Chaotic Exploration Generator for Evolutionary Reinforcement Learning Agents in Nondeterministic Environments

Akram Beigi, Nasser Mozayani, and Hamid Parvin

School of Computer Engineering, Iran University of Science and Technology (IUST), Tehran, Iran
{Beigi,Mozayani,Parvin}@iust.ac.ir

Abstract. In reinforcement learning exploration phase, it is necessary to introduce a process of trial and error to discover better rewards obtained from environment. To this end, one usually uses the uniform pseudorandom number generator in exploration phase. However, it is known that chaotic source also provides a random-like sequence similar to stochastic source. In this paper we have employed the chaotic generator in the exploration phase of reinforcement learning in a nondeterministic maze problem. We obtained promising results in the so called maze problem.

Keywords: Reinforcement Learning, Evolutionary Q-Learning, Chaotic Exploration.

1 Introduction

In reinforcement learning, agents learn their behaviors by interacting with an environment [1]. An agent senses and acts in its environment in order to learn to choose optimal actions for achieving its goal. It has to discover by trial and error search how to act in a given environment. For each action the agent receives feedback (also referred to as a reward or reinforcement) to distinguish what is good and what is bad. The agent's task is to learn a policy or control strategy for choosing the best set of actions in such a long run that achieves its goal. For this purpose the agent stores a cumulative reward for each state or state-action pair. The ultimate objective of a learning agent is to maximize the cumulative reward it receives in the long run, from the current state and all subsequent next states along with goal state.

Reinforcement learning systems have four main elements [2]: policy, reward function, value function and model of the environment.

A policy defines the behavior of learning agent. It consists of a mapping from states to actions. A reward function specifies how good the chosen actions are. It maps each perceived state-action pair to a single numerical reward. In value function, the value of a given state is the total reward accumulated in the future, starting from that state. The model of the environment simulates the environment's behavior and may predict the next environment state from the current state-action pair and it is usually represented as a Markov Decision Process (MDP) [1, 3, and 4]. In MDP

A. Dobnikar, U. Lotrič, and B. Šter (Eds.): ICANNGA 2011, Part II, LNCS 6594, pp. 245–253, 2011.

Model, The agent senses the state of the world then takes an action which leads it to a new state. The choice of the new state depends on the agent's current state and its action.

An MDP is defined as a 4-tuple <*S*,*A*,*T*,*R*> characterized as follows: S is a set of states in environment, A is the set of actions available in environment, T is a state transition function in state *s* and action *a*, R is the reward function.

The optimal solution for an MDP is that of taking the best action available in a state, i.e. the action that collected as much reward as possible over time.

In reinforcement learning, it is necessary to introduce a process of trial and error to maximize rewards obtained from environment. This trial and error process is called an environment exploration. Because there is a trade-off between exploration and exploitation, balancing of them is very important. This is known as the exploration-exploitation dilemma. The schema of the exploration is called a policy. There are many kinds of policies such as ε-greedy, softmax, weighted roulette and so on. In these existing policies, exploring is decided by using stochastic numbers as its random generator.

It is ordinary to use the uniform pseudorandom number generator as the generator employed in exploration phase. However, it is known that chaotic source also provides a random-like sequence similar to stochastic source. Employing the chaotic generator based on the logistic map in the exploration phase gives better performances than employing the stochastic random generator in a nondeterministic maze problem. Morihiro et al. [5] proposed usage of chaotic pseudorandom generator instead of stochastic random generator in an environment with changing goals or solution paths along with exploration. That algorithm is severely sensitive to ε in ε-greedy. It is important to note that they don't use chaotic random generator in nondeterministic environments. In that work, it can be inferred that stochastic random generator has better performance in the case of using random action selection instead of ε-greedy one.

On the other hand, because of slowness in learning by reinforcement learning, evolutionary computation techniques are applied to improve learning in nondeterministic environments.

In this work we propose a modified reinforcement learning algorithm by applying population-based evolutionary computation technique and an application of the random-like feature of deterministic chaos as the random generator employed in its exploration phase, to improve learning in multi task agents. To sum up, our contributions are:

- Employing evolutionary strategies to reinforcement learning algorithm in support of increasing performance both in speed and accuracy of learning phase,
- Usage of chaotic generator instead of uniform pseudorandom number generator in the exploration phase of evolutionary reinforcement learning,
- Multi task learning in nondeterministic environments.

2 Chaotic Exploration

Chaos theory studies the behavior of certain dynamical systems that are highly sensitive to initial conditions. Small differences in initial conditions (such as those due to rounding errors in numerical computation) result in widely diverging outcomes for chaotic systems, and consequently obtaining long-term predictions impossible to take in general. This happens even though these systems are deterministic, meaning that their future dynamics are fully determined by their initial conditions, with no random elements involved. In other words, the deterministic nature of these systems does not make them predictable if the initial condition is unknown [6, 7].

As it is mentioned, there are many kinds of exploration policies in the reinforcement learning, such as ε-greedy, softmax, weighted roulette. It is common to use the uniform pseudorandom number as the stochastic exploration generator in each of the mentioned policies. There is another way to deal with the problem of exploration generators which is to utilize chaotic deterministic generator as their stochastic exploration generators [5]. As the chaotic deterministic generator, a logistic map which generates a value in the closed interval [0 1] according to equation 1, is used as stochastic exploration generators in this paper.

$$x_{t+1} = alpha\ x_t(1 - x_t). \tag{1}$$

In equation 1, x_0 is a uniform pseudorandom generated number in the [0 1] interval and alpha is a constant in the interval [0 4]. It can be showed that sequence x_i will be converged to a number in the [0 1] interval provided that the coefficient alpha be a number near to and below 4 [8, 9]. It is important to note that the sequence may be divergent for the alpha greater than 4. The closer the alpha to 4, the more different convergence points of the sequence. If alpha is selected 4, the vastest convergence points (maybe all points in the [0 1] interval) will be covered per different initializations of the sequence. So here alpha is chosen 4 to making the output of the sequence as similar as to uniform pseudorandom number.

3 Population Based Evolutionary Computation

One of research has done in Evolutionary Computation introduced by Handa [10]. It has used a kind of memory in Evolutionary Computation for storing past optimal solutions. In that work, each individual in population denotes a policy for a routine task. The best individual in current population is selected to insert in archive as environmental is changed. After that individuals in the archive are randomly selected to be moved into the population. The algorithm is called Memory-based Evolutionary Programming which is depicted in Fig 1.

A large number of studies concerning dynamic or uncertain environments have been made; have used Evolutionary Computation algorithms [11]. These problems try to reach their goal as soon as possible. The significant issue is that the robots could get assistance from their previous experiences.

In this paper a population based chaotic evolutionary computation for multitask reinforcement learning problems is examined.

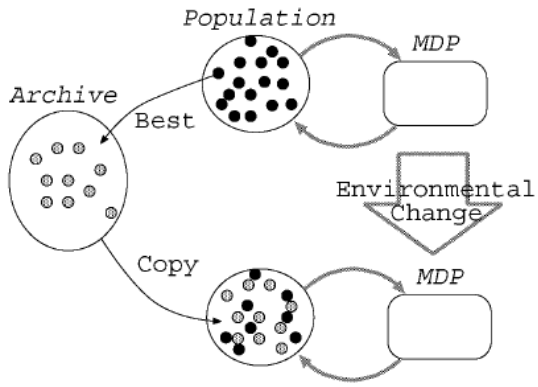

Fig. 1. Handa algorithm's Diagram for evolutionary computation

4 Q-Learning

Among reinforcement learning algorithms, Q-learning method is considered as one of the most important algorithms [1]. It consists of a Q-mapping from state-action pairs by rewards obtained from the interaction with the environment. In this case, the learned action-value function, Q, directly approximates Q*, the optimal action-value function, independent of the policy being followed. This simplifies the analysis of the algorithm and enabled early convergence proofs. The pseoduecode of Q-learning algorithm is shown in Fig 2.

Q-Learning Algorithm:

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., ε-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. 2. Q- Learning Algorithm

5 Evolutionary Reinforcement Learning

Evolutionary Reinforcement Learning (ERL) is a method of probing the best policy in RL problem by applying GA. In this case, the potential solutions are the policies and are represented as chromosomes, which can be modified by genetic operators such as crossover and mutation [12].

GA can directly learn decision policies without studying the model and state space of the environment in advance. The fitness values of different potential policies are used by GA. In many cases, fitness function can be computed as the sum of rewards, which are used to update the Q-values.

We use a modified Q-learning algorithm with applying memory-based Evolutionary Computation technique for improving learning in multi task agents [13].

6 Chaotic Based Evolutionary Q-Learning

With applying Genetic Algorithms for reinforcement learning in nondeterministic environments, we propose a Q-learning method called Evolutionary Q-learning. The algorithm is presented in Fig 3.

Chaotic Based Evolutionary Q Learning (CEQL):

Initialize Q(s,a) by zero

Repeat (for each generation):

Repeat (for each episode):

Initialize s

Repeat (for each step of episode):

Initiate($X_{current}$) by Rnd[0,1]

Repeat

$X_{next}=4 * X_{current} * (1- X_{current})$

Until ($X_{next} - X_{current} < \varepsilon$)

Choose a from s using X_{next}

Take action a, observe r, s'

$s \leftarrow s'$

Until s is terminal

Add visited path as a Chromosome to Population

Until population is complete

Do Crossover() by CRate

Evaluate the created Childs

Do tournament Selection()

Select the best individual for updating Q-Table as follows:

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

Copy the best individual in next population

Until satisfying convergence

Fig. 3. Chaotic base evolutionary Q- Learning Algorithm

7 Simulation and Results

7.1 Nondeterministic Maze Task

Assume that a number of robots are working in a mine and their task is to search for gold from an initial point. The mine has a group of corridors which robots can pass through them. In specific paths there exist some barriers which do not let robots to continue. Now, suppose that because of decadent corridors, it is possible that in some places, there could be some pits.

If a robot enters to such pits, it may be not to able exit from that pit with probability above zero by some moves. If it fails to exit by the moves, it has to try again. The aim of robots is finding the gold state as soon as possible. Note that the robots can use their past experiences.

In such problems, the shortest path may be not the best path; because it may have some pit cells and it leads agent acts many movement until finding goal state. Therefore the optimum path has less pit cells and short length. So applying an evolutionary version of Q-learning is more useful.

For validation of the proposed algorithm we turn to a modified version of Sutton's maze problem which is depicted in Fig 4. The original Sutton's maze problem consists of 6 x 9 cells, 46 common states, 1 goal state and 7 collision cells (gray cells in Fig 4). An agent can occupy each common state. It can't pass through collision cells. For each agent, there are at most four actions in each state to take: Up, Down, Left, and Right. The original Sutton's Maze problem is a deterministic problem [1].

In nondeterministic version of the problem one adds a number of probabilistic cells or holes (hachured cells in Fig 4). An agent can't leave each of the holes by its taken actions certainly and it is probable for them to remain in their previous states. That is, they have to stay the same cell with above zero probability; this probability is sampled from Normal distribution with average = 0, and variance = 1. For example, if an agent take Left action in a hole with transition probability according to Fig 5 a, next state may be the same state with probability of 0.6.

Agents will gain a reward +1, if they reach goal state. All other states don't give any reward to agents. There is no punishment in the problem.

7.2 Actions

Each action of agent could be an MDP sample such as delineated in Fig. 5.

Right part shows certain case which in it agents move to next state by choosing any possible action with probability equal 1. On the other hand left part reveals that it is possible that the agents cannot move to next state by choosing any possible action and it would remain in its position. For some do actions.

These MDPs presented to learning algorithm sequentially. The presentation time of each problem instance is enough to learn. The problem is to maximize the total acquired rewards for lifespan. Agents return to start state after arriving goal state.

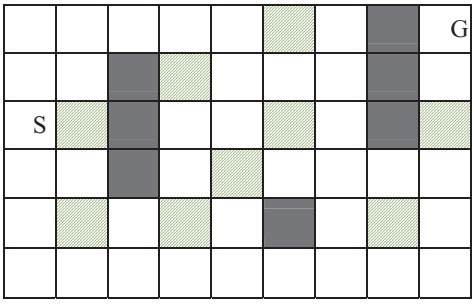

Fig. 4. Modified Sutton's maze

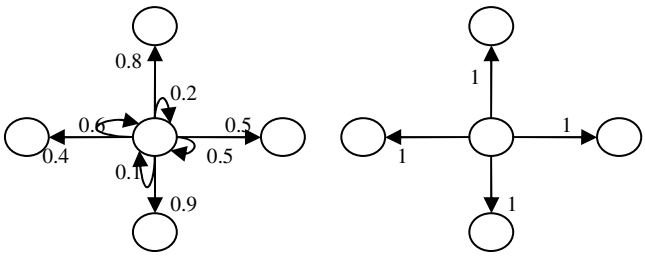

Fig. 5. Actions MDP Models

7.3 Experimental Results

A single experiment is composed of 100 tasks, where each task consists of 100 generations. In this work 50 experiments have been done. Hence, 500,000 generations in 50 experiments have been executed. In the case of Evolutionary Q-Learning and Chaotic Evolutionary Q-Learning, The sizes of population are set to 100.

Table 1 summarizes experiment results. As it can be inferred from the Table 1, usage of chaotic generator instead of stochastic random generator results in improvements, both in original Q-learning and evolutionary Q-learning in terms of averaged path length. Usage of chaotic generator in Original Q-Learning results in 6.85% improvement. Also the usage of it in the Evolutionary Q-Learning results in 5.09% improvement. This isn't unexpected, because it is explored in [5] before and the superior of chaotic generator over stochastic random generator in exploration phase of reinforcement learning has been shown. But here, as it is reported in [12] the evolutionary reinforcement learning can improve the average found paths significantly comparing with the average paths found by original version of reinforcement learning. Also it is shown here that the performance of Evolutionary Q-Learning can precede Chaotic based Q-learning. So it can be expected that the employing of chaotic generator instead of stochastic random generator in evolutionary reinforcement learning can also yields to a better performance. As it is reported in the table 1, the chaotic based evolutionary reinforcement learning can improve the average found paths comparing with

Table 1. Experimental Result

	Best average of path length in population	worst average of path length in population	Total Average of path length
OQL	54.26	1600	449.1149
CQL	60.49	1500.33	418.33
EQL	28.34	66.79	42.4738
CEQL	25.11	99.99	40.3098
Improvement CEQL to OQL	53.7%	93.7%	91%

OQL: Original Q-learning
CQL: Chaotic based Q-learning
EQL: Evolutionary Q-Learning
CEQL: Chaotic based Evolutionary Q-learning

the average paths found by non-chaotic version of evolutionary reinforcement learning by 5.09% expectedly. So it can be concluded that leveraging the chaotic random generator in exploration phase of the reinforcement learning can be effective in a better environment exploration both in its evolutionary base version or its non-evolutionary one.

8 Conclusion

In reinforcement learning, exploration phase takes a significant role aiming to fastness in learning period. Different algorithms exist as random number generator in exploration phase. We showed that the deterministic chaotic generator for the exploration gives better performances than the stochastic random generator.

For improving the slowness of Q-learning, we applied genetic algorithms and used Population based evolutionary computation along with chaos as random generator in exploration phase to improve efficiency in nondeterministic environment. Sutton's Maze is a well-known problem in reinforcement learning field and we implement a modified version of it for evaluation proposed algorithm in nondeterministic environment.

Our proposed algorithm has about 91.02% improvement compared to original Q-learning algorithm in average case of found path length. Also it has also about 5.09% improvement compared with non-chaotic evolutionary Q-learning in average.

This can be inferred from experimental results that employing chaos as random generator in exploration phase as well as evolutionary-based computation have improved the reinforcement learning, both in rate of learning and accuracy. It can also be concluded that using chaos in exploration phase is efficient for both evolutionary based version and non-evolutionary one.

References

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
2. Cuayáhuitl, H.: Hierarchical Reinforcement Learning for Spoken Dialogue Systems, PhD thesis, University of Edinburgh (2009)
3. Vidal, J.M.: Fundamentals of Multi Agent Systems (2009) (unpublished)
4. Shoham, Y., Leyton-Brown, K.: MULTIAGENT SYSTEMS Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, Cambridge (2009)
5. Morihiro, K., Matsui, N., Nishimura, H.: Effects of Chaotic Exploration on Reinforcement Maze Learning. In: Negoita, M.G., Howlett, R.J., Jain, L.C. (eds.) KES 2004. LNCS (LNAI), vol. 3213, pp. 833–839. Springer, Heidelberg (2004)
6. Kellert, S.H.: In the Wake of Chaos: Unpredictable Order in Dynamical Systems. University of Chicago Press, Chicago (1993) ISBN 0226429768
7. Meng, X.P., Meng, J., Lui, L.J.: Quantum Chaotic Reinforcement Learning. In: Fourth International Conference on Natural Computation (2008)
8. Parker, T.S., Chua, L.O.: Practical Numerical Algorithms for Chaotic Systems. Springer, Heidelberg (1989)
9. Ott, E., Sauer, T., Yorke, J.A.: Coping with Chaos: Analysis of Chaotic Data and the Exploitation of Chaotic Systems. John Wiley & Sons, Inc., New York (1994)
10. Handa, H.: Evolutionary Computation on Multitask Reinforcement Learning Problems. In: IEEE International Conference on Networking, Sensing and Control, pp. 685–688 (2007)
11. Goh, K., Tan, K.: Evolutionary Multi-objective Optimization in Uncertain Environments. Springer, Heidelberg (2009)
12. Jiang, J.: A Framework for Aggregation of Multiple Reinforcement Learning Algorithms. PhD thesis, University of Waterloo (2007)
13. Beigi, A., Parvin, H., Mozayani, N., Minaei, B.: Improving Reinforcement Learning Agents Using Genetic Algorithms. In: An, A., Lingras, P., Petty, S., Huang, R. (eds.) AMT 2010. LNCS, vol. 6335, pp. 330–337. Springer, Heidelberg (2010)