A Q-learning Based Evolutionary Algorithm for Sequential Decision Making Problems

Haobo Fu[1], Peter R. Lewis[2], and Xin Yao[1]

[1] CERCIA, School of Computer Science, University of Birmingham, UK
[2] School of Engineering and Applied Science, Aston University, UK
hxf990@cs.bham.ac.uk, p.lewis@aston.ac.uk, x.yao@cs.bham.ac.uk

Abstract. Both Evolutionary Dynamic Optimization (EDO) methods and Reinforcement Learning (RL) methods tackle forms of Sequential Decision Making Problems (SDMPs), yet with different key assumptions. In this paper, we combine the strength of both EDO methods and RL methods to develop a new algorithm for SDMPs. Assuming that the environmental state is observable and that a computational model of the reward function is available, the key idea in our algorithm is to employ an evolutionary algorithm to search on the reward function at each time step, the outcome of which is exploited to speed up convergence to optimal policies in RL methods. Some preliminary experimental studies demonstrate that our algorithm is a promising approach for SDMPs.

Keywords: Reinforcement Learning, Evolutionary Dynamic Optimization, Sequential Decision Making

1 Introduction

Sequential Decision Making Problems (SDMPs) are those where a decision maker is required to make a sequence of decisions over time, and the overall performance is dependent on all the decisions made. In the Reinforcement Learning (RL) community, SDMPs are mainly studied as Markov Decision Problems (MDPs), whereas in the Evolutionary Dynamic Optimization (EDO) community, SDMPs are usually interpreted as tracking moving optimum problems. More formally, SDMPs[3] can be defined as follows:

Definition 1. *Consider a discrete time interval $[0, t_e]$, during which the environmental state at time step t, $\boldsymbol{s}_t$, follows a probabilistic distribution $P(\boldsymbol{s}_t|\boldsymbol{s}_0, ..., \boldsymbol{s}_{t-1}, \boldsymbol{x}_0, ..., \boldsymbol{x}_{t-1})$, where $\boldsymbol{s}_i$ and $\boldsymbol{x}_i$ denote the environmental state at time step i and the decision/action made at time step i respectively, $0 \leq i \leq t-1$. A SDMP can be stated as making a sequence of decisions, one at each time step during $[0, t_e]$, so that the decision sequence maximizes the expected accumulated rewards[4] over the entire time interval $[0, t_e]$:*

$$\max \sum_{i=0}^{t_e} E(f_i(\boldsymbol{s}_i, \boldsymbol{x}_i)), \tag{1}$$

[3] We consider SDMPs and dynamic optimization problems (as defined in [1]) to be interchangable terms, describing the same general problem form.

[4] In general, we maximize the discounted accumulated rewards: $\sum_{i=0}^{t_e} E(\gamma^i \cdot f_i(\mathbf{s}_i, \mathbf{x}_i))$, where γ is a user defined discount factor, $0 \leq \gamma \leq 1$. When t_e goes to infinity, $0 \leq \gamma < 1$. In this paper, we only consider finite t_e with $\gamma = 1$.

where f_i is the reward function at time step i, which returns an immediate reward of an action taken at time step i in state $\mathbf{s}_i$. f_i can be either deterministic or stochastic. $E()$ returns the expected value of the random variable $f_i(\mathbf{s}_i, \mathbf{x}_i)$. Also, some computational time is allowed for the decision maker to make a decision at each time step.

As demonstrated in [1], both RL methods and EDO methods tackle forms of SDMPs, albeit forms with different key assumptions. More specifically, for RL methods, the key assumption is that the environmental state **s** in Equation 1 is at least partially observable to the decision maker. For EDO methods, the key assumption is that the decision maker is able to evaluate the immediate reward f of a decision at each time step without actually implementing that decision. In other words, a computational model of $f(\mathbf{s}, \mathbf{x})$ at each time step is assumed to be available in EDO methods.

In this paper, we combine the strength of learning in RL methods and the strength of optimizing in EDO methods to develop a new algorithm for SDMPs, assuming that the environmental state **s** is observable and that a computational model of $f(\mathbf{s}, \mathbf{x})$ at each time step is available. The main idea in our algorithm is to use an Evolutionary Algorithm (EA) to search on $f(\mathbf{s}, \mathbf{x})$ at each time step, and the search result from the EA is exploited to speed up convergence to optimal decisions in a RL method. The new algorithm is described in detail in Section 2, followed by some preliminary experimental studies in Section 3. Conclusions and future work are discussed in Section 4.

2 A New Algorithm for Sequential Decision Making

We base our new algorithm on Q-learning [2] and EAs, and hence call it the Q-learning Based Evolutionary Algorithm (QBEA). Given a computational model of the reward function f at each time step, an EA is used in QBEA to search on f, and all evaluated decisions/actions together with their immediate rewards are then used to update the Q values to hopefully gain a faster convergence to optimal Q values. The pseudo code of QBEA is given in Algorithm 1.

Algorithm 1 Pseudo code of QBEA

```
1:  Arbitrarily initialize the Q values Q(s, x) for each state-action pair (s, x);
2:  Observe the initial state s_0;
3:  for t = 0 → t_e do
4:      Employ an EA to search on the reward function f_t(s_t, x);
5:      for each evaluated x_i on f_t do
6:          Q(s_t, x_i) ← (1 − α)Q(s_t, x_i) + α(f_t(s_t, x_i) + γ max_j Q(ŝ, x_j));
7:      end for
8:      Select an action x_t in s_t based on the Q values;
9:      Take action x_t, and observe the immediate reward r and a new state s_{t+1};
10:     Q(s_t, x_t) ← (1 − α)Q(s_t, x_t) + α(r + γ max_j Q(s_{t+1}, x_j));
11: end for
```

In Line 6 of Algorithm 1, when we evaluate $\mathbf{x}_i$ on f_t in state $\mathbf{s}_t$, we are not sure which new state $\mathbf{x}_i$ would lead to if $\mathbf{x}_i$ is implemented. Therefore, in order to update the

Q value $Q(\mathbf{s}_t, \mathbf{x}_i)$, we sample the currently estimated probability distribution $P(\mathbf{s}|\mathbf{s}_t, \mathbf{x}_i)$ (assuming it is Markovian) to produce a state $\hat{\mathbf{s}}$.

3 Preliminary Experimental Studies

In this section, we conduct some preliminary experimental studies of QBEA on two benchmark instances of the conceptualised dynamic optimization problem benchmark we developed in [1].

For both benchmark instances, the reward function at time step t is:

$$f_t(\mathbf{s}_t, x_t) = 30 - 2|x_t - c_t| + b_t, \tag{2}$$

where $\mathbf{s}_t$ represents the state at time step t: $\mathbf{s}_t = (c_t, b_t)$. x_t belongs to the interval $[-10, 10]$ and represents the decision at time step t. $c_0 = 5$, and $c_t = c_{t-1} * -1$. The dynamic of the bias b_t is as follows ($b_0 = \theta_b$):

$$b_t = \begin{cases} \theta_b & if \quad x_{t-1} \geq 0, \\ -\theta_b & otherwise, \end{cases} \tag{3}$$

where θ_b is a parameter, which controls the influence of x_{t-1} on b_t, with larger values being more influential. In the first benchmark instance, θ_b is set to 100, and in the second benchmark instance, θ_b is set to 15.

We compare QBEA to an ideal EDO method, where at each time step a decision that maximizes the current reward function is implemented (denoted as 'EDO' hereafter), but without considering any impact of the action on the future. We compare QBEA also to the Q-learning algorithm (denoted as 'Q-learning' hereafter). We use semi-uniform random exploration strategy to select an action in 'Q-learning', where at each time step the best action in terms of the Q value is selected with probability $1 - \epsilon$. Otherwise, a random action is chosen. ϵ is set to 0.1. The learning rate α in 'Q-learning' is updated as $\alpha = q_1/(q_2 + t)$, where q_1 and q_2 are set to 200 and 300 respectively, and t is the index of the time step. The discount factor in 'Q-learning' is set to 0.7. For 'Q-learning', the action space is equally discretized into 21 different actions: $\{-10, -9, -8, ..., 8, 9, 10\}$. The Q values in 'Q-learning' are all initialised to 0 at the beginning of the time (i.e., time step $t = 0$). The settings of QBEA is the same as 'Q-learning' except that all actions are evaluated at each time step and that the true probability distribution $P(\mathbf{s}|\mathbf{s}_t, \mathbf{x}_i)$ is used to generate $\hat{\mathbf{s}}$.

The performance of 'EDO', 'Q-learning', and QBEA, together with the optimal accumulated rewards, on the first and second benchmark instances are presented in Figures 1 and 2 respectively. We can see that QBEA achieves a consistently better performance than 'EDO' and 'Q-learning' in both instances.

4 Conclusions and Future Work

In this paper, we developed a new algorithm, QBEA, for SDMPs. Assuming that the environmental state is observable and that a computational model of the reward function is available, the main idea in QBEA is that an EA is used to search on the reward

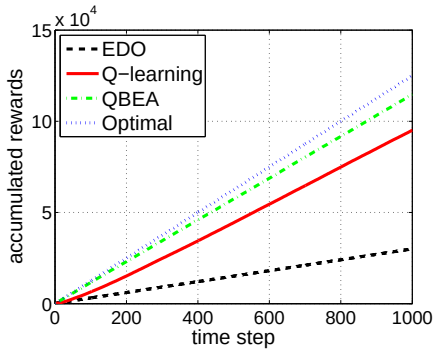

Fig. 1. The averaged accumulated rewards in Equation 1 over 30 runs at each time step.

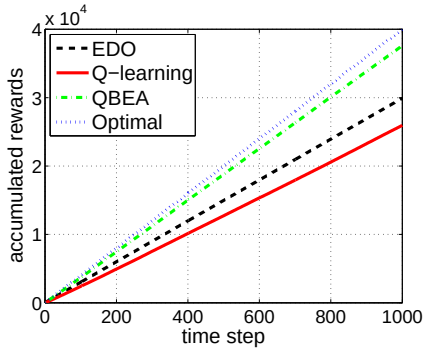

Fig. 2. The averaged accumulated rewards in Equation 1 over 30 runs at each time step.

function, and the evaluation experience of the EA is used to update the Q values in Q-learning, which can speed up the convergence in Q-learning. Some preliminary experimental studies demonstrated the effectiveness of QBEA over an ideal EDO method and the Q-learning algorithm.

Some enhancements to QBEA are still needed to make it applicable to more general cases. Firstly, a regression model is needed to generalize QBEA to continuous state and action space. Secondly, in cases where it is impossible to evaluate all the actions at a time step, some mechanisms are required as to which solutions are evaluated in an EA. Finally, models to estimate the probability distribution $P(\mathbf{s}|\mathbf{s}_t, \mathbf{x}_i)$ is lacking in the current version of QBEA.

References

1. H. Fu, P. R. Lewis, B. Sendhoff, K. Tang, and X. Yao. What are dynamic optimization problems? In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, 2014. in press.
2. C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.