Reinforcement Learning Based Dynamic Selection of Auxiliary Objectives with Preservation of the Best Found Solution

Irina Petrova
ITMO University
49 Kronverkskiy ave.
Saint-Petersburg, Russia 197101
irenepetrova@yandex.com

Arina Buzdalova
ITMO University
49 Kronverkskiy ave.
Saint-Petersburg, Russia 197101
abuzdalova@gmail.com

ABSTRACT

Efficiency of single-objective optimization can be improved by introducing auxiliary objectives. In practice, they may be efficient on some optimization stages but obstructive on others. We propose to modify the EA+RL method which dynamically selects objectives using reinforcement learning. The proposed modification prevents from losing the best found solution.

We analysed the proposed modification and compared it with the EA+RL method and Random Local Search on XdivK, Generalized OneMax and LeadingOnes problems. The proposed modification outperforms the EA+RL method on all problem instances. It outperforms the single objective approach on most problem instances as well. We also present theoretical analysis of the proposed modification on the XdivK problem.

CCS CONCEPTS

•**Computing methodologies → Control methods;** Reinforcement learning; •**Theory of computation** → *Bio-inspired optimization;*

KEYWORDS

multiobjectivisation, reinforcement learning, Markov chain

ACM Reference format:
Irina Petrova and Arina Buzdalova. 2017. Reinforcement Learning Based Dynamic Selection of Auxiliary Objectives with Preservation of the Best Found Solution. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3067695.3082499

1 INTRODUCTION

An evolutionary algorithm (EA) can reach the optimum of the target objective in less number of fitness evaluations using *auxiliary* objectives [1, 6, 8, 10, 12]. In practice, objectives can be generated automatically and may be efficient on some optimization stages but obstructive on others [9]. We call such objectives *non-stationary*. One of the approaches to deal with such objectives is dynamic selection of the best objective at the current stage of optimization. The objectives may be selected randomly [7]. Another method is EA+RL which uses reinforcement learning (RL) [4, 13].

In reinforcement learning (RL) an *agent* applies an *action* to an *environment*. Then the environment returns a numerical *reward* and a representation of its *state* and the process repeats. The goal of the agent is to maximize the total reward [13]. In the EA+RL method, EA is treated as an environment. To make an action means to select an objective to be optimized in the current generation of EA. The reward is equal to difference of the best target objective value in two consecutive generations.

It was theoretically shown for a number of optimization problems that EA+RL efficiently works with stationary objectives [2, 3]. However, theoretical analysis of EA+RL with non-stationary objectives showed that EA+RL does not ignore obstructive objectives on the XdivK and the Generalized OneMax problems [5, 11]. Selection of an inefficient objective causes loss of the best found solution and the algorithm needs a lot of steps to find a good solution again.

We propose a modification of EA+RL which preserves the best found solution and analyse it theoretically on XdivK and experimentally on XdivK, Generalized OneMax and LeadingOnes. The full version of the paper is available at arXiv[1].

2 MODIFIED EA+RL

In the EA+RL method, if the newly generated individual is better than the existing one according to the selected objective, the new individual is accepted. However, if the selected objective is obstructive, the new individual may be worse than the existing individual in terms of the target objective. In this case EA loses the individual with the best target objective value.

In the modified EA+RL, if the newly generated individual is better than the existing one according to the selected objective, but is worse according to the target objective, the new individual is rejected. As in the recent theoretical works, we use RLS as optimization algorithm and apply Q-learning to select objectives [11]. Population consists of a single individual. Individuals are represented as bit strings, the flip-one-bit mutation is used. The modified EA+RL is presented in Algorithm 1. In Q-learning, the efficiency of selecting an objective h in a state s is measured by $Q(s, h)$, which is updated dynamically as shown in line 10 of the pseudocode, where α is the learning rate and γ is the discount factor.

We consider two versions of modified EA+RL with different ways of reward calculation. In both these versions a reward is equal to the difference of the target value in two consecutive generations, except the following case. If the new individual is better than the

[1] http://arxiv.org/abs/1704.07187

Algorithm 1 Modified EA+RL

```
1:  Individual $y \leftarrow$ random bit string
2:  Construct set $H$ of auxiliary objectives and target objective
3:  $Q(s, h) \leftarrow 0$ for each state $s$ and objective $h \in H$
4:  while (Optimum of target objective $t$ is not found) do
5:      Calculate current state $s$
6:      Individual $y' \leftarrow$ mutate $y$ (flip random bit)
7:      Objective $h$: $Q(s, h) = \max_{h' \in H} Q(s, h')$   ▷ If Q-values are
        equal, objectives are selected equiprobably
8:      if $h(y') \geq h(y)$ and $t(y') \geq t(y)$ then $y \leftarrow y'$
9:      Calculate state $s'$ and reward $r$
10:     $Q(s, h) \leftarrow Q(s, h) + \alpha(r + \max_{h' \in H} Q(s', h') - Q(s, h))$
```

current one according to the selected objective, but its target value is lower, the new individual is rejected. In the first version of modified EA+RL the agent achieves zero reward because the individual is not changed. In the second version the agent achieves negative reward, as if the new individual was accepted. Thereby, in the first version the agent does not learn if the action was inefficient and learns only if the target objective was increased. We call this algorithm *modification of EA+RL without learning on mistakes* (EA+RLnM). In the second version the agent learns in both cases: when the action was efficient and inefficient. We call this algorithm *modification of EA+RL with learning on mistakes* (EA+RLM).

In the existing theoretical works on EA+RL, an RL state is defined as the target objective value [2, 5]. We denote it as *target* state, which we use in this work. However, if the individual with the best target value is preserved, the algorithm will never return to the state where it achieved a positive reward. So the agent never knows which objective is helpful. It only can learn that an objective is obstructive if the agent achieved a negative reward for it. Therefore, we also consider the *single* state. This state is the same during the optimization process so the agent can learn which objective is good.

3 MODEL PROBLEMS

We consider three model problems: Generalized OneMax, XdivK and LeadingOnes, which were used in studies of EA+RL [3, 5, 11]. In all considered problems, an individual is a bit string of length n. The target objective of Generalized OneMax, denoted as OM_d, is calculated as the number of bits in an individual that matches a given bit mask. The bit mask has d 0-bits and $n - d$ 1-bits. If $d = 0$ or $d = n$ the OM_d problem is called OneMax or ZeroMax respectively. The XdivK is calculated as $\lfloor \frac{x}{k} \rfloor$, where x is the number of 1-bits, k is a constant, k divides n. The target objective of LeadingOnes is equal to the length of the maximal prefix of 1-bits.

We used two non-stationary auxiliary objectives defined in (1) for all the considered problems. These auxiliary objectives can be OneMax (OM) or ZeroMax (ZM) at different stages of optimization. They switch at a *switch point* defined by the parameter p.

$$h_1(x) = \begin{cases} \text{OM}, x \leq p \\ \text{ZM}, p < x \leq n \end{cases} \qquad h_2(x) = \begin{cases} \text{ZM}, x \leq p \\ \text{OM}, p < x \leq n \end{cases} \tag{1}$$

In LeadingOnes and XdivK problems the objective which is equal to OneMax at the current stage of optimization is helpful and ZeroMax is obstructive [3, 11]. In the OM_d problem, both objectives may be obstructive or neutral [5].

4 THEORETICAL ANALYSIS

Previously, it was shown that the EA+RL method gets stuck in local optima on XdivK with non-stationary objectives [11]. Below we present theoretical runtime analysis of the proposed EA+RLnM on this problem. The target state is used. To compute the expected runtime of the algorithm, we construct the Markov chain that represents the corresponding optimization process [2, 11]. We distinguish between RL states determined by the target objective value and states of the Markov chain, which we call Markov states. Markov states correspond to the number of 1-bits in an individual. Therefore, an RL state includes k Markov states with different number of 1-bits. To analyse the runtime of EA+RLnM, we also need to construct Markov chain for RLS without auxiliary objectives.

The Markov chains for the XdivK problem are shown in the Fig. 1. The labels on transitions have the format F, M, where F is a fitness function that can be chosen for this transition, M is the corresponding effect of mutation.

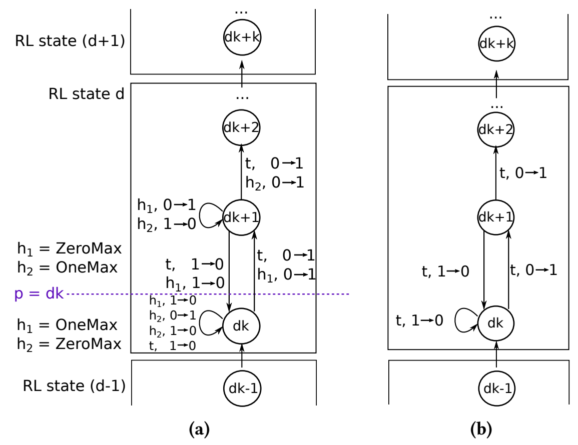

Figure 1: Markov chains on XdivK: EA+RLnM (a), RLS (b)

The expected runtime of EA+RLnM is equal to the number of fitness evaluations needed to get from the Markov state 0 to n. Each transition in the chain corresponds to one fitness evaluation. So the expected runtime is equal to the number of transitions. Denote it as $T(n) = \sum_{i=0}^{n-1} E(i \longrightarrow i+1)$, where $E(i \longrightarrow i+1)$ is the expected number of transitions needed to reach the state $i+1$ from i.

Consider two cases for the state i. The first one is $i = dk$, where d is a constant. The expected number of transitions needed to reach the state $dk + 1$ from dk is evaluated using probabilities of mutation and objective selection as $z_{dk} = \frac{2}{3} \cdot \frac{(n-dk)}{n} \cdot 1 + (\frac{2}{3} \cdot \frac{dk}{n} + \frac{1}{3}) \cdot (1 + z_{dk})$. From this we obtain that $z_{dk} = \frac{3n}{2(n-dk)}$.

The second case is $i = dk + t$, where $1 \leq t \leq k - 1$. The expected number of transitions needed to reach the state $dk + t + 1$ from the state $dk + t$ is evaluated as $z_{dk+t} = \frac{2(n-dk-t)}{3n} + \frac{2(dk+t)}{3n} \cdot (1 + z_{dk+t-1} + z_{dk+t}) + (\frac{dk+t}{3n} + \frac{n-dk-t}{3n}) \cdot (1 + z_{dk+t})$. From this we obtain that $z_{dk+t} = z_{dk+t-1} \cdot \frac{dk+t}{n-dk-t} + \frac{3n}{2(n-dk-t)}$.

To estimate the efficiency of EA+RLnM, we calculate the expected runtime of RLS without auxiliary objectives. The total runtime is calculated as $T(n)$ from the EA+RLnM analysis. Analogically to EA+RLnM, we consider two cases: $i = dk$ and $i = dk + t$. The expected number of transitions needed to reach the state $dk + 1$

from the state dk is evaluated as $a_{dk} = \frac{(n-dk)}{n} \cdot 1 + \frac{dk}{n} \cdot (1 + a_{dk})$. From this we obtain that $a_{dk} = \frac{n}{(n-dk)}$. The expected value of transitions needed to reach the state $dk + t + 1$ from the state $dk + t$ is evaluated as $a_{dk+t} = \frac{(n-dk-t)}{n} \cdot 1 + \frac{dk+t}{n} \cdot (1 + a_{dk+t-1} + a_{dk+t})$. From this we obtain that $a_{dk+t} = a_{dk+t-1} \cdot \frac{dk+t}{n-dk-t} + \frac{n}{(n-dk-t)}$.

From the equations for z_{dk} and a_{dk} we obtain that $z_{dk} = \frac{3}{2} a_{dk}$. From the equations for $T(n)$, z_{dk+t}, a_{dk+t} using mathematical induction we obtain that the runtime of EA+RLnM for XdivK with non-stationary objectives is 1.5 times greater than the runtime of RLS. Therefore, the EA+RLnM has asymptotically the same runtime as RLS, which is bounded by $\Omega(n^k)$ and $O(n^{k+1})$ [2]. Recall that asymptotic of EA+RL runtime is worse [11]. So EA+RLnM better deals with non-stationary objectives. Also from the Markov chain we can see that transitions and, as a consequence, performance of EA+RLnM does not depend on the number and positions of switch points.

5 EXPERIMENT DESCRIPTION AND RESULTS

We empirically analysed EA+RLM, EA+RLnM, EA+RL and RLS on OM_d, XdivK and LeadingOnes. The non-stationary objectives described in (1) were used for all the problems. For XdivK we analysed two cases of the switch point position. The first case is the worst case [11], when the switch point is in the end of optimization process, $p = n - k + 1$. In the second case, the switch point is in the middle of optimization process, $p = n/2$. For each algorithm we analysed two state definitions: the single state (ss) and the target state (ts). Also we studied applying of ε-greedy strategy when the agent selects the objective with the maximum expected reward with probability $1 - \varepsilon$ and with probability ε the agent selects a random objective. We used the Q-learning algorithm with $\alpha = 0.5$ and $\gamma = 0.5$ [11]. The ε-greedy strategy was used with $\varepsilon = 0.1$. The obtained numbers of fitness evaluations needed to reach the optimum averaged by 1000 runs are presented in Table 1. The best and the second best results are colored dark grey and grey. None of the algorithms reached the optimum using the single state and $\varepsilon = 0$, so these results are not presented. Whenever the optimum has not been reached within 10^9 iterations, the corresponding result is marked as ∞.

We can see from Table 1 that EA+RLM using the single state and $\varepsilon = 0.1$ is the most efficient algorithm on OM_d, LeadingOnes and XdivK with switch point in the middle. On two latter problems EA+RLM achieves better results than RLS and on OM_d the results are almost the same. On XdivK with switch point in the end the best results are achieved using RLS and EA+RLnM. For each problem, we picked the best configuration of each algorithm and compared them by Mann-Whitney test with Bonferroni correction. The algorithms were statistically distinguishable with p-value less than 0.05.

6 DISCUSSION OF RESULTS

We can see from the results that EA+RLM outperforms EA+RLnM on LeadingOnes and XdivK with switch point in the middle. Therefore, learning on mistakes is useful because it allows the agent to remember that the objective is obstructive and not to select it subsequently. However, on the XdivK problem with switch point in the end, the best results are achieved using EA+RLnM. Below we explain why sometimes it is better not to learn on mistakes.

In EA+RLM, if the agent obtained a negative reward for some objective, it will take a lot of steps to re-learn when this objective becomes efficient. The agent re-learns when a sufficient amount of positive reward is obtained using this objective. The agent obtains a positive reward when the target objective is increased. To make things worse, in the XdivK problem it is not always possible to increase the target objective in one iteration of the algorithm. Let the number of 1-bits be dk, so the RL state is d. To move to the state $d + 1$, the algorithm needs to mutate k 0-bits. Let switch point p be equal to $dk + l$, where $0 < l < k$. Then if the number of 1-bits is greater than $dk + l$, the algorithm can increase the number of 1-bits only if 0-bit is mutated and the target objective is selected. However, whatever bit is mutated and whatever objective is selected, the target objective value stays unchanged until an individual with $dk + k$ 1-bits is obtained. So the agent does not recognize if its action is good or bad because the reward is equal to zero. Therefore, to increase the target objective value algorithm needs a lot of steps.

Consider the worst switch point position. If the switch point is in the end of optimization, probability to mutate a 0-bit and select the target objective at the same time is low. The worst case is when the switch point is equal to $n - k + 1$, because the agent needs to select the target objective and increase the number of 1-bits during $k - 1$ iterations. EA+RLnM does not have this drawback: as it can be seen from Section 4, it does not depend on the number of switch points and their positions.

Consider influence of the state definition. In the single state the rewards are accumulated during the whole optimization process, while in the target state if the agent obtains a positive reward it moves to a new state and loses the previous experience. So the single state (unlike the target one) allows to learn which objective is helpful. Thus the single state is better than the target state for EA+RLM. However, in the single state it is more difficult to re-learn that the objective which was helpful became obstructive. Therefore, when the single state with $\varepsilon = 0$ is used, none of the algorithms reaches the optimum. Also EA+RL does not reach the optimum using the single state. For EA+RLnM single state is also inefficient because the agent does not learn if the objective became obstructive and the only way to stop selection of this objective is to re-learn. On the contrast, the target state allows agent to move to the new state where it has no experience.

7 CONCLUSION

We proposed a modification of the EA+RL method which preserves the best found solution. We considered two versions of the proposed modification called EA+RLnM and EA+RLM. In EA+RLnM, the RL agent learns only when it finds a better solution. In EA+RLM, the RL agent also learns when it obtains an inefficient solution.

We considered two auxiliary objectives which change their efficiency at a switch point. We experimentally analysed the two proposed modifications and the EA+RL method on Generalized OneMax (OM_d), LeadingOnes, XdivK with switch point in the middle of optimization and XdivK with switch point in the end. Two RL states were considered: the single state and the target state.

Both proposed modifications reached the optimum on OM_d and LeadingOnes unlike the EA+RL method did. EA+RLM with

Table 1: Averaged number of fitness evaluations needed to reach the global optimum

		EA+RLM			EA+RLnM			EA+RL		
Parameters	RLS	ss, $\varepsilon = 0.1$	ts, $\varepsilon = 0$	ts, $\varepsilon = 0.1$	ss, $\varepsilon = 0.1$	ts, $\varepsilon = 0$	ts, $\varepsilon = 0.1$	ss, $\varepsilon = 0.1$	ts, $\varepsilon = 0$	ts, $\varepsilon = 0.1$
n	LeadingOnes									
141	$1.00 \cdot 10^4$	$4.61 \cdot 10^3$	$7.20 \cdot 10^3$	$7.80 \cdot 10^3$	$1.36 \cdot 10^4$	$1.49 \cdot 10^4$	$1.49 \cdot 10^4$	∞	∞	∞
151	$1.13 \cdot 10^4$	$5.08 \cdot 10^3$	$8.33 \cdot 10^3$	$8.90 \cdot 10^3$	$1.57 \cdot 10^4$	$1.72 \cdot 10^4$	$1.72 \cdot 10^4$	∞	∞	∞
161	$1.30 \cdot 10^4$	$5.44 \cdot 10^3$	$9.39 \cdot 10^3$	$1.01 \cdot 10^4$	$1.81 \cdot 10^4$	$1.94 \cdot 10^4$	$1.96 \cdot 10^4$	∞	∞	∞
171	$1.45 \cdot 10^4$	$6.04 \cdot 10^3$	$1.06 \cdot 10^4$	$1.13 \cdot 10^4$	$2.05 \cdot 10^4$	$2.18 \cdot 10^4$	$2.19 \cdot 10^4$	∞	∞	∞
181	$1.65 \cdot 10^4$	$6.60 \cdot 10^3$	$1.18 \cdot 10^4$	$1.27 \cdot 10^4$	$2.29 \cdot 10^4$	$2.47 \cdot 10^4$	$2.46 \cdot 10^4$	∞	∞	∞
191	$1.81 \cdot 10^4$	$7.28 \cdot 10^3$	$1.33 \cdot 10^4$	$1.41 \cdot 10^4$	$2.58 \cdot 10^4$	$2.73 \cdot 10^4$	$2.73 \cdot 10^4$	∞	∞	∞
n, d	OMd									
100, 50	$4.51 \cdot 10^2$	$4.93 \cdot 10^2$	$5.65 \cdot 10^2$	$5.69 \cdot 10^2$	$6.49 \cdot 10^2$	$6.75 \cdot 10^2$	$6.81 \cdot 10^2$	∞	∞	∞
200, 100	$1.04 \cdot 10^3$	$1.09 \cdot 10^3$	$1.26 \cdot 10^3$	$1.31 \cdot 10^3$	$1.47 \cdot 10^3$	$1.55 \cdot 10^3$	$1.57 \cdot 10^3$	∞	∞	∞
300, 150	$1.72 \cdot 10^3$	$1.74 \cdot 10^3$	$2.03 \cdot 10^3$	$2.05 \cdot 10^3$	$2.40 \cdot 10^3$	$2.51 \cdot 10^3$	$2.51 \cdot 10^3$	∞	∞	∞
400, 200	$2.43 \cdot 10^3$	$2.43 \cdot 10^3$	$2.80 \cdot 10^3$	$2.90 \cdot 10^3$	$3.42 \cdot 10^3$	$3.56 \cdot 10^3$	$3.53 \cdot 10^3$	∞	∞	∞
500, 250	$3.12 \cdot 10^3$	$3.16 \cdot 10^3$	$3.65 \cdot 10^3$	$3.72 \cdot 10^3$	$4.34 \cdot 10^3$	$4.58 \cdot 10^3$	$4.60 \cdot 10^3$	∞	∞	∞
n, k	XdivK, switch point in the end									
60, 3	$3.94 \cdot 10^4$	$2.44 \cdot 10^5$	$2.79 \cdot 10^5$	$2.53 \cdot 10^5$	$7.10 \cdot 10^4$	$5.82 \cdot 10^4$	$5.95 \cdot 10^4$	∞	$2.95 \cdot 10^5$	$3.16 \cdot 10^7$
72, 3	$6.79 \cdot 10^4$	$4.18 \cdot 10^5$	$4.93 \cdot 10^5$	$4.19 \cdot 10^5$	$1.15 \cdot 10^5$	$1.03 \cdot 10^5$	$1.03 \cdot 10^5$	∞	$4.98 \cdot 10^5$	$3.30 \cdot 10^8$
84, 3	$1.08 \cdot 10^5$	$6.57 \cdot 10^5$	$7.69 \cdot 10^5$	$6.55 \cdot 10^5$	$1.72 \cdot 10^5$	$1.61 \cdot 10^5$	$1.64 \cdot 10^5$	∞	$7.81 \cdot 10^5$	∞
96, 3	$1.60 \cdot 10^5$	$9.91 \cdot 10^5$	$1.21 \cdot 10^6$	$9.97 \cdot 10^5$	$2.59 \cdot 10^5$	$2.39 \cdot 10^5$	$2.44 \cdot 10^5$	∞	$1.05 \cdot 10^6$	∞
108, 3	$2.28 \cdot 10^5$	$1.34 \cdot 10^6$	$1.75 \cdot 10^6$	$1.45 \cdot 10^6$	$3.49 \cdot 10^5$	$3.34 \cdot 10^5$	$3.34 \cdot 10^5$	∞	$1.63 \cdot 10^6$	∞
120, 3	$3.12 \cdot 10^5$	$1.93 \cdot 10^6$	$2.32 \cdot 10^6$	$1.97 \cdot 10^6$	$4.87 \cdot 10^5$	$4.70 \cdot 10^5$	$4.76 \cdot 10^5$	∞	$2.37 \cdot 10^6$	∞
n, k	XdivK, switch point in the middle									
60, 3	$3.94 \cdot 10^4$	$6.61 \cdot 10^3$	$1.06 \cdot 10^4$	$1.20 \cdot 10^4$	$7.02 \cdot 10^4$	$5.82 \cdot 10^4$	$5.76 \cdot 10^4$	∞	$1.17 \cdot 10^4$	$1.52 \cdot 10^6$
72, 3	$6.79 \cdot 10^4$	$1.12 \cdot 10^4$	$1.78 \cdot 10^4$	$2.11 \cdot 10^4$	$1.19 \cdot 10^5$	$1.03 \cdot 10^5$	$1.01 \cdot 10^5$	∞	$2.00 \cdot 10^4$	$1.43 \cdot 10^7$
84, 3	$1.08 \cdot 10^5$	$1.79 \cdot 10^4$	$3.02 \cdot 10^4$	$3.35 \cdot 10^4$	$1.73 \cdot 10^5$	$1.61 \cdot 10^5$	$1.64 \cdot 10^5$	∞	$3.08 \cdot 10^4$	$1.57 \cdot 10^8$
96, 3	$1.60 \cdot 10^5$	$3.01 \cdot 10^4$	$4.13 \cdot 10^4$	$5.15 \cdot 10^4$	$2.60 \cdot 10^5$	$2.39 \cdot 10^5$	$2.44 \cdot 10^5$	∞	$4.75 \cdot 10^4$	∞
108, 3	$2.28 \cdot 10^5$	$4.54 \cdot 10^4$	$5.86 \cdot 10^4$	$6.71 \cdot 10^4$	$3.74 \cdot 10^5$	$3.34 \cdot 10^5$	$3.43 \cdot 10^5$	∞	$6.88 \cdot 10^4$	∞
120, 3	$3.12 \cdot 10^5$	$6.46 \cdot 10^4$	$8.25 \cdot 10^4$	$9.37 \cdot 10^4$	$4.98 \cdot 10^5$	$4.70 \cdot 10^5$	$4.59 \cdot 10^5$	∞	$9.32 \cdot 10^4$	∞

the single state and $\varepsilon = 0.1$ was the most efficient algorithm for LeadingOnes, OM_d and XdivK with switch point in the middle.

We theoretically proved that EA+RLnM on XdivK has asymptotically the same runtime as RLS. Also we shown that performance of EA+RLnM is independent of the number of switch points and their positions, while performance of EA+RLM depends on these factors. Particularly, EA+RLnM achieves the best results on XdivK with switch point in the end.

8 ACKNOWLEDGMENTS

This work was financially supported by the Government of Russian Federation, Grant 074-U01, and by RFBR according to the research project No. 16-31-00380 mol_a.

REFERENCES

[1] Dimo Brockhoff, Tobias Friedrich, Nils Hebbinghaus, Christian Klein, Frank Neumann, and Eckart Zitzler. 2009. On the Effects of Adding Objectives to Plateau Functions. *IEEE Transactions on Evolutionary Computation* 13, 3 (2009), 591–603.

[2] Maxim Buzdalov and Arina Buzdalova. 2014. OneMax Helps Optimizing XdivK: Theoretical Runtime Analysis for RLS and EA+RL. In *Proceedings of Genetic and Evolutionary Computation Conference Companion.* ACM, 201–202.

[3] Maxim Buzdalov and Arina Buzdalova. 2015. Can OneMax Help Optimizing LeadingOnes using the EA+RL Method?. In *Proceedings of IEEE Congress on Evolutionary Computation.* 1762–1768.

[4] Arina Buzdalova and Maxim Buzdalov. 2012. Increasing Efficiency of Evolutionary Algorithms by Choosing between Auxiliary Fitness Functions with Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning and Applications*, Vol. 1. 150–155.

[5] Arina Buzdalova, Irina Petrova, and Maxim Buzdalov. 2016. Runtime Analysis of Different Approaches to Select Conflicting Auxiliary Objectives in the Generalized OneMax Problem. In *Proceedings of IEEE Symposium Series on Computational Intelligence.* 280–286.

[6] Julia Handl, Simon C. Lovell, and Joshua D. Knowles. 2008. Multiobjectivization by Decomposition of Scalar Cost Functions. In *Parallel Problem Solving from Nature – PPSN X.* Number 5199 in Lecture Notes in Computer Science. Springer, 31–40.

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

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

[9] Darrell F. Lochtefeld and Frank W. Ciarallo. 2011. Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem. *Applied Soft Computing* 11, 6 (2011), 4161–4174.

[10] Frank Neumann and Ingo Wegener. 2008. Can Single-Objective Optimization Profit from Multiobjective Optimization? In *Multiobjective Problem Solving from Nature.* Springer Berlin Heidelberg, 115–130.

[11] Irina Petrova, Arina Buzdalova, and Georgiy Korneev. 2016. Runtime analysis of random local search with reinforcement based selection of non-stationary auxiliary objectives: initial study. In *Proceedings of 22nd International Conference on Soft Computing MENDEL 2016.* Czech Republic, 95–102.

[12] C. Segura, C. A. C. Coello, G. Miranda, and C. Léon. 2013. Using multi-objective evolutionary algorithms for single-objective optimization. *4OR* 3, 11 (2013), 201–228.

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