SELECTION OF EXTRA OBJECTIVES USING REINFORCEMENT LEARNING IN NON-STATIONARY ENVIRONMENT: INITIAL EXPLORATIONS

Irina Petrova, Arina Buzdalova and Maxim Buzdalov

ITMO University
Computer Technologies Laboratory
49 Kronverkskiy prosp., Saint-Petersburg, Russia
petrova@rain.ifmo.ru, abuzdalova@gmail.com, mbuzdalov@gmail.com

Abstract: *Using extra objectives in evolutionary algorithms helps to avoid getting stuck in local optima and increases genetic diversity. We consider a method based on a reinforcement learning (RL) algorithm that selects objectives in evolutionary algorithms (EA) during optimization. The method is called EA+RL. In some researches, reinforcement learning algorithms for stationary environments were used to adjust evolutionary algorithms. However, when properties of extra objectives change during the optimization process, we propose that it is better to use reinforcement learning algorithms which are specially developed for non-stationary environments. We present an initial research towards EA+RL for a non-stationary environment. A new reinforcement learning algorithm is proposed to be used in the EA+RL method. We also formulate a benchmark problem with some extra objectives, which behave differently at different stages of optimization. Thus, non-stationarity arises. The new algorithm is applied to this problem and compared with the methods which were used in other researches. It is shown that the proposed method chooses the extra objectives which are efficient at the current optimization stage more often and obtains higher values of the target objective being optimized.*

Keywords: *evolutionary algorithms, fitness function, multiobjectivization, reinforcement learning, non-stationary*

1 Introduction

Using extra objectives in evolutionary algorithms helps to avoid getting stuck in local optima and increases genetic diversity [9]. The corresponding approach is called multiobjectivization. Several methods of multiobjectivization and details of one of them, the previously proposed EA+RL method, are described below.

1.1 Multiobjectivization

Some methods of multiobjectivization that increase efficiency of evolutionary algorithms (EA) exist. These methods help to avoid local optima [7,9]. One of them is based on decomposing the target objective into several components, which are optimized simultaneously [9]. The components should be independent, but it is not always easy or possible. Another method is to use some additional objectives that are used in combination with the original objective [7]. This approach was successfully applied to some practical problems, for example, the Job Shop Scheduling Problem and the Traveling Salesman Problem [7, 10]. At each step of the algorithm, one or several extra objectives are optimized [10]. They can be selected in random order [7] or using an ad-hoc heuristic [10]. The first approach is general, but does not take into account any features of an optimization problem, while the second one can be applied only to a specific problem. The EA+RL method was designed to deal with these issues [2].

1.2 EA+RL Method

In the EA+RL method, reinforcement learning (RL) [13] is used to select an objective, which is optimized in the current iteration of an EA. So the EA is treated as an environment for the RL agent. The method was previously shown to be efficient for a number of problems [2].

In RL, an agent applies some action to an environment, then the environment returns some representation of its state as well as a numerical reward, and the process repeats. The EA+RL method guides the EA using RL by choosing optimization objectives, or fitness functions, for each iteration. The EA is treated as an environment. Each action of the agent corresponds to an objective to be optimized at the current iteration. There is the target objective, which should be optimized by the EA, and a set of extra objectives. Note that we do not aim to

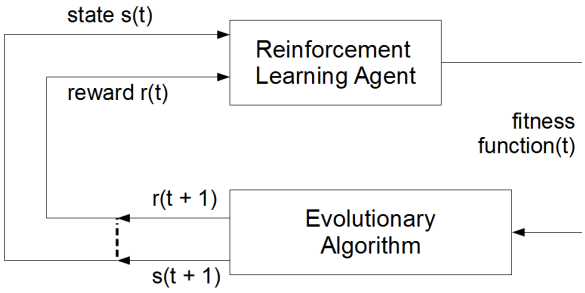

Figure 1: EA+RL general scheme

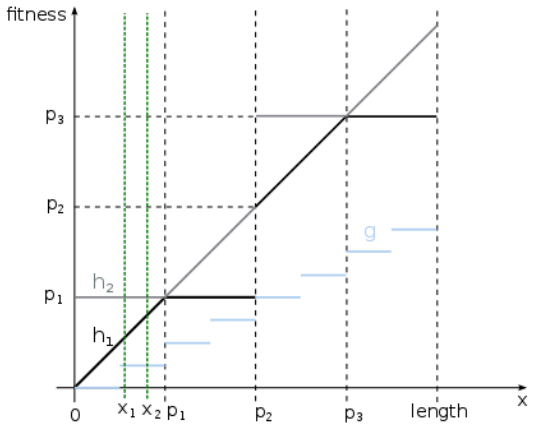

Figure 2: Benchmark problem

maximize the extra objectives, they are just used to increase the efficiency of the target objective optimization. The method is illustrated in Fig. 1, where t is the number of the current iteration of the EA.

Generally, the goal of RL is to maximize the total reward [13]. In the EA+RL method, the reward is based on the difference of the target objective values in two sequential iterations. The total reward, which is roughly equivalent to the difference between the final and the initial values of the target objective, is maximized. So, the target objective is maximized by RL and its optimization by EA is not necessary. Therefore, on each iteration only the selected objective is optimized by the EA. Hence, we can use a single-objective EA which typically runs faster than a multi-objective EA.

In some previous researches, it was implied that the environment was stationary. Precisely, the result of the action depended only on the state of the environment. Consequently, RL algorithms for stationary environments, such as Q-learning, were used. However, in the case when properties of extra objectives depend on the stage of the optimization process, the result of the action can be different in the same RL state. Therefore, RL algorithms for non-stationary environments should probably be used.

The rest of the paper is organized as follows. First, a benchmark problem is described. Second, we analyze existing reinforcement learning algorithms for non-stationary environments and propose a new RL algorithm, which is specially designed to be efficient when dealing with evolutionary algorithms. Then the experiments are described. Finally, the proposed RL algorithm is compared with the other RL algorithms, which were studied in other researches of EA+RL.

2 Benchmark Problem

In [1], a benchmark problem with two extra objectives which can be both efficient and inefficient at different optimization stages was studied. In the current research, a more complex problem, which has the similar properties, is described.

In this problem, an individual is a bit string of length n. Let x be the number of bits in an individual which are set to one. Then the target fitness function (objective) is $g(x) = \lfloor \frac{x}{k} \rfloor$, where k is constant, $k < n$ and k divides n. The extra objectives are $h_1(x)$ and $h_2(x)$, which are described below (1).

$$h_1(x) = \begin{cases} x, x \leq p_1 \\ p_1, p_1 < x \leq p_2 \\ x, p_2 < x \leq p_3 \\ \dots \\ x, p_s < x \leq n \end{cases} \quad h_2(x) = \begin{cases} p_1, x \leq p_1 \\ x, p_1 < x \leq p_2 \\ p_3, p_2 < x \leq p_3 \\ \dots \\ n, p_s < x \leq n \end{cases} \tag{1}$$

Here, each p_i is called a *switch point*. Hence, s is the number of switch points. Note that at each switch point the properties of h_i are changed. The extra-objectives and target objective are illustrated in Fig. 2.

The h_1 objective is efficient when $x \in [0, p_1], (p_2, p_3] ... (p_s, n]$ while the h_2 objective is efficient in all other cases. Note that using the proper extra objective allows to distinguish individuals with the same value of the target objective and give preference to the individual with higher x value. Such an individual is more likely to produce a descendant with a higher target objective value. Ideally for each EA iteration an extra objective which is not constant at the current interval of x values should be selected.

3 Reinforcement Learning in Non-stationary Environments

RL algorithms designed for non-stationary environments are based on algorithms of RL in stationary environments. There exist *model-free* and *model-based* algorithms. In model-free algorithms, the expected reward estimation is updated using the agent experience. In model-based algorithms, the agent experience is used to update the models of transition and reward. The expected reward estimation is updated using these models.

Consider existing methods of RL in non-stationary environments. There exist methods for the case of continuous time and space [4]. However, we consider discrete optimization problems. Some RL algorithms for discrete problems are designed for the case of non-stationary reward function [6]. Such algorithms are not relevant to our research as well, since the reward function in EA+RL does not change.

The Reinforcement Learning Context Detection (RLCD) algorithm is the closest to our needs [11]. This algorithm deals with a non-stationary environment which can be presented as a composition of stationary contexts. Models for different contexts are maintained using model-based RL algorithms, such as Dyna or Prioritized Sweeping [8]. Quality of a model depends on its ability to predict behavior of the environment. At each algorithm step, the most qualitative model is active.

In our case, a context should be switched when properties of an extra objective change. We have tried to apply the RLCD algorithm in the EA+RL approach for solving the benchmark problem. However, it appeared that contexts were not switched when it was needed. What is more, too many models were created, which slowed down the learning process. So the preliminary experiment showed that RLCD was not efficient for the considered problem. Therefore, we developed another RL approach inspired by RLCD.

4 Proposed Approach

The preliminary experiment showed that the best results are obtained using Q-learning with greedy exploration strategy which is a model-free RL algorithm [13]. So, this RL algorithm is used in the proposed approach. As in the conventional Q-learning, at each iteration, the RL agent selects an action a and applies it to the environment, which is at state s. Then the expected reward estimation $Q(s, a)$ is updated using the obtained reward. The core idea of the approach is to reset Q values when the context is changed. Q values are reset when the following condition is fulfilled: $|Q(s, a) - Q(s', a')| < \delta$ for some pair (s, a) and (s', a'), where δ is some constant, called *distinction factor*. In this case the expected reward is roughly the same for at least one pair of the actions and it is hard to decide properly which action to choose. We assume that such situation indicates that the context has changed and previously obtained Q values are not relevant. The pseudocode of the proposed approach is presented in Algorithm 1.

The main difference between the proposed approach and RLCD is that, unlike RLCD, a model-free RL algorithm is used in the new approach. This changed the way the context change is detected.

5 Experiment Description

Several configurations of the formulated benchmark problem were solved using EA+RL with three different RL algorithms. All the considered algorithms were run 100 times on every problem instance, then the results were averaged. Configurations with 5 and 10 switch points were considered. Different values of individual length n were considered. For the case of 10 switch points, individuals with greater lengths were used. The switch points were evenly distributed along the length of an individual. The cases of $k = 10$ and $k = 25$ were considered.

There were 100 individuals in a generation. Mutation operator flipped each bit with the probability of 0.001. Shift crossover [3] was applied with the probability of 0.7. Problem instances with 5 and 10 switch points were optimized for 3000 and 7000 iterations, correspondingly. The considered RL algorithms were Q-learning with ε-greedy strategy [13], Delayed Q-learning [12] and the proposed algorithm. The first two algorithms were previously used in the EA+RL method for various problems. Their parameters were taken from [2]. For the ε-greedy Q-learning, learning rate was set to $\alpha = 0.6$, discount factor was $\gamma = 0.01$, exploration probability was $\varepsilon = 0.3$. In Delayed Q-learning [12] the following parameters were used: update period $m = 5$, discount factor $\gamma = 0.01$, bonus reward $\varepsilon = 0.4$. Parameters of the proposed method were set during a preliminary experiment. They were: learning rate $\alpha = 0.6$, discount factor $\gamma = 0.01$ and distinction factor $\delta = 0.001$. The reward was calculated as the difference of the target objective value calculated on the best individuals in two successive iterations of the EA. As in [2], RL states were represented as vectors of the fitness functions ordered by the value of $(f(x_c) - f(x_p))/f(x_c)$, where f is a fitness function to be measured, x_p is the number of bits set to one in the best individual from the previous generation, and x_c is the number of bits set to one in the best individual from the current generation.

The Wilcoxon signed ranks test was performed to check if the new algorithm and the previously used ones are statistically distinguishable. This test is suitable for analyzing of results obtained by evolutionary approaches, since it is non-parametric [5]. Statistical significance was tested at $\alpha = 10^{-2}$ level.

Algorithm 1 The proposed approach: Q-learning with resets

Form initial generation G_0
Initialize $Q(s, a) \leftarrow 0$ for each state s and action a
Initialize iteration counter: $k \leftarrow 0$
while (specified number of generations or maximum value of target objective not reached) **do**
  Evaluate current state s_k and pass it to agent
  Select action: a where $Q(s, a) \leftarrow \max_{a'} Q(s, a')$
  Generate next generation G_{k+1}
  Calculate reward r
  $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s, a') - Q(s, a))$
  Initialize reset identifier: $reset \leftarrow false$
  for (each pair (s, a)) **do**
    for (each pair (s', a')) **do**
      if $Q(s, a) \neq 0$ and $Q(s', a') \neq 0$ and $|Q(s, a) - Q(s', a')| \leq \delta$ **then**
        Reset of learning is needed: $reset \leftarrow true$
      end if
    end for
  end for
  if reset of learning is needed: $reset = true$ **then**
    Set *Q(s, a)* for each s and a to zero : $Q(s, a) \leftarrow 0$
  end if
  Update iteration counter: $k \leftarrow k + 1$
end while

6 Experiment Results

Results for the $k = 10$ and $k = 25$ are shown in Table 1. The first column of each table contains the number of switch points. The second column contains the length of an individual. The last three columns contain average fitness achieved using new method, ε-greedy and Delayed Q-learning, respectively. The deviation of the average fitness obtained by the first two algorithms is about 0.5%, the deviation obtained using Delayed Q-learning is about 20%. For all analyzed problem instances the new method seems to outperform previously used ones. It can be noticed that results obtained using Delayed Q-learning algorithm are much worse than the other results.

The p-values obtained when comparing with ε-greedy and Delayed Q-learning are presented in parentheses. In the case when the length is greater than 1000 p-values are less than 8×10^{-3}, so in that case the new method is statistically distinguishable from ε-greedy. However, there are instances for which the Delayed Q-learning and the new method are not distinguishable, although average fitness achieved in the new method is much higher. This can be due to the fact that the deviation for the Delayed Q-learning algorithm is very high compared to the one for the proposed method.

The RL agent can choose h_0, h_1 or target objective at each iteration. The most efficient choice is the choice of an extra objective, which is equal to x at the current generation. We call such a choice a *good* one. The Fig. 3 shows the number of different current fitness function choices made by the algorithms in optimization problem with 5 switch points, $k = 10$, $n = 750$. The horizontal axis refers to the number of an iteration, the vertical axis refers to the number of different choices made at the corresponding generation in 100 runs. At first the proposed method makes a good choice (h_1) more often than other methods, then the proposed method makes a good choice (h_2) more often than other methods and so on. Therefore, the new method makes good choices more often than ε-greedy algorithm, and the latter makes good choices more often than Delayed Q-learning.

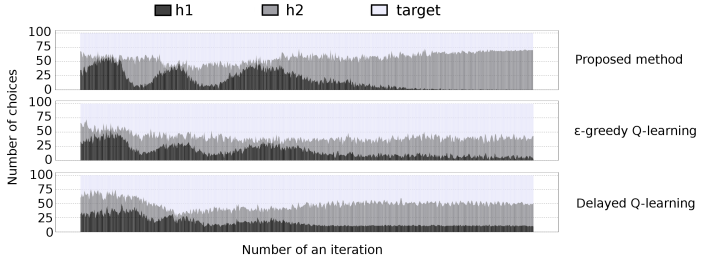

Figure 3: Choices of objectives

Table 1: Average target objective values

Points	Iterations	Length	New method	ε-greedy	Delayed Q-learning
$k = 10$					
5	3000	750	74.52	74.49 (3.4×10^{-1})	68.39 (4.4×10^{-10})
		1000	99.55	99.47 (1.3×10^{-1})	87.39 (2.2×10^{-16})
		1250	124.44	124.19 (8.3×10^{-4})	113.69 (2.2×10^{-16})
		1500	149.03	148.02 (1.5×10^{-3})	133.50 (8.1×10^{-15})
		1750	173.98	173.63 (8.4×10^{-8})	156.93 (1.9×10^{-13})
	5000	2000	198.93	197.98 (2.2×10^{-16})	186.37 (9.5×10^{-14})
		2250	222.01	220.23 (7.2×10^{-11})	202.80 (1.5×10^{-3})
		2500	245.52	244.55 (3.0×10^{-3})	232.81 (9.9×10^{-1})
10	5000	2000	198.94	198.34 (1.8×10^{-12})	170.59 (2.7×10^{-13})
		2250	223.36	220.79 (2.2×10^{-16})	184.47 (1.1×10^{-12})
		2500	245.28	244.61 (1.2×10^{-4})	204.42 (7.2×10^{-1})
	9000	2750	269.38	269.14 (5.0×10^{-3})	226.14 (5.7×10^{-1})
		3000	294.22	293.73 (2.8×10^{-5})	249.96 (9.6×10^{-1})
		3250	318.92	318.70 (1.4×10^{-2})	268.66 (9.7×10^{-1})
		3500	343.79	343.33 (4.0×10^{-5})	285.76 (9.9×10^{-1})
		3750	368.52	367.90 (1.3×10^{-5})	307.72 (9.9×10^{-1})
$k = 25$					
5	3000	750	29.45	29.40 (2.5×10^{-1})	26.01 (2.2×10^{-16})
		1000	39.18	39.14 (2.2×10^{-1})	36.20 (8.9×10^{-14})
		1250	49.02	49.00 (2.4×10^{-1})	45.86 (2.8×10^{-9})
		1500	59.00	58.92 (6.0×10^{-3})	54.77 (4.3×10^{-8})
		1750	68.96	68.02 (4.0×10^{-15})	61.17 (1.8×10^{-11})
	5000	2000	78.17	77.23 (4.9×10^{-16})	70.54 (1.9×10^{-1})
		2250	87.30	87.12 (8.0×10^{-3})	79.70 (9.8×10^{-1})
		2500	97.01	96.89 (4.0×10^{-3})	84.62 (5.3×10^{-1})

In Table 2 averaged percentage of good choices is presented for the constant $k = 10$. The first column contains the number of switch points. The second column contains the length of an individual. The last three columns contain average number of good choices in percentage of all fitness function choices using the proposed approach (new), ε-greedy Q-learning (Q) and Delayed Q-learning (DQ) respectively. Deviation of the average choices percentage obtained by the first two algorithms is about 8%, by Delayed Q-learning – about 100%. The p-values obtained when comparing with ε-greedy and Delayed Q-learning are presented in parentheses. In the case when the length is greater than 1000 p-values are less than 7×10^{-4}. Hence, in that case the new method is statistically distinguishable from the previously used ones.

It can be noted that the proposed method makes good choices more often than the ε-greedy. However, there exist problem instances on which Delayed Q-learning was choosing the effective extra-objective more often than the new algorithm. At the same time, in this case the average fitness value obtained by using Delayed Q-learning is much worse than the one obtained by using the proposed approach, as shown in Table 1. It can be explained by the fact that the corresponding fitness and choices deviation for the Delayed Q-learning algorithm is very high compared to the one for the proposed method. The results concerning the number of different choices for the case of $k = 25$ are quite similar. For brevity, they are not presented.

7 Conclusion

A new reinforcement learning approach was proposed, which can be used in the EA+RL framework. It was designed for dealing with non-stationarity, which arises when properties of extra objectives depend on the stage of optimization. The proposed approach was used to solve a benchmark problem. The achieved results are better than the results obtained with ε-greedy Q-learning and Delayed Q-learning algorithms. The Wilcoxon signed ranks test showed that the performance of the proposed method is statistically distinguishable from the others. In the future research the proposed algorithm can probably be improved by using the previous experience, i. e. storing Q values obtained before reset and re-using them.

Acknowledgement: This work was financially supported by the Government of Russian Federation, Grant 074-U01.

Table 2: Different objective choices, $k = 10$

Points	Length	Good,%		
		New	Q	DQ
5	750	62	50 (1.3×10^{-5})	46 (8.0×10^{-3})
	1000	55	51 (1.0×10^{-2})	47 (2.6×10^{-1})
	1250	51	43 (7.0×10^{-4})	36 (1.7×10^{-5})
	1500	50	39 (2.2×10^{-16})	35 (1.0×10^{-10})
	1750	48	39 (2.2×10^{-16})	37 (9.2×10^{-10})
	2000	46	39 (2.2×10^{-16})	29 (2.1×10^{-15})
	2250	37	23 (2.2×10^{-16})	37 (8.5×10^{-7})
	2500	30	17 (2.2×10^{-16})	26 (4.1×10^{-14})
10	2000	47	49 (1.3×10^{-8})	33 (6.9×10^{-13})
	2250	42	29 (2.2×10^{-16})	36 (3.6×10^{-6})
	2500	26	19 (2.2×10^{-16})	38 (1.2×10^{-4})
	2750	29	21 (2.2×10^{-16})	41 (3.9×10^{-3})
	3000	34	26 (2.2×10^{-16})	33 (5.1×10^{-7})
	3250	39	29 (2.2×10^{-16})	36 (1.6×10^{-5})
	3500	41	33 (2.2×10^{-16})	38 (4.6×10^{-5})
	3750	43	35 (2.2×10^{-16})	37 (4.6×10^{-5})

References

[1] Afanasyeva, A., Buzdalov, M.: Choosing Best Fitness Function with Reinforcement Learning. In: Proceedings of the Tenth International Conference on Machine Learning and Applications, vol. 2, pp. 354–357. IEEE Computer Society, Honolulu, HI, USA (2011)

[2] Afanasyeva, A., Buzdalov, M.: Optimization with Auxiliary Criteria using Evolutionary Algorithms and Reinforcement Learning. In: Proceedings of 18th International Conference on Soft Computing MENDEL 2012, pp. 58–63. Brno, Czech Republic (2012)

[3] Arkhipov, V., Buzdalov, M., Shalyto, A.: Worst-Case Execution Time Test Generation for Augmenting Path Maximum Flow Algorithms using Genetic Algorithms. In: Proceedings of the International Conference on Machine Learning and Applications, vol. 2, pp. 108–111. IEEE Computer Society (2013)

[4] Basso, E.W., Engel, P.M.: Reiforcement learning in non-stationary continuous time and space scenarios. In: Proceedings of the VII Brasilian Meeting on Artificial Intelligence ENIA, pp. 687–696. SBC Press (2009)

[5] Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3–18 (2011)

[6] Granmo, O.C., Berg, S.: Solving non-stationary bandit problems by random sampling from sibling kalman filters. In: IEA/AIE (3), pp. 199–208 (2010)

[7] Jensen, M.T.: Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation: Evolutionary Computation Combinatorial Optimization. Journal of Mathematical Modelling and Algorithms **3**(4), 323–347 (2004)

[8] Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research **4**, 237–285 (1996)

[9] Knowles, J.D., Watson, R.A., Corne, D.: Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, pp. 269–283. Springer-Verlag (2001)

[10] Lochtefeld, D.F., Ciarallo, F.W.: Deterministic Helper-Objective Sequences Applied to Job-Shop Scheduling. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 431–438. ACM (2010)

[11] da Silva, B.C., Basso, E.W., Bazzan, A.L.C., Engel, P.M.: Dealing with non-stationary environments using context detection. In: Proceedings of the 23rd International Conference on Machine Learning, ICML '06, pp. 217–224. ACM, New York, NY, USA (2006). DOI 10.1145/1143844.1143872. URL http://doi.acm.org/10.1145/1143844.1143872

[12] Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC Model-free Reinforcement Learning. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 881–888 (2006)

[13] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA (1998)