OPTIMIZATION WITH AUXILIARY CRITERIA USING EVOLUTIONARY ALGORITHMS AND REINFORCEMENT LEARNING[*]

Arina Afanasyeva and Maxim Buzdalov

National Research University of Information Technologies, Mechanics and Optics
Computer Technologies Department
49 Kronverkskiy prosp., Saint-Petersburg
Russia
afanasyevarina@gmail.com, mbuzdalov@gmail.com

Abstract: *In this paper, an optimization method EA + RL based on an evolutionary algorithm controlled by reinforcement learning is proposed. Reinforcement learning is used to choose the most effective fitness function at each generation of the evolutionary algorithm. The method can be applied in scalar optimization with auxiliary criteria to speed up the optimization process. Experimental results for a model problem H-IFF are given. Applying of the method doubles mean fitness obtained with evolution strategy. A comparison with other evolutionary optimization methods is performed. The proposed method outperforms all the considered scalar optimization methods and most of the multicriteria ones.*

Keywords: *scalar optimization, multicriteria optimization, reinforcement learning, evolutionary algorithms, fitness function, H-IFF*

1 Introduction

There are various ways to increase the efficiency of scalar optimization. Some of them involve using auxiliary criteria. For example, a scalar optimization problem can be multi-objectivized [13] in order to prevent getting stuck in a local optima. Auxiliary criteria can also be provided by the object domain [4]. In this case some of them may be useful at different stages of scalar optimization, but it is unknown which criterion should be used actually.

In this paper a method for increasing of the efficiency of scalar optimization using auxiliary criteria is proposed. It is implied that the criteria are already developed and there is no prior knowledge about them. Thus a scalar optimization problem with auxiliary criteria arises. This problem is solved with evolutionary algorithm (EA) guided by reinforcement learning (RL) [2, 9, 10, 20]. The proposed method will be referred to as *EA + RL method*.

It should be noted that the auxiliary criteria are unimportant by themselves. In a conventional multicriteria optimization [5, 6] all the criteria must be optimized. There exists problems in which some criteria are less important than the main one [3], but all the criteria are optimized eventually. In the proposed problem only the main criterion should be optimized. We call it the *target* criterion.

The proposed method implements on-the-fly adjusting of an evolutionary algorithm. The auxiliary criteria, as well as the target criterion, correspond to fitness functions used in the evolutionary algorithm. The EA + RL method chooses of the most effective fitness function at each generation. To our knowledge, in other EA-adjusting methods some fixed fitness function is usually tuned [7, 8]. What is more, reinforcement learning is a promising modern approach itself. Its applicability in different fields is not fully investigated yet [9, 10]. There are few works that explore adjusting of such parameters as probabilities of applying evolutionary operators or some quantitative properties of individuals generation using reinforcement learning [7, 17]. This work contributes to investigation of reinforcement learning applicability for adjusting fitness functions in evolutionary algorithms.

In our previous work [1] a prototype of the proposed method was developed for solving a model problem using a genetic algorithm. The results of that work confirm that the method allows to choose the most effective auxiliary fitness functions at different stages of optimization process. In this paper we formulate the scalar optimization problem with auxiliary criteria and describe the EA + RL method in general case. The efficiency of the proposed method is tested on the H-IFF problem [13] that allows to compare it with some multicriteria optimization methods. It is implied, though, that the range of the EA + RL applicability is wider than the one of multicriteria methods, since the proposed method is not designed for optimizing all the criteria and can omit the ineffective ones.

[*]This paper appeared in the Proceedings of the 18th International Conference on Soft Computing MENDEL 2012, Brno, Czech Republic, June 27-29, 2012, pp. 58-63.

2 Optimization Problem with Auxiliary Criteria

Consider the formalization of the optimization problem with auxiliary criteria. Let W be the search space. Denote all acceptable solutions contained in the search space by $X : X \subseteq W$. Consider the *target criterion*:

$$g : W \to \mathbb{R}. \tag{1}$$

Consider a set H of k *auxiliary criteria*:

$$H = \{h_i(x)\}_{i=1}^{k}, h_i : W \to \mathbb{R}. \tag{2}$$

The problem is to maximize the target criterion (1) using the auxiliary criteria (2) to speed up the optimization process if it is possible:

$$g(x) \to \max_{x \in X}, X \subseteq W. \tag{3}$$

The solution of the problem is $x^* \in X : g(x^*) \geq g(x), \forall x \in X$.

In general, we have no exact prior knowledge about the kind of correlation between the target function and the auxiliary criteria. However, it often happens that some of the auxiliary criteria correlate with the target one at least at some stages of the optimization process. Such an assumption makes the proposed method suitable to speed up the optimization.

3 EA + RL Method

In this section a method of solving the stated optimization problem with auxiliary criteria is proposed. The proposed EA + RL method is based on an evolutionary algorithm (EA) guided by reinforcement learning (RL) [2,9,10,20]. The *fitness function* term is usually used instead of *criterion* when speaking about evolutionary algorithms. It is used further as well.

The scheme of the proposed method is shown in Fig. 1, where t is the number of the current generation evolved by the evolutionary algorithm. The *agent* interacts with the *environment* associated with the evolutionary algorithm. It chooses the fitness function to be used in the next generation. Then the agent gets a *reward* based on the difference of the target fitness in two sequential generations as well as a *state* mapped from the newly evolved generation and the process repeats. The agent uses a reinforcement learning algorithm that maximizes the total reward. The higher is the reward, the bigger is the increase of the target fitness, so good choices of the agent lead to the better performance of the optimization algorithm.

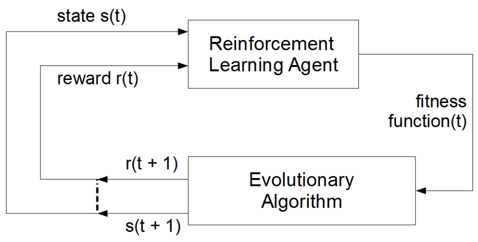

Figure 1: Scheme of the EA + RL method.

In the following sections we consider speeding up of an evolutionary algorithm as a reinforcement learning task [20] formally and then present the method itself in more detail.

3.1 Reinforcement Learning Task

The reinforcement learning task [20] consists of the set of actions A, a definition of the environment states $s \in S$ and the reward function $R : S \times A \to X \subseteq \mathbb{R}$. In this section, speeding up an evolutionary algorithm using auxiliary criteria is formulated as a reinforcement learning task.

Let x be an individual evolved by the evolutionary algorithm. Denote the i-th generation by G_i. The set of actions A corresponds to the set of all criteria, consisting of g — the target criterion and the elements of H — the set of auxiliary criteria: Taking an action means choosing some criterion $f_i \in A$ as the fitness function that is used in the generation G_i.

$$A = H \cup \{g\}. \tag{4}$$

Consider the best individual contained in the G_i in terms of the currently chosen fitness function: $z_i = \arg\max_{x \in G_i} f_i(x)$. Also consider a fitness difference in two sequential generations: $\Delta(f, i) = \frac{f(z_i) - f(z_{i-1})}{f(z_i)}, f \in A$.

We map the generations of individuals to the states of the environment. The state s_i corresponding to the generation G_i is a vector of criteria $f \in A$ sorted in descending order of the $\Delta(f, i)$ values:

$$s_i = \langle f_1, f_2, \dots f_{k+1} \rangle : \Delta(f_1, i) \geq \Delta(f_2, i) \geq \dots \Delta(f_{k+1}, i). \tag{5}$$

If $\Delta(f_a, i)$ is equal to $\Delta(f_b, i)$, then f_a, f_b are placed in some predefined order. For example, if the number of auxiliary criteria is $k = 2$ and in some generation G_i the inequality $\Delta(h_2, i) = \Delta(g, i) > \Delta(h_1, i)$ is satisfied, then the state may be represented by one of the following vectors: $s_i = \langle h_2, g, h_1 \rangle$ or $s_i = \langle g, h_2, h_1 \rangle$, depending on the initial agreement.

Finally, the reward function $R : S \times A \to \{0, \frac{1}{2}, 1\}$, that is calculated after choosing the criterion f_i in the state s_{i-1} and generating G_i, is defined. It depends on the difference between the target fitness of the best individuals at sequential generations and is the highest when the target fitness increases:

$$R(s_{i-1}, f_i) = \begin{cases} 1 & \text{if } g(z_i) - g(z_{i-1}) > 0 \\ \frac{1}{2} & \text{if } g(z_i) - g(z_{i-1}) = 0 \\ 0 & \text{if } g(z_i) - g(z_{i-1}) < 0 \end{cases} \tag{6}$$

3.2 Method Description

Now we can present the EA + RL method using the previously described notation. The pseudocode of the method is shown in Listing 1.

Listing 1 The EA + RL method.

```
1: Initialize the RL agent
2: Set the number of the current generation: $i \leftarrow 0$
3: Generate the initial generation $G_0$
4: while (EA termination condition is not reached) do
5:    Evaluate the state $s_i$ and pass it to the RL agent
6:    Get the FF for the next generation $f_{i+1}$ from the RL agent
7:    Evolve the next generation $G_{i+1}$
8:    Calculate the reward: $r \leftarrow R(s_i, f_{i+1})$ and pass it to the RL agent
9:    Increase the number of evolved generations: $i \leftarrow i + 1$
10: end while
```

The agent used in the method can encapsulate any reinforcement learning algorithm. We have implemented four different algorithms: two model-free algorithms with discounted reward (Q-learning [20] and Delayed Q-learning [19]), a model-free algorithm with average undiscounted reward (R-learning [14,18]) and a model-based algorithm with discounted reward (Dyna [20]). The experimental comparison of the algorithms can be found in Section 5.

4 Application to the H-IFF Problem

In this section a Hierarchical-if-and-only-if model problem is considered. We propose the EA + RL as a new method that can be used to solve this problem after multi-objectivization [13] without performing multicriteria optimization.

4.1 H-IFF Problem

Consider Hierarchical-if-and-only-if function, H-IFF [13]. Maximizing this function is a model problem used for testing genetic algorithms. The search space consists of bit strings of a fixed length l. The function takes a bit string $B = b_1 b_2 \dots b_l$ as an argument and interprets it as a binary tree of string blocks. The top of the tree is the input string itself, and the siblings are the left (B_L) and the right (B_R) halves of their parental blocks. The fitness function f is a sum of lengths of those blocks that consist of equally valued bits. Its recursive formula is given in (7). Notice that there are two possible optima in this problem. One optimum is a string of 0-bits and another is a string of 1-bits.

$$f(B) = \begin{cases} 1 & \text{if } |B| = 1, \text{ else} \\ |B| + f(B_L) + f(B_R) & \text{if } \forall i \{b_i = 0\} \text{ or } \forall i \{b_i = 1\} \\ f(B_L) + f(B_R) & \text{otherwise} \end{cases} \tag{7}$$

Various scalar optimization methods get stuck in local optima when solving this problem, but it can be reliably solved with multicriteria optimization methods after multi-objectivization [13]. These methods use two criteria f_0 and f_1 that consider blocks of only 0-bits or 1-bits respectively (8). The corresponding multicriteria problem is called MH-IFF.

$$f_n(B) = \begin{cases} 0 & \text{if } |B| = 1 \text{ and } b_1 \neq n, \text{ else} \\ 1 & \text{if } |B| = 1 \text{ and } b_1 = n, \text{ else} \\ |B| + f_n(B_L) + f_n(B_R) & \text{if } \forall i \{b_i = n\} \\ f_n(B_L) + f_n(B_R) & \text{otherwise} \end{cases} \tag{8}$$

4.2 EA + RL Solution

Consider the solution of the H-IFF problem with the proposed EA + RL method. To start with, represent the H-IFF problem as an optimization problem with auxiliary criteria (which will be referred to as H-IFFA).

Let the f be the target criterion: $g = f : W \to \mathbb{R}, W = \{B : |B| \leq l\}$. Then the criteria f_0 and f_1 can be considered as the auxiliary ones: $H = \{f_0, f_1\}, f_i : W \to \mathbb{R}$. The goal is to maximize f on the set of all possible bit strings of a fixed length l: $f(x) \to \max_{x \in X}, X = \{B : |B| = l\}$

Each of the solutions of the H-IFFA problem is the solution of the H-IFF problem and vice versa. The H-IFFA problem can be solved with the EA + RL method using the task defined by the equations (4), (5), (6). Thus we have a new solution for the H-IFF problem. The experimental results of applying this solution are presented in the next section.

5 Experiment

In this section the experiment results of applying the proposed method to solving the H-IFF problem are described. It is demonstrated that the EA + RL performance is better than the performance of the evolutionary algorithm alone. The proposed approach is also empirically compared with the multicriteria one.

5.1 Description of The Experiment

During the experiment the H-IFF and the H-IFFA problems where solved with the corresponding methods. The characteristics of the performed experiment are in accordance with the characteristics of the experiment taken in [13], in which the (M)H-IFF problem was solved. The length l of an individual was 64 bits. Notice that the optimal fitness of such individual is 448. 30 runs of each algorithm were performed. In each run 500000 fitness calculations were made. The statistics shown further is based on the best individuals from the last generations of each run.

The H-IFF problem was solved with two different types of evolutionary algorithms, such as a genetic algorithm (GA) and an evolution strategy (ES). The type of crossover operator used in the genetic algorithm was one-point crossover [16]. It was applied with the probability of 70%. The mutation operator flipped each bit of each individual with the probability of $2/l = 3.125\%$. The selection mechanism was implemented as tournament selection with the probability of 90%. In each generation of the genetic algorithm, five fittest individuals were preserved via elitism.

In the evolution strategy, the $(1 + m)$ selection was used, and the mutation operator flipped one randomly chosen bit of each individual. This variation is very likely to get stuck in a local optimum. One of the aims of the experiment was to show that the proposed method can effectively guide such algorithms preventing them from getting stuck.

The H-IFFA problem was solved with EA + RL method that controlled the previously described evolutionary algorithms. The parameter values used in various reinforcement learning algorithms are shown in Table 1. They were tuned manually so as the results of applying these algorithms to the H-IFFA solving were the highest. In the algorithms that explicitly require an exploration strategy the ε-greedy strategy [10, 20] was implemented.

5.2 Results of The Experiment

At first, the H-IFF was solved with a genetic algorithm without any controlling method. Then the H-IFFA problem was solved with a genetic algorithm where fitness functions were chosen randomly for each generation. Finally, H-IFFA was solved with the proposed method using various kinds of reinforcement learning algorithms. The EA + RL methods outperformed both the genetic algorithm and the genetic algorithm with randomly chosen fitness functions. The R-learning [14, 18] appeared to be the most efficient reinforcement learning algorithm for this problem.

Table 1: Parameter values of reinforcement learning methods.

Parameter	Description	Value	Parameter	Description	Value
Q-learning [20]			R-learning [18]		
α	learning rate	0.6	α	learning rate for reward ρ	0.5
γ	discount factor	0.1	β	learning rate for R-values	0.35
ε	exploration probability	0.01	ε	exploration probability	0.25
Delayed Q-learning [19]			Dyna [20]		
m	update period	5	k	number of updates	20
γ	discount factor	0.1	γ	discount factor	0.1
ϵ	bonus reward	0.2	ε	exploration probability	0.01

The results of the described runs are shown in Table 2. The results obtained using the proposed method are highlighted. The notation used to represent them is as follows. The "EA" in the EA + RL method is replaced with the abbreviation of the actual evolutionary algorithm used. The same operation is made with "RL". For example, the EA + RL method that use the genetic algorithm controlled by the Q-learning is denoted by *GA + Q-learning*.

There are also the results of solving H-IFF with some scalar optimization methods and MH-IFF with multicriteria methods in Table 2. They are taken from the article [13] that proposes multiobjectivization for efficient solving of scalar optimization problems. Compare this results with the ones obtained using the EA + RL methods. The performance of the GA(ES) + R-learning method is the same as of the PESA [12] multicriteria optimization method. Both of the methods find the ideal individuals in all runs which is the best possible performance. Most of the EA + RL methods outperform the other multicriteria method called PAES [11] and all of them outperform scalar optimization methods such as SHC [13] and DCGA [15].

Table 2: Results of solving (M)H-IFF(A) sorted by descending mean fitness.

Algorithm	Best fitness	Mean fitness	σ
(1+10) ES + R-learning	448	448.00	0.00
GA + R-learning	448	448.00	0.00
PESA	448	448.00	0.00
GA + Q-learning	448	435.61	32.94
GA + Dyna	448	433.07	38.07
PAES	448	418.13	50.68
GA + Delayed Q-learning	448	397.18	49.16
GA + Random-FF-chooser	384	354.67	29.24
DCGA	448	323.93	26.54
GA	384	304.53	27.55
SHC	336	267.47	29.46
(1+10) ES	228	189.87	17.21

At the second stage of the experiment we investigated how using of reinforcement learning can improve evolution strategy. Different evolution strategies were implemented to solve the H-IFF problem. Each strategy was also used in the corresponding ES + R-learning algorithm. The proposed algorithm significantly outperforms considered evolution strategies. The fitness values of the individuals evolved with the ES + R-learning method are approximately twice better than the ones obtained with the conventional evolution strategy. The results are shown in Table 3.

Table 3: Results of solving H-IFF(A) with evolution strategy and R-learning.

Algorithm	Best fitness	Mean fitness	σ
(1+1) ES + R-learning	448	403.49	59.48
(1+1) ES	188	167.07	11.98
(1+5) ES + R-learning	448	448.00	0.00
(1+5) ES	216	179.07	16.99
(1+10) ES + R-learning	448	448.00	0.00
(1+10) ES	228	189.87	17.21

6 Conclusion

In this paper, we formulated the scalar optimization problem with auxiliary criteria and proposed the EA + RL method that solves it using an evolutionary algorithm guided by reinforcement learning. No prior knowledge of the auxiliary criteria was used.

The method allows to increase the efficiency of scalar optimization, which is confirmed by the experimental results. Using of the proposed method prevents getting stuck in local optima of the H-IFF function. What is more, it doubles mean fitness obtained with an evolution strategy.

It is shown that reinforcement learning applied to on-the-fly fitness function control allows to increase fitness obtained during the fixed number of generations. This is a qualitatively new result, since reinforcement learning has not been applied to fitness function control in evolutionary algorithms yet [7].

References

[1] Afanasyeva, A., Buzdalov, M., Choosing Best Fitness Function with Reinforcement Learning. In *Proceedings of the Tenth International Conference on Machine Learning and Applications ICMLA 2011*, Honolulu, HI, USA, 18-21 December 2011. IEEE Computer Society, 2011. Vol.2, pp. 354-357.

[2] Alpaydin, E., *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, MIT Press, 2004

[3] Azuma, E., Shimada, T., Takadama, K., Sato, H., Hattori, K., The Biased Multi-Objective Optimization using the Reference Point: Toward the industrial logistics network. In *Proceedings of the Tenth International Conference on Machine Learning and Applications ICMLA 2011*, Honolulu, HI, USA, 18–21 December 2011. IEEE Computer Society, 2011. Vol.2, pp. 27-30.

[4] Buzdalov, M., Generation of Tests for Programming Challenge Tasks Using Evolution Algorithms. In *Proceedings of the 2011 GECCO Conference Companion on Genetic and Evolutionary Computation*, New York, US, ACM, 2011, pp. 763-766.

[5] Deb, K., *Multi-objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, 2001

[6] Ehrgott, M., *Multicriteria Optimization*, Springer, 2005

[7] Eiben, A. E., Horvath, M., Kowalczyk, W., Schut, M. C, Reinforcement learning for online control of evolutionary algorithms. In *Proceedings of the 4th international conference on Engineering self-organising systems ESOA'06*, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 151-160.

[8] Eiben, A., Michalewicz, Z., Schoenauer, M., Smith, J., Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, Lobo F., Lima C., Michalewicz Z., Eds. Springer Berlin / Heidelberg, 2007, vol. 54, pp. 19-46.

[9] Gosavi, A., Reinforcement Learning: A Tutorial Survey and Recent Advances, *INFORMS Journal on Computing*, Vol.21, No.2, 2009, pp. 178-192.

[10] Kaelbling, L. P., Littman, M. L., Moore, A. W., Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research*, Vol.4, 1996, pp. 237-285.

[11] Knowles, J. D., Corne, D. W., Approximating the nondominated front using the Pareto archived evolution strategy, *Evolutionary Computation*, Vol.2, No.8, 2000, pp. 149-172.

[12] Knowles, J. D., Corne, D. W., The Pareto-envelope based selection algorithm for multiobjective optimization. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature PPSN VI*, Berlin, Springer-Verlag, 2000, pp. 839-848.

[13] Knowles, J. D., Watson, R. A., Corne, D. W., Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization EMO'01*, London, UK, Springer-Verlag, 2001, pp. 269-283.

[14] Mahadevan, S., *Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results*, 1996

[15] Mahfoud, S. W., *Niching methods for genetic algorithms*, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995

[16] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, Cambidge, MA, 1996

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

[18] Schwartz, A., A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, 1993, pp. 298-305.

[19] Strehl, A. L., Li, L., Wiewora, E., Langford, J., Littman, M. L., PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning ICML'06*, 2006, pp. 881-888.

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