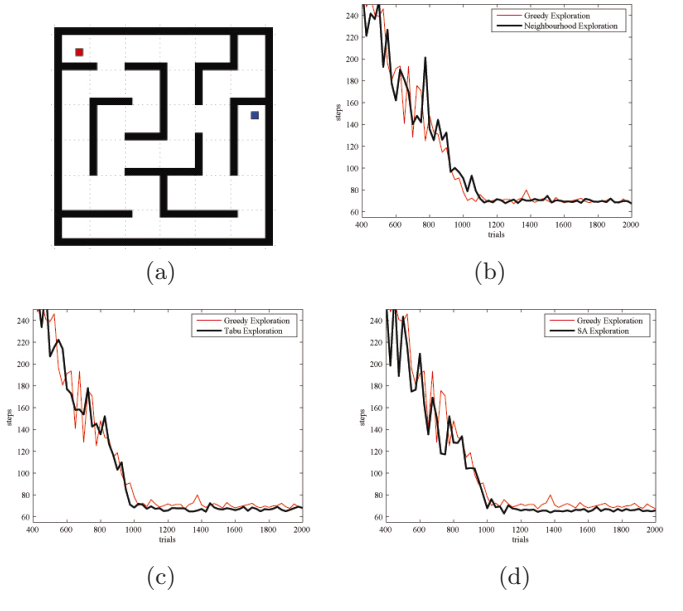

Fig. 2. a) The task is to navigate the robot (top-left rectangle) to its destination (right rectangle) on the shortest possible path. b), c), d) Convergence comparisons.

based exploration seems smoother than $\varepsilon - greedy$ when the algorithm converges. This smoothness can be explained that: because neighborhood based exploration only chooses the neighboring action to the previous action, so that the exploring action can not be so far from the convergent action. Therefore the trials in the convergent stage is looked like smooth. Tabu search based exploration seems not only improve the smoothness of convergent stage but the shortest path convergence also. Simulated annealing based exploration seems to be the best. It found the shortest path in comparison with the other.

5 Conclusion

This paper discussed about the fundamental heuristic based exploration in reinforcement learning. We proposed three exploration models based on heuristic search: neighborhood search based, simulated annealing based, and tabu search based. These techniques utilize heuristic search to drive the exploration part in the learning process. They are similar to $\varepsilon - greedy$ in that they still behave greedily with probability $1-\varepsilon$, but every once in a while, with small probability ε, they not choose random action otherwise they choose the action by using heuristic. The experimental results show that in robot navigation problem, heuristic based exploration methods worked better than $\varepsilon - greedy$ method on aspects: complexity of the first trial, and finding the shortest path.

References

1. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
2. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
3. Wiering, M. A.: Explorations in efficient reinforcement learning. Ph.D. dissertation, University of Amsterdam IDSIA (February 1999)
4. Thrun, S., Moller, K.: Active exploration in dynamic environments. In: Moody, J.E., Hanson, S.J., Lippmann, R. (eds.) Advances in Neural Information Processing Systems 4, pp. 531–538. Morgan Kaufmann, Washington (1992)
5. Nguyen, D., Widrow, B.: The truck backer upper: An example of self-learning in neural networks. In: Proceedings of the First International Joint Conference on Neural Networks Washington DC San Diego. IEEE TAB Neural Network Committee
6. Thrun, S. B., Moller, K. ,Linden, A.: Planning with an adaptive world model. Advances in Neural Information Processing Systems, San Mateo, Morgan Kaufmann
7. Holland, J.H.: Adaptation in Natural and Artificial System, 2nd edn. MIT Press, Cambridge (1992)
8. Macready, W., Wolpert, D.H.: Bandit problems and the Exploration/Exploitation Tradeoff. IEEE Transactions on Evolutionary Computation 2(1), 2–22 (1998)
9. Reeves, C.R.: Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publication, Oxford (1993)
10. Downsland, K.: Simulated annealing. In: Downsland, K. (ed.) Modern Heuristic Techniques for Combinatorial Problems, Chapter II, Blackwell Scientific Publication, Oxford (1993)

11. Davies, S., Andrew, Ng., Moore, A.: Applying Online Search Techniques to Continuous-State Reinforcement Learning. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-1998)
12. Atiya, A.F., Parlos, A.G., Ingber, L.: A reinforcement learning method based on adaptive simulated annealing. Circuits and Systems, 2003. MWSCAS '03. In: Proceedings of the 46th IEEE International Midwest Symposium on, vol. 1, pp. 121–124 (December 2003)
13. Abramsan, M., Wechsler, H.: Competitive reinforcement learning for combinatorial problems. In: International Joint Conference on Neural Network (2001)