Selection of Auxiliary Objectives with Multi-Objective Reinforcement Learning

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

Anna Matveeva
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
matveeva@rain.ifmo.ru

Georgiy Korneev
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
kgeorgiy@rain.ifmo.ru

ABSTRACT

Efficiency of evolutionary algorithms may be increased using multi-objectivization. Multi-objectivization is performed by adding some auxiliary objectives. We consider selection of these objectives during a run of an evolutionary algorithm.

One of the selection methods is based on reinforcement learning. There are several types of rewards previously used in reinforcement learning for adjusting of evolutionary algorithms. However, there is no superior reward. At the same time, reinforcement learning itself may be enhanced by multi-objectivization. So we propose a method for selection of auxiliary objectives based on multi-objective reinforcement learning, where the reward is composed of the previously used single rewards. Hence, we have double multi-objectivization: several rewards are involved in selection of several auxiliary objectives.

We run the proposed method on different benchmark problems and compare it with a conventional evolutionary algorithm and a method based on single-objective reinforcement learning. Multi-objective reinforcement shows competitive behavior and is especially useful in the case when we do not know in advance which of the single rewards is efficient.

CCS Concepts

•**Computing methodologies** → *Genetic algorithms; Reinforcement learning;*

Keywords

multi-objectivization; helper objectives; H-IFF; LeadingOnes

1. INTRODUCTION

Efficiency of a single-objective evolutionary algorithm (EA) can be increased by using auxiliary objectives [6,8]. Once we have auxiliary objectives, we should decide how to use them in EA. One way is to simultaneously optimize all auxiliary objectives obtained by decomposing the target objective [8]. Another way was proposed in [6]: it was shown that it may be efficient to dynamically select an auxiliary objective from the set of objectives and apply it in a number of EA generations, then select another objective and so on. The objectives were selected in a random order. Then a special approach for selection of objectives in Job-Shop Scheduling problem appeared [9]. Later it was proposed to select auxiliary objectives using reinforcement learning (RL) [4].

Basically, reinforcement learning is used in evolutionary algorithms for the two main purposes: for parameter control and for selection of auxiliary objectives [7]. In reinforcement learning, it is important to define a reward function. In both mentioned cases, reward functions are used to evaluate the growth of efficiency of an evolutionary algorithm after applying a specific parameter setting or selecting a specific auxiliary objective. In all the corresponding research works just one reward function is used at a time. However, in reinforcement learning it may sometimes be of benefit to multi-objectivize reward [2]. The goal of the present paper is to investigate whether using multi-objective reward in EA+RL is promising.

In the following sections, we propose a method to select auxiliary objectives in evolutionary algorithms using multi-objective reinforcement learning (MORL) called EA+MORL and evaluate the proposed method on a number of benchmark problems of different difficulty.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739482.2768473

2. RELATED WORK: EA+RL METHOD

We consider single-objective optimization of a *target* objective t. The optimization is performed using a single-objective EA. There is a set A of predefined auxiliary objectives. We do not know properties of auxiliary objectives in advance, although it is implied that optimizing some of them instead of t may decrease the number of EA generations needed to find the optimum of t. An objective to be optimized at the current EA generation is selected from $A \cup \{t\}$ with reinforcement learning.

In reinforcement learning, an agent selects an action and applies it to an environment. The environment returns a numerical reward and some representation of its state. The agent updates the quality estimation of the actions according to the reward, makes a new selection and so on.

The general scheme of using reinforcement for selection of auxiliary objectives was proposed in the EA+RL method [4]. In this method the environment corresponds to EA and the actions correspond to the auxiliary objectives. To apply an action $a \in A \cup \{t\}$ means to select an objective a to be used as a fitness function in the current generation of EA.

We consider optimization problems where all objectives, including the target objective, are calculated during one processing of an individual [3, 4]. Therefore, evaluation of reward does not increase the number of objective evaluations.

3. EA+MORL APPROACH

Below we describe the rewards used in the proposed method and then the method itself. We use the following notation: the i^{th} generation of EA is denoted as G_i; an individual is $x \in G_i$; the number of individuals in a generation is considered to be the same for all generations, it is denoted as $|G|$. The target objective is referred to as t.

3.1 Rewards

The *Max* reward reflects how the best target objective value has changed. It is calculated using the following formula, where i is the number of EA generation: $\text{Max}(i) = \max_{x \in G_i} t(x) - \max_{x \in G_{i-1}} t(x)$. Similar rewards based on the best fitness in a generation were previously used in both EA+RL [4] and parameter control methods [5].

The *Avg* reward is the difference between the averaged target fitness of the current generation and the previous generation [3]: $\text{Avg}(i) = \left(\sum_{x \in G_i} t(x) - \sum_{x \in G_{i-1}} t(x)\right) / |G|$.

The *Sgn* reward corresponds to the sign of difference between the best target values in two subsequent generations: $\text{Sgn}(i) = \text{sgn}(\text{Max}(i))$. Similar reward was used for parameter control in evolution strategies [10].

We also propose the *Div* reward, which may be interpreted as difference of fitness diversity in subsequent generations: $\text{Div}(i) = \min_{x \neq y \in G_i} |t(x) - t(y)| - \min_{x \neq y \in G_{i-1}} |t(x) - t(y)|$.

3.2 EA+MORL method description

The pseudocode of the proposed method is presented in Algorithm 1. We call it EA+MORL, because it is based on reinforcement learning with multi-objective reward, or multi-objective reinforcement learning (MORL) [2].

Algorithm 1 The EA+MORL method

Require: t — target obj.; A — aux. objectives; ε — exploration pr.; α — learning rate; γ — discount factor.

1: Initialize $Q(a)_i$ for each $a \in A \cup \{t\}$ and reward type r_i
2: **while** target optimum is not found by EA **do**
3: $p \leftarrow$ random number $\in [0, 1]$
4: **if** $(p < \varepsilon)$ **then**
5: $a \leftarrow$ random objective $a \in A \cup \{t\}$
6: **else**
7: NonDominatedSet $\leftarrow \{a | \forall a' ((\forall i\ Q(a)_i \geq Q(a')_i)$
8: or $(\exists i\ Q(a)_i > Q(a')_i))\}$
9: $a \leftarrow$ random element of NonDominatedSet
10: Pass objective a to EA as fitness function
11: Evolve the next generation of EA
12: Calculate rewards $r_0, r_1 ... r_n$
13: **for** i = 0 to n **do**
14: $Q(a)_i \leftarrow Q(a)_i + \alpha(r_i + \gamma \max_{a'} Q(a')_i - Q(a)_i)$

For the sake of simplicity, we use a single state in reinforcement learning, i.e. the environment is considered as stateless. The efficiency of an action a is estimated using a vector $Q(a) \in \mathrm{R}^n$, where n is the number of rewards. In experiments we use up to four different rewards described in the previous section. For example, $Q(a)_0$ may be estimated using the *Max* reward, $Q(a)_1$ using *Avg* and so on.

Table 1: Parameters for different target objectives

Parameter	Leading Ones	One Max	H-IFF	H-IFF, obstr.
individual length	300	300	64	64
mutation pr.	0.007	0.007	0.01	0.01
learning rate α	0.6	0.6	0.75	0.75
discount factor γ	0.1	0.1	0.1	0.05
exploration pr. ε	0.05	0.01	0.01	0.01

The main difference between the proposed method and the EA+RL method is that several rewards are used and, consequently, selection of an action (lines 7–9 in Algorithm 1) and updating of Q-values (lines 12–14) are more sophisticated than in EA+RL.

4. DESCRIPTION OF EXPERIMENTS

We consider several benchmark problems of different difficulty and with different types of auxiliary objectives. Since properties of auxiliary objectives are typically not known in advance, we should consider benchmark problems not only with efficient auxiliary objectives, but also with *obstructive* ones. When a target objective is being maximized, an auxiliary objective is considered to be obstructive if optimizing of this objective leads to decrease of the target objective.

In the next sections, problems with both efficient and obstructive objectives are considered. The difficulty of the target objective optimization varies from the simple linear OneMax function and a more difficult LeadingOnes to a highly multi-modal H-IFF function with both efficient objectives and an obstructive one.

Each of the considered problems were solved using a conventional EA, EA+RL and EA+MORL. In EA+RL, each of the rewards described in Section 3.1 was used in its own series of runs. In EA+MORL, different combinations of rewards were used. Each algorithm variation was run 45 times until the optimal value of the target criterion was found or the limit of 300000 generations was reached. We calculated the number of generations needed to reach the optimum in each run and then averaged the results. In each generation an equal number of fitness function evaluations was made. This number equals generation size of 100.

The results were tested for statistical significance when applicable. We used Wilcoxon rank sum test. As the input for the test, the numbers of generations needed to reach the optimum in each of the performed runs were used. The test was performed with the `stats::wilcox.test()` procedure from the R language [12]. We applied Holm correction to compare each algorithm with the rest of algorithms. For the level of statistical significance, we used $p_0 = 0.05$.

Let us consider settings used in the experiments. An evolutionary algorithm from the framework [1] was used with elite count of 5. The code which can be used to reproduce the experiments is published at GitHub[1]. The parameters of the evolutionary algorithm and reinforcement learning are given in Table 1. They were set during preliminary experiments. For all the problems, a homogeneous mutation were applied, i.e. each bit were inverted with a certain probability given in the table. Crossover was used only in the LeadingOnes problem. More precisely, we applied a single point crossover with probability of 0.1 in this problem.

[1]https://github.com/anna2912/MultiEARL

Table 2: Number of generations to optimize LeadingOnes with the OneMax auxiliary obj. Med – median, Mean – average, Std – standard deviation

Alg	Reward type	Med	Mean	Std
EA	No reward	2949	2950.13	272.89
EA+RL	Max	2501	2689.47	600.02
EA+RL	Avg	1676	1743.53	465.01
EA+RL	Sgn	2768	2804.49	649.44
EA+RL	Div	1045	1087.36	198.67
EA+MORL	Max, Avg, Sgn, Div	796	810.67	215.54
EA+MORL	Max, Sgn, Div	1056	1032.60	253.78
EA+MORL	Max, Avg	1137	1179.71	282.18
EA+MORL	Max, Sgn	2378	2463.22	533.86
EA+MORL	Max, Div	1122	1146.16	250.42
EA+MORL	Max, Avg, Sgn	1150	1191.78	347.80
EA+MORL	Max, Avg, Div	800	848.47	224.73
EA+MORL	Avg, Div	929	890.91	213.46
EA+MORL	Sgn, Div	976	988.20	229.69
EA+MORL	Avg, Sgn	1897	1921.73	305.95

5. RESULTS OF EXPERIMENTS

5.1 LeadingOnes with OneMax

The first problem we consider is LeadingOnes with OneMax auxiliary objective. In LeadingOnes the number of one bits from the beginning of an individual to the first zero-bit is calculated [11]. OneMax has the same optimum as LeadingOnes and is easier to optimize, so it is expected that using of OneMax would lead to optimizing of LeadingOnes in less number of generations.

The results are presented in Table 2. The best (second best) performing algorithms are highlighted with grey (light grey) background. It can be seen from the results that using several rewards simultaneously may lead to less number of generations needed to reach the optimum in the considered problem. The best combinations *(Max, Avg, Sgn, Div)* and *(Max, Avg, Div)* show statistically equal performance and significantly differ from the rest of algorithms, excluding *(Avg, Div)*.

5.2 OneMax with ZeroMax

OneMax is a well known benchmark function which is easily optimized by various EAs [11]. We add an obstructive objective ZeroMax in order to show that RL is able to ignore such objectives. Since OneMax counts the number of ones in a bit string and ZeroMax is just the opposite and counts the number of zeros, this is an extreme example of a problem with an obstructive objective. It was theoretically shown for randomized local search that EA+RL is able to solve the OneMax with ZeroMax problem asymptotically as fast as a conventional EA without ZeroMax.

As it was expected, OneMax without an obstructive objective is maximized with EA within the minimal number of generations comparing to the rest of approaches, since they use the obstructive objective. The question is how close the other approaches are to this baseline. It may be seen that in this problem the single reward *Avg* is the best one. According to the statistical test, it is significantly distinguishable from all the other approaches. Multiple rewards are not so efficient, though the *(Max, Avg)*, *(Avg, Sgn)*, *(Avg, Div)* and *(Max, Avg, Sgn)* combinations show competitive behavior.

Table 3: Number of generations to optimize OneMax with the ZeroMax obstructive objective

Alg	Reward type	Med	Mean	Std
EA	No reward	423	434.80	48.99
EA+RL	Max	944	960.53	398.04
EA+RL	Avg	470	471.93	55.39
EA+RL	Sgn	925	909.13	280.42
EA+RL	Div	1076	1028.29	258.53
EA+MORL	Max, Avg, Sgn, Div	1009	1019.00	444.27
EA+MORL	Max, Sgn, Div	1150	1087.29	319.55
EA+MORL	Max, Avg	657	748.89	261.32
EA+MORL	Max, Sgn	896	911.04	349.49
EA+MORL	Max, Div	1046	1082.96	351.68
EA+MORL	Max, Avg, Sgn	836	857.60	281.74
EA+MORL	Max, Avg, Div	1077	1067.76	347.23
EA+MORL	Avg, Div	830	850.20	250.43
EA+MORL	Sgn, Div	1056	1008.16	342.87
EA+MORL	Avg, Sgn	749	787.27	289.14

Table 4: Number of generations to optimize H-IFF with efficient auxiliary objectives, % – percent of runs when the optimum was found

Alg	Reward type	Med	Mean	Std	%
EA	No reward	–	–	0	0
EA+RL	Max	2094	–	–	73
EA+RL	Avg	562	765.07	785.49	100
EA+RL	Sgn	1353	–	–	67
EA+RL	Div	–	–	–	33
EA+MORL	Max, Avg, Sgn, Div	4766	4709.44	1799.42	100
EA+MORL	Max, Sgn, Div	–	–	–	55
EA+MORL	Max, Avg	4518	4404.07	2193.69	100
EA+MORL	Max, Sgn	2645	–	–	71
EA+MORL	Max, Div	3630	–	–	73
EA+MORL	Max, Avg, Sgn	3879	4396.93	1791.25	100
EA+MORL	Max, Avg, Div	2754	4463.62	2011.34	100
EA+MORL	Avg, Div	2530	2776.38	717.5	100
EA+MORL	Sgn, Div	–	–	–	55
EA+MORL	Avg, Sgn	4252	4385.89	2003.79	100

5.3 H-IFF

Consider the problem of maximizing the Hierarchical-if-and-only-if function, H-IFF [8]. The H-IFF target objective may be represented as a sum of functions f_0 and f_1. To obtain these functions, consider $k = 0$ and $k = 1$ in Eq. 1, where B is a bit string individual, B_L and B_R are its left and right halves respectively. The functions f_0 and f_1 are used as auxiliary objectives, it was shown that they allow to escape from local optima of H-IFF [8].

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

The results of optimizing H-IFF with auxiliary objectives are presented in Table 4. The "–" sign means that the corresponding value is greater than 10000. The results of the best (second best) algorithm are highlighted with grey (light

Table 5: Number of generations to optimize H-IFF with an obstructive objective

Alg	Reward type	Med	Mean	Std	%
EA	No reward	–	–	0	0
EA+RL	Max	1179	–	–	64
EA+RL	Avg	542	715.93	610.41	100
EA+RL	Sgn	–	–	–	60
EA+RL	Div	–	–	–	40
EA+MORL	Max, Avg, Sgn, Div	6149	5919.67	2172.51	100
EA+MORL	Max, Sgn, Div	4806	–	–	53
EA+MORL	Max, Avg	4238	4505.27	1860.75	100
EA+MORL	Max, Sgn	3928	–	–	60
EA+MORL	Max, Div	–	–	–	47
EA+MORL	Max, Avg, Sgn	4118	4718.67	2062.41	100
EA+MORL	Max, Avg, Div	5161	5121.49	1953.41	100
EA+MORL	Avg, Div	3006	3186.11	609.09	100
EA+MORL	Sgn, Div	–	–	–	38
EA+MORL	Avg, Sgn	4433	4704.49	2359.51	100

grey) background. According to the statistical test, they significantly differ from all other algorithms with 100% success.

Evolutionary algorithm without auxiliary functions is unable to find the optimum during the given number of generations. The optimum may be found using the EA+RL method with different single rewards. The *Avg* reward is the most efficient with 100% success. While multi-objective rewards yield higher number of generations to find an optimum, they still provide a good compromise when we do not want to test every single reward, but rather run a multi-objective combination to get an optimum with 100% success. For this purpose, the (Avg, Div) combination is the most useful.

5.4 H-IFF with obstructive objective

This problem is similar to the one described in Section 5.3. Here we add an obstructive objective, namely the number of overlaps with the string of alternating ones and zeros: 101010... Such an objective tends to destroy building blocks needed for H-IFF, so it is obstructive.

The results (Table 5) are similar to the results for the analogous problem without the obstructive objective, so the obstructive objective is efficiently ignored as intended. The first and second best rewards are *Avg* and *Avg, Div* correspondingly.

6. CONCLUSION

We proposed a method of using reinforcement learning with multi-objective reward for selection of auxiliary objectives in evolutionary algorithms. This method was compared with a conventional evolutionary algorithm and previously used reinforcement learning with various single rewards. Auxiliary objectives of both efficient and obstructive types were considered.

For the most of the considered problems, the average fitness difference *Avg* turned to be the most efficient single reward. The multi-objective reward comprised of *Avg* and the fitness diversity difference also showed high efficiency in the most cases and outperformed all single rewards in the LeadingOnes problem. Therefore, using of multi-objective reinforcement learning for selection of auxiliary objectives is promising.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] Watchmaker framework for evolutionary computation. http://watchmaker.uncommons.org.

[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*, pages 2315–2322, 2014.

[3] M. Buzdalov, A. Buzdalova, and I. Petrova. Generation of Tests for Programming Challenge Tasks Using Multi-Objective Optimization. In *Proceedings of Genetic and Evolutionary Computation Conference Companion*, pages 1655–1658. ACM, 2013.

[4] 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*, volume 1, pages 150–155, 2012.

[5] A. E. Eiben, M. Horvath, W. Kowalczyk, and M. C. Schut. Reinforcement Learning for Online Control of Evolutionary Algorithms. In *Proceedings of the 4th international conference on Engineering self-organising systems*, pages 151–160. Springer-Verlag, Berlin, Heidelberg, 2006.

[6] M. T. Jensen. 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.

[7] G. Karafotias, M. Hoogendoorn, and A. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *Evolutionary Computation, IEEE Transactions on*, PP(99):1–1, 2014.

[8] 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*, pages 269–283. Springer-Verlag, 2001.

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

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

[11] 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*, 4(3):281–293, 2007.

[12] R Core Team. R: A language and environment for statistical computing. http://www.R-project.org/, 2013.