Selecting Evolutionary Operators using Reinforcement Learning: Initial Explorations

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

Vladislav Kononov
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
kononov@rain.ifmo.ru

Maxim Buzdalov
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia
mbuzdalov@gmail.com

ABSTRACT

In evolutionary optimization, it is important to use efficient evolutionary operators, such as mutation and crossover. But it is often difficult to decide, which operator should be used when solving a specific optimization problem. So an automatic approach is needed. We propose an adaptive method of selecting evolutionary operators, which takes a set of possible operators as input and learns what operators are efficient for the considered problem. One evolutionary algorithm run should be enough for both learning and obtaining suitable performance. The proposed EA+RL(O) method is based on reinforcement learning. We test it by solving H-IFF and Travelling Salesman optimization problems. The obtained results show that the proposed method significantly outperforms random selection, since it manages to select efficient evolutionary operators and ignore inefficient ones.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

evolutionary algorithms; parameter control

1. INTRODUCTION

Evolutionary operators such as mutation and crossover are used to generate offsprings. Operators influence the performance of an algorithm a lot [8]. Choosing the best operator manually is resource and time consuming, so automation is needed. There are different techniques of adjusting evolutionary algorithms [4], including methods for adjusting and modifying of operators, but there is no the most efficient one. Therefore, a new method of adjusting evolutionary algorithm by choosing evolutionary operators is of interest.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2605681.

Reinforcement learning is usually used to adjust some numerical parameters of evolutionary algorithms, such as mutation rate or population size [3, 9, 11]. To the best of our knowledge, there is only one method of selecting operators with reinforcement learning [10]. One of the most important differences between the method we propose and this one is that we apply the selected operator to the whole population, while in the method described in [10] an operator is chosen separately for each individual. We also applied simpler definitions of states and reward, which are needed for reinforcement learning.

In our previous works, we proposed the EA+RL method of extra fitness function selection based on reinforcement learning [2]. The method was shown to be efficient, both empirically (for a number of problems, including a real-world application [1]) and theoretically (for a model OneMax problem [2]). Thus in this paper we propose a method of evolutionary operators selection based on our previous ideas, which seems to be promising.

2. METHOD DESCRIPTION

It is assumed that there is a finite set of evolutionary operators consisting of mutation and crossover operators. It is unknown which operators are the most efficient ones, this information should be learned by the reinforcement learning agent (an efficient operator allows to optimize the fitness function in a less number of generations). The agent chooses an operator and the next population of the evolutionary algorithm is generated. Then an *immediate* numerical *reward*, as well as some representation of the algorithm *state*, are returned to the agent. The agent updates its estimations with the new values and the process repeats.

The scheme of the described method is shown in Fig. 1. The proposed method is similar to the previously proposed EA+RL, but here an evolutionary operator is selected instead of an extra fitness function. Thus let us call the newly proposed method EA+RL(O), where "O" stands for "operator".

The reward is calculated as the difference of the best individual fitness in a two consequent generations of the evolutionary algorithm. We used different states definitions for different problems, they are described later in the corresponding sections.

In reinforcement learning algorithms, the agent learns to perform such actions, that maximize the total reward, which is proportional to the sum of all immediate rewards [5, 13]. So in the proposed method the operators that maximize the fitness function should be selected eventually. In other

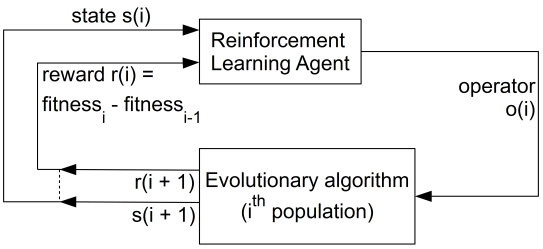

Figure 1: Scheme of the proposed EA+RL(O) method

words, the most efficient operators should be selected, since the goal of an evolutionary algorithm is the fitness function optimization.

3. EXPERIMENT 1: H-IFF OPTIMIZATION PROBLEM

In this section the first problem solved using the proposed method is described. It is a model problem called Hierarchical-if-and-only-if function (H-IFF), which is often used to test genetic algorithms [6, 14].

3.1 Problem Description

Consider the H-IFF optimization problem. 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 root 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 below. 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.

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

3.2 State definition

Consider a string block $B_{\max}$ that consists of equal valued bits (all zeros or all ones) and has the maximal length among all other such blocks in the current population. The state of the evolutionary algorithm used in the reinforcement learning algorithm is calculated as $\log_2 B_{\max}$.

3.3 Evolutionary Operators

The following operators are used in the evolutionary algorithm optimizing H-IFF:

- inversion mutation: a one random bit of an individual is inverted;
- tail inversion mutation: all the bits after a random position in an individual are inverted;
- obstructive mutation: an individual is replaced with the $010101 \dots 01$ string;
- two-point crossover: a segment of a random length is chosen at random, then the corresponding bits of parents are swapped;
- hybrid crossover: each bit of the first parent are replaced with the corresponding bit of the second parent with some probability (we used probability of 0.5).

Applying the *obstructive* mutation operator decreases the fitness function. This operator is included in the set of operators in order to demonstrate that the proposed EA+RL(O) method is able to ignore inefficient operators. The inversion mutation is expected to be less powerful than the tail inversion one, because the former inverts just one bit. Thus it should be checked that EA+RL(O) selects inversion mutation less frequently.

3.4 Experiment Description

During the experiment, the EA+RL(O) algorithm was run 101 times with the obstructive operator included, as well as with the obstructive operator excluded for the same number of times. An evolutionary algorithm with a random operator selection was also run for 101 times in order to compare it with the proposed method. The length of an individual was 64 bits, hence the maximal possible fitness value was 448. In each run, the evolutionary algorithm was stopped when the maximal possible fitness value was reached. There were 100 individuals in a generation. The probability of applying an operator was 100% in all cases. To perform a crossover, the tournament selection of one tour was used, the probability of selecting the fitter individual was 90%. The Q-learning algorithm with softmax and ε-greedy ($\varepsilon = 0.1$) exploration strategies [13] was used as the reinforcement learning algorithm.

3.5 Experiment Results

The average number of generations needed to obtain the maximal possible value of H-IFF function using considered algorithms is shown in Table 1. One can conclude, that using EA+RL(O) allows to maximize H-IFF in less number of generations than a random strategy does. According to the one-way ANOVA test we have performed [12], p-value for the EA+RL(O) and random selection was less than 5×10^{-4}, so they are statistically distinguishable.

It is also important to notice that EA+RL(O) successfully handles the obstructive operator case. The number of generations needed to optimize H-IFF in the presence of the obstructive operator is comparable to the number of generations needed when there is no obstructive operator.

Table 1: Average number of generations needed to obtain the maximal possible H-IFF value

Algorithm	Generations in average	Deviation
Without the obstructive operator		
EA+RL(O), softmax	801.8	298.1
EA+RL(O), ε-greedy	850.0	292.4
Random operator selection	1295.7	454.7
With the obstructive operator		
EA+RL(O), softmax	863.8	277.1
EA+RL(O), ε-greedy	890.6	308.0
Random operator selection	1654.2	478.8

In Fig. 2 the number of times when the operator was selected is shown for each operator. As expected, EA+RL(O) selects the obstructive operator for the least number of times.

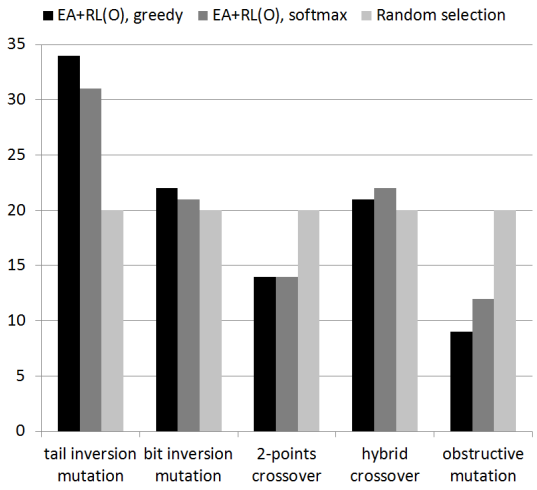

Figure 2: Number of times each operator was selected, %

It also selects the tail inversion mutation more often than the less powerful inversion mutation. Random selection chooses each operator equiprobably, which leads to worse performance of the evolutionary algorithm.

4. EXPERIMENT 2: TRAVELLING SALESMAN PROBLEM

Consider another optimization problem. Let us test the proposed EA+RL(O) method by solving the Travelling Salesman Problem (TSP), which is NP-hard [7]. We consider it as a problem of finding a Hamiltonian cycle with minimal weight in a fully connected weighted graph. Therefore this is a minimization problem.

4.1 State Definition

The state used in the reinforcement learning algorithm is represented as a vector of values, those are listed below.

- Population number interval. Numbers of generations are split in four intervals: $[0 \dots 400)$, $[400 \dots 1600)$, $[1600 \dots 3200)$, $[3200 \dots 6400)$ and the number of a current interval is put in the first position of the vector.
- Fitness function interval. The fitness of the best individual in the current population is divided by the fitness of the best individual in the initial population. This value can belong to a one of the following intervals: $[0, 0.2)$, $[0.2, 0.5)$, $[0.5, 0.8)$, $[0.8, 1]$. The number of the corresponding interval is put in the second position of the vector.

Notice that, if we replace the numerical values mentioned above with parameters, this state is problem-independent, thus can be used for solving any problem with EA+RL(O) method. Concept of this state is close to the one proposed in [10].

4.2 Evolutionary operators

In our approach, an individual is a correct permutation of vertices. All the considered operators preserve the correctness. The operators are as follows:

- inversion mutation: the order of vertices is inverted starting from a randomly chosen position;
- shuffle mutation: vertices are shuffled starting from a randomly chosen position;
- swapping mutation: two randomly chosen vertices are swapped;
- one-point match crossover with repair [7];
- two-point match crossover with repair [7].

4.3 Experiment Description

Five fully connected graphs of 80 vertices with edge weights from 0 to 100 were randomly generated. Then both an evolutionary algorithm with random operator selection and EA+RL(O) were run on these graphs for 100 times each. The algorithms were stopped after 6400 generations were evolved. There were 100 individuals in a generation. The probability of applying an operator was 100% in all cases. To perform a crossover, the tournament selection of one tour was used, the probability of selecting the fitter individual was 90%. As the reinforcement learning algorithm, the Q-learning algorithm with softmax and ε-greedy ($\varepsilon = 0.1$) exploration strategies was used [13].

4.4 Experiment Results

In Table 2 the best fitness in the final generation averaged over all the algorithm runs is shown. It can be seen that EA+RL(O) outperforms the evolutionary algorithm with random selection of operators. According to the one-way ANOVA test we performed [12], p-value for the EA+RL(O) and random selection is less than 5×10^{-4}, so they are statistically distinguishable.

Table 2: Average best fitness of TSP obtained in the final generation

Algorithm	Best fitness in average	Deviation
EA+RL(O), ε-greedy	1304.2	73.9
EA+RL(O), softmax	1323.9	69.8
Random operator selection	1436.0	68.4

In Fig. 3 the number of times when the operator was selected is shown for each operator. It can be assumed that the proposed EA+RL(O) algorithm selects the efficient operators more often than less efficient ones, since it outperforms random selection. Random selection chooses each operator equiprobably, which leads to worse performance of the evolutionary algorithm.

In Fig. 4 fitness optimization until the 9600 generations limit is shown. One can conclude that EA+RL(O) outperforms random selection during almost all optimization process.

5. CONCLUSION

A new EA+RL(O) method of adaptive selection between evolutionary operators using reinforcement learning is proposed. The method is aimed to enhance the performance of an evolutionary algorithm by selecting efficient operators from a set of ones. EA+RL(O) is based on the previously

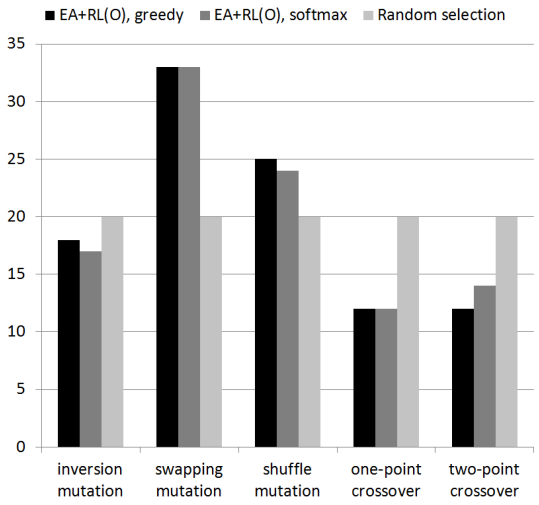

Figure 3: Number of times each operator was selected, %

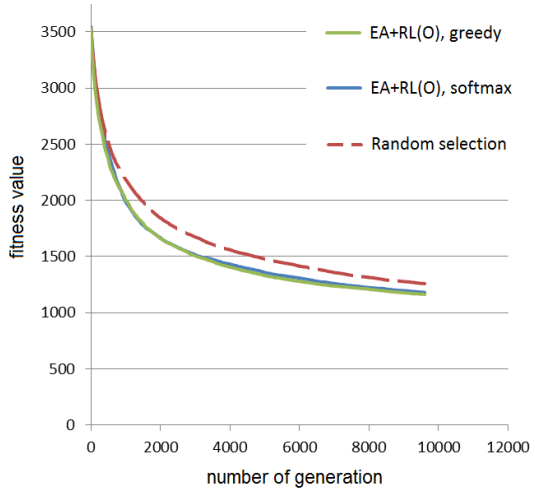

Figure 4: TSP fitness minimization

proposed EA+RL method used for selection between fitness functions. The proposed method was tested on two optimization problems. In both cases, it significantly outperformed random selection. Future work includes testing the proposed method on a wider range of problems, as well as obtaining its runtime analysis. It is also important to compare it with some other methods of evolutionary algorithms adjusting.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] M. Buzdalov, A. Buzdalova, and I. Petrova. Generation of Tests for Programming Challenge Tasks Using Multi-Objective Optimization. In C. Blum and E. Alba, editors, *GECCO (Companion)*, pages 1655–1658. ACM, 2013.

[2] M. Buzdalov, A. Buzdalova, and A. Shalyto. A First Step towards the Runtime Analysis of Evolutionary Algorithm Adjusted with Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning and Applications*, volume 1, pages 203–208. IEEE Computer Society, 2013.

[3] 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 ESOA'06*, pages 151–160. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 19–46. 2007.

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[6] 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*, EMO '01, pages 269–283, London, UK, 2001. Springer-Verlag.

[7] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artif. Intell. Rev.*, 13(2):129–170, Apr. 1999.

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

[9] 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*, pages 151–156. IEEE, 2002.

[10] J. E. Pettinger and R. M. Everson. Controlling genetic algorithms with reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, page 692, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[11] Y. Sakurai, K. Takada, T. Kawabe, and S. Tsuruta. A method to control parameters of evolutionary algorithms by using reinforcement learning. In *Signal-Image Technology and Internet-Based Systems (SITIS), 2010 Sixth International Conference on*, pages 74–79, 2010.

[12] D. S. Soper. Analysis of variance (anova) calculator - one-way anova from summary data [software], 2014.

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

[14] R. A. Watson and J. B. Pollack. Analysis of recombinative algorithms on a hierarchical building-block problem. in foundations of genetic algorithms. *Foundations of Genetic Algorithms (FOGA)*, 2000.