2014 13th International Conference on Machine Learning and Applications

A New Algorithm for Adaptive Online Selection of Auxiliary Objectives

Arina Buzdalova, Maxim Buzdalov
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia, 197101
Email: {abuzdalova, mbuzdalov}@gmail.com

***Abstract*—Consider optimization problems, where a target objective should be optimized. Some auxiliary objectives can be used to obtain the optimum of the target objective in less number of objective evaluations. We call such auxiliary objective a *supporting* one. Usually there is no prior knowledge about properties of auxiliary objectives, some objectives can be *obstructive* as well. What is more, an auxiliary objective can be both supporting and obstructive at different stages of the target objective optimization. Thus, an adaptive online method of objective selection is needed. Earlier, we proposed a method for doing that, which is based on reinforcement learning.**

In this paper, a new algorithm for adaptive online selection of optimization objectives is proposed. The algorithm meets the interface of a reinforcement learning agent, so it can be fit into the previously proposed framework. The new algorithm is applied for solving some benchmark problems with single-objective evolutionary algorithms. Specifically, LEADINGONES with ONEMAX auxiliary objective is considered, as well as the MH-IFF problem. Experimental results are presented. The proposed algorithm outperforms Q-learning and random objective selection on the considered problems.

I. INTRODUCTION

Evolutionary algorithms (EAs) are often used to optimize a single objective (in other words, a *fitness function*). Single-objective optimization can often benefit from multiple objectives [1], [2]. Some researchers introduce additional objectives to escape from the plateaus [3]. Decomposition of the primary objective into several objectives also helps in many problems [4], [5]. Additional objectives may also arise from the problem structure [6].

Different approaches may be applied to a problem with the "original" objective, which can be called the *target* objective, and some auxiliary objectives. The *multi-objectivization* approach is to optimize *all* auxiliary objectives at once using a multi-objective optimization algorithm [4], [5]. The *helper-objective* approach is to optimize simultaneously the target objective and some (not necessarily all, in most cases, only one) auxiliary objectives, switching between them from time to time [7].

The approaches above are designed in the assumption that the auxiliary objectives are crafted to help optimizing the target objective. However, this is not always true, especially when the auxiliary objectives are generated automatically, or their properties are unknown. In fact, the auxiliary objectives may support or obstruct the optimization process. To cope with such situations, the EA+RL method was developed [8], where EA stands for an evolutionary algorithm and RL stands for reinforcement learning. The idea of this method is to use a single-objective evolutionary algorithm and switch between the objectives. To find the most suitable objective for the optimization, RL algorithms are used [9].

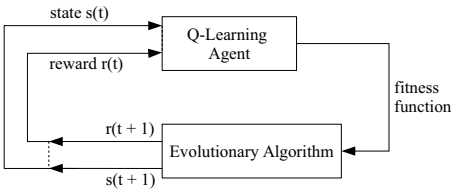

Fig. 1. EA+RL general scheme

In this paper a new EA+RL-like method is proposed. It does not use any particular RL algorithm, but suggests a new approach to maintain a selection strategy using rewards. The method is evaluated on some benchmark problems and compared with the original EA+RL approach.

The rest of the paper is organized as follows. First, the previously proposed EA+RL method is described. Second, the new algorithm is described and compared with the approach used in EA+RL. Finally, the experimental study is presented.

II. EA+RL METHOD

In the EA+RL method, a RL agent interacts with an evolutionary algorithm. The scheme of the method is shown in the Fig. 1, where t is the number of the current iteration. The agent selects an objective between the auxiliary objectives and the target one. Then, the selected objective is passed to the evolutionary algorithm. The next generation is evolved using the selected objective as the fitness function. The agent receives a numerical reward and some representation of the state. Then the agent updates its strategy using the obtained information and the process repeats. There are a number of approaches of state definition [8], [10], [11]. In this paper a single state is used, i. e. the evolutionary algorithm is treated as a stateless environment.

Let us discuss EA+RL in more detail. We should explain how the RL agent updates its strategy and how it selects an objective. We also should consider how rewards are defined.

978-1-4799-7415-3/14 $31.00 © 2014 IEEE
DOI 10.1109/ICMLA.2014.100

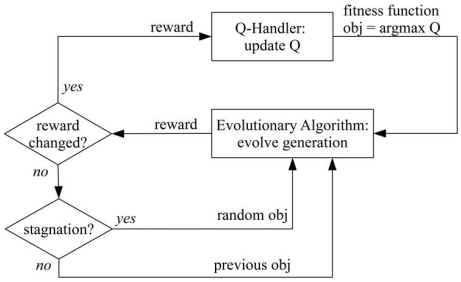

Fig. 2. General scheme of the proposed algorithm

A. Selection Strategy

Different RL algorithms can be used to implement the agent. Let us consider the Q-learning algorithm with a greedy strategy. This algorithm maintains the total expected reward estimation $Q(s, a)$ for each pair of state s and action a (in the EA+RL method an action corresponds to a selected objective). At each time when a reward r is obtained, Q is updated using the following formula: $Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$, where s' is the next state, the parameter α influences the rate of learning and γ is a discount factor [9]. The selection of an action (objective) to be applied is made using Q values. In the ε-greedy exploration strategy, a random action is taken with a probability of ε, otherwise the action $a = \arg\max_{a'} Q(s,a')$ is taken, which is the most efficient in the current state s. If $\varepsilon = 0$, we call such exploration strategy a *greedy* one.

B. Reward Definition

The goal of RL is to optimize total amount of reward [9]. In the EA+RL method, an immediate reward is based on the difference between the best target objective values in two sequential iterations. Hence, the total reward being maximized is roughly equivalent to the difference between the final and the initial values of the target objective.

III. Proposed Algorithm

A general scheme of the proposed algorithm is shown in Fig. 2. Compare it with the EA+RL scheme (Fig. 1). The reward is passed to the Q-handler (equivalent of the RL agent) only when it is changed since the previous iteration. Otherwise, it is checked whether a stagnation occurred. If so, a random fitness function is selected. Otherwise, a previously selected fitness function is used.

Therefore, the idea behind this algorithm is to gain reward during a number of iterations (while it does not change) and only then update selection strategy. Thus, the same objective can be applied for some number of iterations before gaining a positive effect on the target objective value.

If the reward does not change for a long time, a random objective should be chosen. To be more precise, the stagnation criterion is as follows. Let "ch" be the number of iterations performed since the reward changed for the last time. Let "lu" be the number of iterations between the start of the algorithm and the last change of reward. Stagnation appears when "lu" is equal to "ch".

Algorithm 1 The new algorithm for objective selection

```
1:  iterations counter: i = 0
2:  counter of iterations since reward change: ch = 0
3:  iteration number of last update: lu = 0
4:  previous reward value: prev = 0
5:  randomly choose current objective obj
6:  while (terminal state is not reached) do
7:    evolve new EA generation using obj
8:    calculate reward: r = new fitness − previous fitness
9:    if (reward changed: r ≠ prev) then
10:     calculate update value: val = r − prev
11:     update Q-value: Q[obj] = val + Q[obj]
12:     select objective: obj = arg max_a Q[a]
13:     update previous reward: prev = r
14:     reset iterations since change: ch = 1
15:     set iteration number of last update: lu = i
16:   else
17:     if (stagnation: ch == lu) then
18:       randomly select obj different from current
19:       reset iterations since change: ch = 1
20:     else
21:       increase iterations since change: ch = ch + 1
22:   increase iterations counter: i = i + 1
```

In Algorithm 1 a detailed pseudocode of the proposed algorithm is shown. The term "Fitness" refers to the best value of the target objective in a generation. The "New fitness" is the target fitness obtained after applying the selected objective and generating a new generation. The "Previous fitness" refers to the target fitness in the previous generation.

Comparison of the proposed algorithm with the EA+RL method is as follows.

A. Selection Strategy

Consider how an objective is selected. The selection of an objective is made with the Q-handler in a greedy way: obj $= \arg\max_{\text{obj}'} Q(\text{obj})$. But note that not only the Q-handler can select an objective. An objective can also be selected randomly in the case of stagnation, or the previously selected objective can be selected again in the case when the reward value did not change.

B. Reward Definition

The reward itself is calculated as the difference between the best target fitness values in two sequential generations, as in the previous EA+RL implementations. However, the update formula for the Q-handler is $Q[\text{obj}] = Q[\text{obj}] + r - r'$, there "obj" is the selected objective, i.e. the action, r is the new reward, r' is the previous reward value. So Q values are updated with reward differences, not with rewards.

IV. Experiments

In this section results of several experiments are shown. We consider three combinations of toy yet theoretically important problems: LeadingOnes and OneMax [12], XDivK and OneMax [13] and MH-IFF mh-iff. In the first two problems, the OneMax function is used as auxiliary objective. The

TABLE I
AVERAGE NUMBER OF GENERATIONS NEEDED TO REACH THE OPTIMUM FOR THE XDIVK AND ONEMAX PROBLEM

n	k	Fixed obj.	Random	EA+RL	Proposed
12	2	111.614	83.099	76.960	59.564
16	2	200.663	122.871	132.683	86.218
20	2	273.861	181.505	184.564	125.921
18	3	1212.22	372.624	439.277	116.733
24	3	2685.05	910.267	861.921	186.109
30	3	5365.62	1592.54	1611.42	245.109
24	4	15649.6	2437.60	2358.84	183.584
32	4	41529.4	5597.32	8301.37	277.238
40	4	93228.1	19659.2	14004.8	415.446
30	5	1.887e5	16378.8	15187.3	248.218
40	5	∞	55365.7	56800.2	360.436
50	5	∞	1.834e5	∞	588.792

value of ONEMAX is the number of bits in a candidate solution set to one. The rest of the functions are described in the corresponding sections. In each of the following sections, the proposed algorithm is compared with random selection of objectives, as well as with the previously known EA+RL method.

In all problems, a candidate solution is a bit vector of length n. We apply different objective selection methods to Randomized Local Search (RLS), also known as Random Mutation Hill Climber [5]. The mutation operator that flips one randomly chosen bit is used.

In the EA+RL method, the Q-learning algorithm with the greedy exploration strategy and a single state (i.e. without states) is used [9]. The parameters of Q-learning are a learning rate $\alpha = 0.5$ and a discount factor $\gamma = 0.5$. They were set during a preliminary experiment.

For all the considered problems, 100 runs of each algorithm are performed for each problem instance, then the results are averaged. The results of the experiments are tested for statistical significance at a level $p_0 = 0.05$.

A. XdivK and OneMax Objectives

Consider two objectives: the target one is XDIVK and the auxiliary one is ONEMAX. XDIVK is evaluated as follows: ONEMAX value is divided by some parameter k and the integer part of the result is taken. It was shown by theoretical analysis that ONEMAX is a supporting objective for XDIVK [13].

Experimental results are shown in Table I. Average number of generations needed to reach the target optimum is presented for each problem size n and for the each considered approach, so the lower values are, the better the corresponding method is. The *Fixed obj.* column refers to XDIVK optimization by RLS when no auxiliary objective is used. "Random" stands for the random selection of objectives applied to RLS. The ∞ sign is used to show that an algorithm have failed to find the target optimum after 10^6 iterations.

According to the experimental results, the proposed method seems to outperform all the considered approaches with growth of n. One can argue that for the proposed algorithm, as well as for the the EA+RL method, a one more fitness evaluation is needed to calculate a reward. But in the most applications of the reward-based selection methods, one expensive calculation per individual is enough to obtain the values of all the objectives [14]. In the problems considered here, one calculation of the number of 1-bits is also enough. Values of all the objectives, as well as the reward value, are simply calculated using this number. So calculating a reward does not increase the number of fitness evaluations.

TABLE II
AVERAGE NUMBER OF GENERATIONS AND ITS STANDARD DEVIATION (IN BRACES) NEEDED TO REACH THE OPTIMUM FOR THE LEADINGONES AND ONEMAX PROBLEM.

n	Fixed obj.	Random	EA+RL	Proposed
20	216.4 (73.9)	164.6 (65.7)	146.9 (84.1)	136.9 (57.2)
30	473.6 (140.4)	342.3 (120.6)	303.4 (197.4)	246.2 (101.9)
40	817.5 (217.8)	588.1 (187.9)	495.8 (355.3)	392.9 (164.8)
50	1294.0 (309.0)	906.9 (260.4)	765.3 (556.5)	548.7 (235.7)
60	1853.3 (405.9)	1294.2 (348.0)	1110.9 (815.8)	719.6 (314.1)
70	2535.1 (509.8)	1710.6 (431.0)	1423.6 (1117.4)	919.0 (398.5)
80	3226.7 (588.1)	2234.0 (512.2)	1837.7 (1476.7)	1126.4 (513.6)
90	4118.6 (716.2)	2800.9 (636.5)	2231.1 (1877.3)	1288.9 (585.5)

The Wilcoxon signed ranks test was performed to check if the new algorithm and the other considered ones can be distinguished. The test was performed using `stats::wilcox.test()` procedure from the R language [15].

The input data was obtained as follows. The integer parameter k was selected uniformly at random from the interval $2..5$ 20 times. An integer multiplier m was selected uniformly at random from the interval $2..10$ 20 times as well. Then each k was multiplied with the corresponding m, so we got $n = k * m$. Thus, we obtained 20 randomly generated pairs n, k. The average number of generations needed to reach the optima of the problem instances corresponding to these n, k pairs were taken as the input for the Wilcoxon test.

The proposed method was compared with the RLS without selection, random selection and the EA+RL method using the "less" alternative [15]. The corresponding p-values are 6.461×10^{-5}, 7.515×10^{-5} and 6.461×10^{-5} respectively. All the p-values are less than p_0, so the proposed method is statistically better than the others.

B. LeadingOnes and OneMax Objectives

In this problem, the target objective is LEADINGONES and the auxiliary objective is still ONEMAX. LEADINGONES is another well-known problem, where the number of one bits from the beginning of the candidate solution to the first zero-bit is calculated [12].

Experimental results are shown in Table II. The same notation as for the previous problem is used. Standard deviations are also presented (in braces). The *Fixed obj.* column refers to LEADINGONES optimization by RLS when there is no auxiliary objective.

As for the previous problem, the proposed algorithm seems to outperform all the considered approaches with growth of n. The Wilcoxon signed ranks test with the "less" alternative [15] was used to compare the proposed algorithm with the other methods. Integer lengths n were selected uniformly at random from the interval $10..100$, then the corresponding average number of generations needed to reach the optima were used as

TABLE III
AVERAGE FITNESS VALUES AND THEIR STANDARD DEVIATION (IN BRACES) FOR THE MH-IFF PROBLEM

n	Fixed obj.	Random	EA+RL	Proposed
16	41.6 (6.2)	80.0 (0.0)	49.1 (15.6)	63.0 (19.0)
32	83.4 (9.5)	62.0 (8.7)	87.2 (20.8)	108.9 (42.5)
64	167.7 (14.3)	122.6 (11.8)	170.0 (15.9)	203.9 (87.2)
128	338.5 (21.3)	242.5 (16.5)	336.6 (21.1)	387.9 (175.9)
256	675.1 (30.6)	488.9 (23.3)	676.6 (30.6)	718.5 (280.0)
512	1345.6 (37.1)	978.9 (32.3)	1345.9 (35.7)	1389.6 (377.3)
1024	2697.5 (59.2)	1954.3 (49.8)	2702.8 (58.5)	2787.2 (854.0)
2048	5401.7 (79.1)	3920.2 (68.9)	5390.2 (75.4)	6147.6 (3762.5)

an input for the test. All the p-values were 4.764×10^{-5}, which is less than p_0. Thus, the proposed algorithm is statistically better than the other ones.

C. MH-IFF Problem

Consider the MH-IFF benchmark problem. The target objective is Hierarchical-if-and-only-if function, H-IFF [5]. H-IFF should be maximized. Its formula f is given below, where B is a bit string individual, B_L and B_R are its left and right halves respectively.

The target objective f and the auxiliary objectives f_0 and f_1 have the following formula:

$$f_k(B) = \begin{cases} 0 & \text{if } |B| = 1 \text{ and } b_1 \neq k, \\ 1 & \text{if } |B| = 1 \text{ and } b_1 = k, \\ |B| + f_k(B_L) + f_k(B_R) & \text{if } \forall i \{b_i = k\}, \\ f_k(B_L) + f_k(B_R) & \text{otherwise.} \end{cases}$$

$$f(B) = f_0 + f_1$$

The objectives f_0 and f_1 allow to escape from local optima of f [5], hence they are supporting ones.

During the experiment, different bit string lengths n were considered. An algorithm was run until 500000 fitness function evaluations were performed. The value of the target objective f obtained in the final generation was averaged over all the runs of an algorithm. Average target objective values obtained using different algorithms are presented in the Table III, as well as the corresponding standard deviation (in braces). The *Fixed obj.* column refers to optimization of f using RLS while no auxiliary objective is used. The greater values correspond to the higher efficiency of the corresponding algorithm, since the goal is to maximize f.

The proposed algorithm was compared with the H-IFF, random selection and EA+RL using Wilcoxon rank sum test with the "greater" alternative [15]. For the each considered n, target objective values obtained after each of 100 runs were taken as an input for the test.

For the comparison with random selection, all the p-values were less than p_0. For the other two methods, p-values are less than p_0 for the lengths from 8 to 64 only. For the greater lengths, it was impossible to distinguish the proposed method from the RLS without selection and the EA+RL method using Wilcoxon rank sum test. Alghough, the average target objective values obtained with the proposed algorithm were still better also for the greater lengths. Thus, probably a more powerful statistical test would help.

V. CONCLUSION

A new algorithm for online auxiliary objectives selection is proposed. It is empirically evaluated on three benchmark problems. For the considered problems, the proposed algorithm seems to outperform not only a conventional evolutionary algorithm without auxiliary objectives, but also the previously known EA+RL method of objectives selection, as well as the random selection approach. The proposed algorithm is shown to be statistically distinguishable from the random selection for all the considered problems.

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

REFERENCES

[1] F. Neumann and I. Wegener, "Can Single-Objective Optimization Profit from Multiobjective Optimization?" in *Multiobjective Problem Solving from Nature*, ser. Natural Computing Series. Springer Berlin Heidelberg, 2008, pp. 115–130.

[2] T. Brys, A. Harutyunyan, P. Vrancx, M. E. Taylor, D. Kudenko, and A. Nowé, "Multi-objectivization of reinforcement learning problems by reward shaping," in *2014 International Joint Conference on Neural Networks*, 2014, pp. 2315–2322.

[3] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler, "On the Effects of Adding Objectives to Plateau Functions," *Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 591–603, 2009.

[4] J. Handl, S. C. Lovell, and J. D. Knowles, "Multiobjectivization by Decomposition of Scalar Cost Functions," in *Parallel Problem Solving from Nature Parallel Problem Solving from Nature X*, ser. Lecture Notes in Computer Science. Springer, 2008, no. 5199, pp. 31–40.

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

[6] D. F. Lochtefeld and F. W. Ciarallo, "Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem," *Applied Soft Computing*, vol. 11, no. 6, pp. 4161–4174, 2011.

[7] M. T. Jensen, "Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation: Evolutionary Computation Combinatorial Optimization," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 323–347, 2004.

[8] A. Buzdalova and M. Buzdalov, "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, 2012, pp. 150–155.

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

[10] S. Müller, N. N. Schraudolph, and P. D. Koumoutsakos, "Step Size Adaptation in Evolution Strategies using Reinforcement Learning," in *Proceedings of the Congress on Evolutionary Computation*. IEEE, 2002, pp. 151–156.

[11] G. Karafotias, Á. E. Eiben, and M. Hoogendoorn, "Generic parameter control with reinforcement learning," in *Genetic and Evolutionary Computation Conference, GECCO '14*, 2014, pp. 1319–1326.

[12] P. S. Oliveto, J. He, and X. Yao, "Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 281–293, 2007.

[13] M. Buzdalov and A. Buzdalova, "Onemax Helps Optimizing XdivK: Theoretical Runtime Analysis for RLS and EA+RL," in *Proceedings of Genetic and Evolutionary Computation Conference Companion*. ACM, 2014, pp. 201–202.

[14] ——, "Adaptive Selection of Helper-Objectives for Test Case Generation," in *2013 IEEE Congress on Evolutionary Computation*, vol. 1, 2013, pp. 2245–2250.

[15] R Core Team. (2013) R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. [Online]. Available: http://www.R-project.org/