RUNTIME ANALYSIS OF RANDOM LOCAL SEARCH WITH REINFORCEMENT BASED SELECTION OF NON-STATIONARY AUXILIARY OBJECTIVES: INITIAL STUDY

Irina Petrova, Arina Buzdalova and Georgiy Korneev

ITMO University
Computer Technologies Department
49 Kronverkskiy ave., Saint Petersburg
197101, Russia
irenepetrova@yandex.com, abuzdalova@gmail.com, kgeorgiy@rain.ifmo.ru

Abstract: *It has been shown that single-objective optimization may be improved by introducing auxiliary objectives. In theoretical studies, auxiliary objectives are usually designed to be helpful. However, in practice, some of the auxiliary objectives may change their properties during optimization process. Such objectives may be efficient and obstructive at different steps of optimization. We analyse EA+RL method of objective selection. Precisely, Random Local Search with Q-learning using greedy exploration strategy is considered on the* XdivK *and* OneMax *problems. We theoretically show on the* OneMax *problem that EA+RL ignores the objective which is currently obstructive. We also show that EA+RL does not manage to ignore a non-stationary obstructive objective on the* XdivK *problem.*

Keywords: *runtime analysis, Markov chain, reinforcement learning, Q-learning*

1 Introduction

Consider single-objective optimization of a *target* objective performed by an evolutionary algorithm (EA). To enhance the efficiency of the target objective optimization, some *auxiliary* objectives may be introduced [8]. Auxiliary objectives may help to find the optimum of the *target* objective in less number of fitness function evaluations or find better solutions within the fixed number of evaluations.

There are several methods of using auxiliary objectives. In multiobjectivization, the objectives are optimized simultaneously [6]. Usually, in this method auxiliary objectives are specially developed by an expert to be efficient. Particularly, the auxiliary objectives are ensured to form Pareto front that includes the optimum of the target objective.

However, in practice, objectives may be generated automatically and change their properties during the optimization process [1]. Such objectives may be both efficient and obstructive at different stages of optimization. We call such objectives *non-stationary*.

In the case of non-stationary objectives, it may be more efficient to dynamically select the auxiliary objective which is the best at the current stage of optimization. The objectives may be selected randomly [5] or with reinforcement learning (RL) [7, 10]. The latter corresponds to the EA+RL method.

It has been shown experimentally that EA+RL efficiently selects non-stationary objectives [9]. EA+RL has been analysed theoretically as well [2–4]. However, none of the corresponding theoretical studies considered non-stationary objectives. At the same time, analysis may give useful insights on how to use EA+RL more efficiently. In the present paper we make initial steps towards theoretical analysis of EA+RL on non-stationary auxiliary objectives.

The rest of the paper is organized as follows. First, the EA+RL method and two model problems with non-stationary auxiliary objectives are described. Second, theoretical runtime analysis of EA+RL method on OneMax problem with non-stationary auxiliary objectives is made. Finally, the behaviour of the EA+RL method on theXdivK problem with non-stationary auxiliary objectives is analysed.

2 Preliminaries

In this section we describe the EA+RL method, which is analysed on two model problems in the rest of the paper. The model problems with non-stationary auxiliary objectives are defined as well.

2.1 EA+RL: selection of auxiliary objectives with reinforcement learning

Typical reinforcement learning scheme is as follows. An *agent* learns to choose efficient *actions* in each *state* of the *environment*. Efficiency of an action is measured by a *reward*. Environment returns some numerical reward for each action chosen by the agent. The goal of the agent is to maximize the reward in each state.

In the EA+RL method, an evolutionary algorithm is considered as the environment, the possible objectives correspond to actions. The agent selects an objective to be used in the next generation, the evolutionary algorithm selects individuals with high value of this objective for the next generation and returns the reward to the agent. The reward depends on the change of the target objective value. The higher the newly obtained target value, the higher the reward.

We consider the following implementation of EA+RL. For evolutionary algorithm, we use random local search (RLS). Q-learning algorithm is used for reinforcement learning. Therefore, we call the considered algorithm *RLS + Q-learning*. In RLS, there is one individual in a generation. The current state s is defined as the target objective value of the current individual. The reward is the difference of the target values in two subsequent generations. The pseudocode of RLS + Q-learning is presented in Algorithm 1.

Algorithm 1 RLS + Q-learning Algorithm

Individual $y \leftarrow$ a random bit string
Define set H of auxiliary objectives and target objective
$Q(s,h) \leftarrow 0$ for each state s and objective $h \in H$
while (Optimum of the target objective t is not found) **do**
  Current state $s \leftarrow t(y)$
  Individual $y' \leftarrow$ mutate y (flip a random bit)
  Objective h: $Q(s,h) = \max_{h' \in H} Q(s,h')$ {If Q-values are equal, objectives are selected equiprobably}
  if $h(y') \geq h(y)$ **then**
    $y \leftarrow y'$
  end if
  New state $s' \leftarrow t(y)$
  Reward $r \leftarrow s' - s$
  $Q(s,h) \leftarrow Q(s,h) + \alpha(r + \max_{h' \in H} Q(s',h') - Q(s,h))$
end while

2.2 OneMax with non-stationary objectives

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 objective OneMax is equal to x and the objective ZeroMax is equal to $n - x$.

We define the target objective t as OneMax. The auxiliary objectives can be both OneMax or ZeroMax at different stages of optimization. More precisely, consider the auxiliary objectives h_1 and h_2 described below in (1).

$$h_1(x) = \begin{cases} \textsc{OneMax} & \text{if } x \leq p \\ \textsc{ZeroMax} & \text{if } p < x \leq n \end{cases} \qquad h_2(x) = \begin{cases} \textsc{ZeroMax} & \text{if } x \leq p \\ \textsc{OneMax} & \text{if } p < x \leq n \end{cases} \tag{1}$$

The parameter p is called a *switch point*. The h_2 objective is obstructive when $x \in [0,p]$ while the h_2 objective is obstructive when $x \in (p,n]$.

2.3 XdivK with non-stationary objectives

Consider XdivK problem with auxiliary objectives defined in (1). The target objective is $t(x) = \lfloor \frac{x}{k} \rfloor$, where x is the number of ones, k is constant, $k < n$ and k divides n. The objective which is equal to OneMax at the current interval of x values is efficient, while the objective equal to ZeroMax is obstructive.

The difference between this problem and the OneMax problem with non-stationary objectives is that auxiliary objective can be efficient or inefficient. Using the objective which is equal to OneMax 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.

3 Expected running time of RLS + Q-learning on OneMax with non-stationary objectives

To compute the expected running time of the algorithm, we construct the Markov chain that represents the corresponding optimization process [4]. Recall that the RL state is determined by the number of one bits in the

current individual. For every RL state, we define *two* Markov chain states: one for the case the corresponding RL state has not been visited previously, and one for the opposite case.

In the first subsection, we describe the Markov chain for the OneMax problem with non-stationary objectives. In the second subsection, we theoretically analyse the expected running time of the RLS + Q-learning algorithm on this problem.

3.1 Markov chain

We denote the efficient objective which is equal to OneMax in the state s as $g(s)$. The obstructive objective which is equal to ZeroMax in the state s is denoted as $b(s)$ (2).

$$g(s) = \begin{cases} h_1 & \text{if } s \leq p \\ h_2 & \text{if } s > p \end{cases} \qquad b(s) = \begin{cases} h_2 & \text{if } s \leq p \\ h_1 & \text{if } s > p \end{cases} \tag{2}$$

The Markov chain which represents the optimization process is shown in Fig. 1. The labels on the transitions have the format F,M,P where F are fitness functions that can be chosen for this transition, M are the corresponding mutation cases, P is the probability of transition. For example, for the transition from the state A_i to the state A_{i+1} we compute the product of two probabilities: the probability to select t and the probability of flipping a 0-bit. In state A_i agent has no experience, so according to Algorithm 1, the objectives are selected equiprobably with probability $\frac{1}{3}$. Probability of flipping a 0-bit in the state A_i is equal to $\frac{n-i}{n}$. So the resulting product is equal to $\frac{n-i}{3n}$. We also compute the product of probability to select g and the probability of flipping a 0-bit, which is also equal to $\frac{n-i}{3n}$. So the resulting probability for the transition from the state A_i to the state A_{i+1} is equal to $\frac{2(n-i)}{3n}$. Further we describe each Markov state and explain why the transitions has such labels.

The Markov state A_i corresponds to RL state with the target objective value $t = i$, the Markov state A_{i+1} corresponds to RL state with $t = i + 1$. The RL agent visits these states for the first time and thus selects objectives equiprobably. The Markov state B_{i-1} corresponds to the RL state with $t = i - 1$. The algorithm reaches this Markov state when the agent selects the obstructive objective $b(i)$ in the RL state i. When this happens, the agent obtains a negative reward, so the agent will not select the objective $b(i)$ in the RL state i anymore. The Markov state A_i is reached from the RL state $i - 1$. It can be done by selection of one of the objectives t and $g(i - 1)$. The agent obtains positive reward for this objective. So only this objective will be selected in the state B_{i-1}. The Markov state C_i corresponds to RL state i. In this state agent has already learned that the objective $b(i)$ is not efficient. So the objective $b(i)$ will not be selected in C_i.

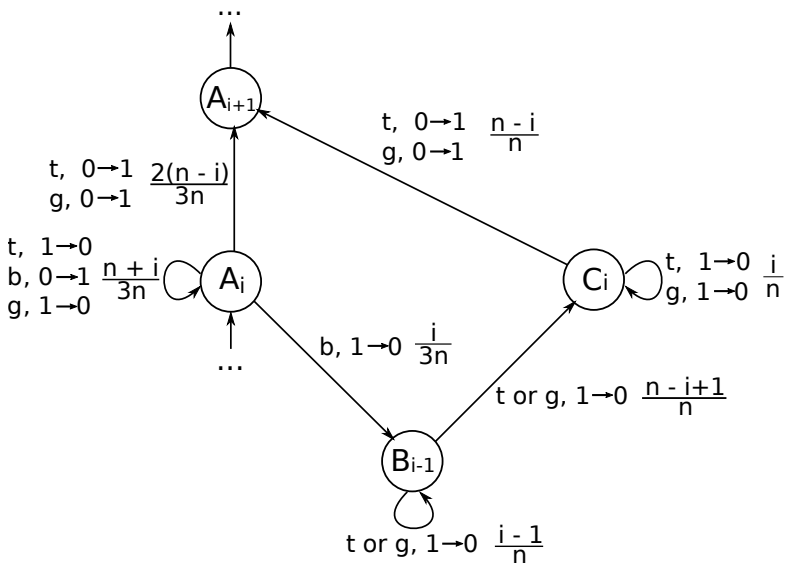

Figure 1: Markov chain for the analysis of the expected running time for OneMax

3.2 Expected running time

Expected running time of RLS + Q-learning on OneMax with non-stationary objectives is equal to the number of fitness function evaluations needed to get from the state A_0 to the state A_n. Each transition in the Markov chain corresponds to one fitness function evaluation of the mutated individual. So the expected running time

of RLS + Q-learning is equal to the number of transitions in the Markov chain. Denote the expected running time of the algorithm as $T(n)$.

$$T(n) = \sum_{i=0}^{n-1} E(A_i \to A_{i+1}), \tag{3}$$

where $E(A_i \to A_{i+1})$ is expected value of transitions needed to reach the state A_{i+1} from state A_i.

The expected value of transitions needed to reach the state A_{i+1} from state A_i is evaluated as:

$$E(A_i \to A_{i+1}) = \frac{2(n-i)}{3n} \cdot 1 + \frac{i+n}{3n} \cdot (1 + E(A_i \to A_{i+1})) + \frac{i}{3n} \cdot (1 + E(B_{i-1} \to A_{i+1})) \tag{4}$$

Consider the expected number of transitions needed to reach the state A_{i+1} from state B_{i-1}:

$$E(B_{i-1} \to A_{i+1}) = E(B_{i-1} \to C_i) + E(C_i \to A_{i+1}) \tag{5}$$

First, the expected number of transitions needed to get from B_{i-1} to C_i is evaluated:

$$E(B_{i-1} \to C_i) = \frac{n-i+1}{n} \cdot 1 + \frac{i-1}{n} \cdot (1 + E(B_{i-1} \to C_i))$$
$$E(B_{i-1} \to C_i) = \frac{n}{n-i+1} \tag{6}$$

Second, the expected number of transitions needed to get from C_i to A_{i+1} is evaluated:

$$E(C_i \to A_{i+1}) = \frac{n-1}{n} \cdot 1 + \frac{i}{n} \cdot (1 + E(C_i \to A_{i+1}))$$
$$E(C_i \to A_{i+1}) = \frac{n}{n-i} \tag{7}$$

Therefore, from (4), (5), (6) and (7):

$$E(A_i \to A_{i+1}) = 1 + \frac{i+n}{3n} \cdot E(A_i \to A_{i+1}) + \frac{i}{3n} \cdot \left(\frac{n}{n-i+1} + \frac{n}{n-i} \right)$$
$$E(A_i \to A_{i+1}) = \frac{3n}{2n-i} + \frac{3n}{2n-i} \cdot \left(\frac{i}{3(n-i+1)} + \frac{i}{3(n-i)} \right) =$$
$$= 2 + \frac{2}{3} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} \right) + \frac{2i-n}{3(2n-i)} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} + 3 \right) \tag{8}$$

The total expected number of transitions to get from A_0 to A_n is equal to

$$T(n) = \sum_{i=0}^{n-1} E(A_i \to A_{i+1}) = \sum_{i=0}^{n-1} 2 + \frac{2}{3} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} \right) +$$
$$\sum_{i=0}^{n-1} \frac{2i-n}{3(2n-i)} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} + 3 \right) \tag{9}$$

Let us estimate the obtained result. The expected number of fitness function evaluations needed to solve OneMax problem using Random Local Search is the following:

$$T_{RLS} = \sum_{i=0}^{n-1} \left(1 + \frac{i}{n-i} \right)$$
$$T_{RLS} = \Theta(n \log n) \tag{10}$$

Let us analyse the first part of the sum in (9).

$$T_1 = \sum_{i=0}^{n-1} 2 + \frac{2}{3} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} \right)$$
$$1 + \frac{i}{n-i} < 2 + \frac{2}{3} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} \right) < 2 \left(1 + \frac{i}{n-i} \right) \tag{11}$$

Therefore, $T_1 = \Theta(n \log n)$.

Let us analyse the second part of the sum in (9).

$$T_2 = \sum_{i=0}^{n-1} \frac{2i-n}{3(2n-i)} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} + 3 \right)$$

$$\frac{1}{6}\left(1+\frac{i}{n-i}\right) < \frac{2i-n}{3(2n-i)} \cdot \left(\frac{i}{n-i+1} + \frac{i}{n-i} + 3 \right) < \frac{i}{n-i+1} + \frac{i}{n-i} + 3 < 3\left(1+\frac{i}{n-i}\right) \tag{12}$$

Therefore, $T_2 = \Theta(n \log n)$. So $T(n) = T_1 + T_2 = \Theta(n \log n)$

So it is finally proved that the expected running time of RLS + Q-learning on OneMax problem with EA+RL using two auxiliary objectives, which are obstructive on some stages of optimization process, is $\Theta(n \log n)$. Note that expected running time of OneMax problem without obstructive objectives is $\Theta(n \log n)$. Hence, the obstructive objectives do not worsen the asymptotic of the optimization algorithm, because Q-learning manages to ignore an obstructive objective.

4 Expected Running time of RLS on XdivK with non-stationary objectives

In the XdivK problem, a Markov state is the number of 1-bits in the individual. Unlike for the OneMax problem, for this problem RL state includes Markov states with different number of 1-bits. As it is shown in the Markov chain for the XdivK problem (see Fig. 2), one RL state includes k Markov states.

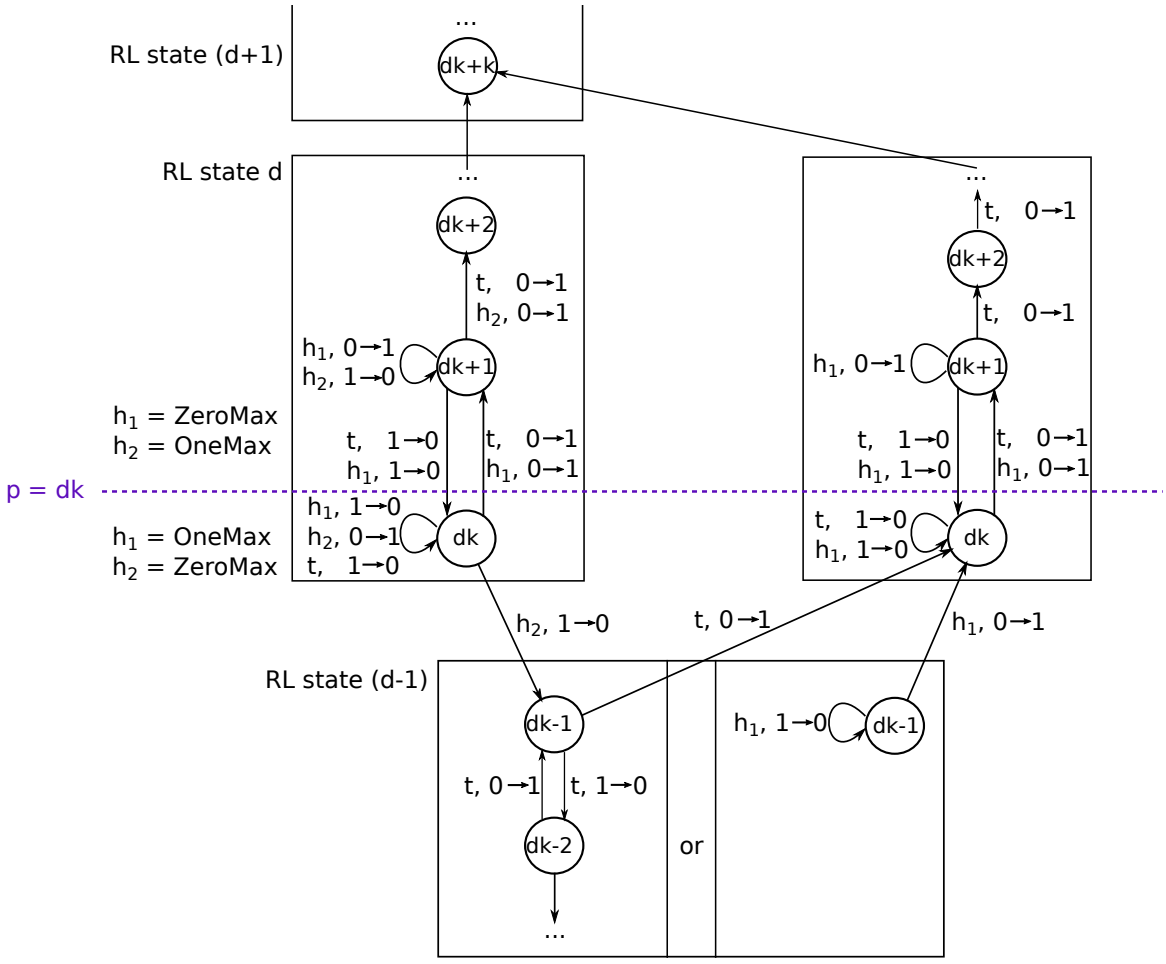

Figure 2: Markov chain for the analysis of the expected running time for XdivK

In the first subsection we describe the Markov chain for the XdivK problem with non-stationary objectives. Also we explain why the expected running time of XdivK problem with non-stationary objectives is higher

than the running time of XdivK with OneMax [2] and XdivK without auxiliary objectives. In the second section we confirm this statement experimentally.

4.1 Markov chain

Consider XdivK problem with non-stationary objectives. Let $x = dk$, where d is a constant and the agent has no experience in RL state d. If the agent selects the objective which is equal to ZeroMax, and mutation operator inverts 1-bit, the new individual has $dk-1$ 1-bits and it is better than the current individual according to ZeroMax. So the new individual will be selected for the next generation. However, the target objective value of the new individual is less than the target objective value of the current individual. Therefore, the agent achieves a negative reward in the RL state d and moves to the RL state $d-1$.

Denote the objective which is equal to OneMax when $x = dk-1$ as g. In the state $d-1$ agent may select either t or g. The objective t is selected if the agent has previously moved from the state d to the state $d-1$ using this objective. Analogously, g is selected if it was used when moving between these states.

After the agent has moved to the state d from the state $d-1$, it selects one of two objectives, the target objective t or the objective, which is equal to OneMax when $x = dk$. Further success of the agent depends on the position of the switch point. If the switch point p lies in the interval $[dk, (d+1)k-1]$, the number of ones when x lies in the interval $[p, (d+1)k-1]$ can be increased only if the target objective is selected and 0-bit is mutated. So the worst case is when $p = dk$.

It follows from the above description that if the algorithm goes down to the state $d-1$, it needs a lot of steps to reach the state $d+1$. Probability of this transition from Markov state dk is $\frac{dk}{3n}$. So this probability grows with the target objective value. Thus, if the switch point splits the RL state $n/k-1$, the expected running time of the algorithm is maximal.

It is important to mention the XdivK problem with OneMax and the XdivK problem without objectives analysed in [2]. The main difference in behaviour of RLS + Q-learning on these problems and on the currently analysed XdivK problem with non-stationary objectives is as follows. In the current problem, selection of the auxiliary objective which is equal to ZeroMax may cause decreasing of the target objective value. In XdivK with OneMax if the target objective value was increased from d to $d+1$, the agent obtained a reward and moved from the state d to $d+1$, the algorithm will never return to state d. In the XdivK problem without objectives, the best target objective value can not be decreased because RLS preserves the individual with the best target objective value.

4.2 Experimental analysis

In Table 1, we present the number of fitness function evaluations needed to reach the optimum of the XdivK problem without auxiliary objectives, XdivK with only OneMax auxiliary objective and XdivK with two non-stationary auxiliary objectives. First two columns correspond to values of n and k. Next two columns

Table 1: The number of fitness function evaluations on XdivK problem with auxiliary objectives

n	k	XdivK	XdivK+OneMax	XdivK + two objectives
40	2	$1.19 \cdot 10^3$	$6.81 \cdot 10^2$	$3.72 \cdot 10^3$
48	2	$1.70 \cdot 10^3$	$9.56 \cdot 10^2$	$5.23 \cdot 10^3$
56	2	$2.30 \cdot 10^3$	$1.28 \cdot 10^3$	$6.81 \cdot 10^3$
64	2	$2.98 \cdot 10^3$	$1.64 \cdot 10^3$	$8.91 \cdot 10^3$
72	2	$3.76 \cdot 10^3$	$2.05 \cdot 10^3$	$1.11 \cdot 10^4$
80	2	$4.62 \cdot 10^3$	$2.51 \cdot 10^3$	$1.41 \cdot 10^4$
60	3	$3.94 \cdot 10^4$	$1.08 \cdot 10^4$	$2.95 \cdot 10^5$
72	3	$6.79 \cdot 10^4$	$1.83 \cdot 10^4$	$4.98 \cdot 10^5$
84	3	$1.08 \cdot 10^5$	$2.86 \cdot 10^4$	$7.81 \cdot 10^5$
96	3	$1.60 \cdot 10^5$	$4.23 \cdot 10^4$	$1.05 \cdot 10^6$
108	3	$2.28 \cdot 10^5$	$5.97 \cdot 10^4$	$1.63 \cdot 10^6$
120	3	$3.12 \cdot 10^5$	$8.14 \cdot 10^4$	$2.37 \cdot 10^6$
80	4	$1.72 \cdot 10^6$	$2.30 \cdot 10^5$	$2.64 \cdot 10^7$
96	4	$3.57 \cdot 10^6$	$4.72 \cdot 10^5$	$5.67 \cdot 10^7$
112	4	$6.61 \cdot 10^6$	$8.68 \cdot 10^5$	$1.01 \cdot 10^8$
128	4	$1.13 \cdot 10^7$	$1.47 \cdot 10^6$	$1.72 \cdot 10^8$
144	4	$1.81 \cdot 10^7$	$2.35 \cdot 10^6$	$2.83 \cdot 10^8$
160	4	$2.76 \cdot 10^7$	$3.57 \cdot 10^6$	$3.96 \cdot 10^8$

contain theoretically calculated number of fitness function evaluations on XdivK without auxiliary objectives and on XdivK with OneMax correspondingly. These results were taken from the paper [2]. The next column contains number of fitness function evaluations on XdivK with OneMax averaged by 1000 runs. For the Q-learning algorithm, learning rate was set to $\alpha = 0.5$, discount factor was $\gamma = 0.5$. We considered the worst case when switch point p is equal to $n - k$.

We can see that RLS + Q-learning on XdivK with two non-stationary objectives needs much more fitness function evaluations than RLS + Q-learning on XdivK with efficient auxiliary objective. Although at each step of optimization process one of two non-stationary objectives is efficient, RLS on XdivK without auxiliary objectives performs better than RLS + Q-learning on XdivK with two non-stationary objectives.

5 Conclusion

We analysed the RLS + Q-learning method with two non-stationary auxiliary objectives on the OneMax and XdivK problems. We proved that the expected running time of RLS + Q-learning on the OneMax problem with objectives which are obstructive on some stages of optimization is $\Theta(n \log n)$. The asymptotic of RLS on OneMax without auxiliary objectives is the same. This means that in OneMax problem EA+RL ignores the objective which is currently inefficient.

We also analysed how RLS + Q-learning works on the XdivK problem with non-stationary objectives. On each step of optimization process one of the auxiliary objective is more efficient than the target objective. However, it was shown that in the case of selection of objective which is obstructive on the current step of optimization, the best found solution may be lost and the algorithm needs a lot of steps to find a good solution again. We experimentally showed that the running time of RLS + Q-learning on XdivK problem with non-stationary objectives is much greater than the running time of RLS on XdivK without auxiliary objectives. So the RLS + Q-learning does not manage to reflect on changing of properties of auxiliary objectives.

There are some ideas of how to deal with non-stationary auxiliary objectives. The first idea is that efficiency of using non-stationary objectives in EA+RL depends on how the states and the reward are defined. Q-learning on XdivK problem with non-stationary objectives does not recognize increasing of the number of ones in the case when the target objective value stayed unchanged. This is due to the fact that the reward and the state depend only on the target objective value. Therefore, reward and state values should depend not only on the target objective value but on the auxiliary objective values too. The second idea is that Q-learning should additionally monitor changes of auxiliary objective properties. For example, the agent may control the number of consequent iterations in which the number of 1-bits stayed unchanged. Thereby, the obtained theoretical results give a direction for the future design of objective selection algorithms.

Acknowledgements: Irina Petrova and Arina Buzdalova were supported by RFBR according to the research project No. 16-31-00380 mol_a. Georgiy Korneev was supported by the Government of Russian Federation, Grant 074-U01.

References

[1] Buzdalov, M., Buzdalova, A.: Adaptive selection of helper-objectives for test case generation. In: 2013 IEEE Congress on Evolutionary Computation, vol. 1, pp. 2245–2250 (2013)

[2] Buzdalov, M., Buzdalova, A.: OneMax helps optimizing XdivK: Theoretical runtime analysis for RLS and EA+RL. In: Proceedings of Genetic and Evolutionary Computation Conference Companion, pp. 201–202. ACM (2014)

[3] Buzdalov, M., Buzdalova, A.: Analysis of Q-Learning with random exploration for selection of auxiliary objectives in random local search. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1776–1783 (2015)

[4] Buzdalov, M., Buzdalova, A., Shalyto, A.: A first step towards the runtime analysis of evolutionary algorithm adjusted with reinforcement learning. In: Proceedings of the International Conference on Machine Learning and Applications, vol. 1, pp. 203–208. IEEE Computer Society (2013)

[5] 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)

[6] 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)

[7] Petrova, I., Buzdalova, A., Buzdalov, M.: Selection of extra objectives using reinforcement learning in non-stationary environment: Initial explorations. In: Proceedings of 20th International Conference on Soft Computing MENDEL 2014, pp. 58–63. Czech Republic (2014)

[8] Segura, C., Coello, C.A.C., Miranda, G., Léon, C.: Using multi-objective evolutionary algorithms for single-objective optimization. 4OR **3**(11), 201–228 (2013)
[9] Silva, B.C.D., 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, pp. 217–224. ACM Press (2006)
[10] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA (1998)