COMPARATIVE STUDY OF METHODS FOR COMBINING ARTIFICIAL IMMUNE SYSTEMS AND RANDOM LOCAL SEARCH

Nina Bulanova, Arina Buzdalova and Vladimir Parfenov

ITMO University
Computer Technologies Laboratory
49 Kronverkskiy prosp., Saint-Petersburg, Russia
ninasbulanova@gmail.com, abuzdalova@gmail.com, parfenov@mail.ifmo.ru

Abstract: *Construction of a new efficient algorithm from several simple ones is an actively researched area and it includes such approaches as operator selection with reinforcement learning, hybridisation, memetic algorithms etc. Artificial immune systems and random local search have remarkable differences in the structure of mutation operators, so they demonstrate different behaviour during optimization process. Combination of mutation operators from these two groups of algorithms may produce a new algorithm which is efficient on fixed budget and reaches optimum within reasonable time bounds. To the best of our knowledge, this is the first attempt to compare different approaches which combine AIS with other optimization heuristic.*

Keywords: *hybridisation, reinforcement learning, memetic algorithms, BCA, CLONALG, AIS, RLS*

1 Introduction

During an optimization process, different algorithms may demonstrate different behaviour. For example, artificial immune systems (AIS) are efficient at the early stages of optimization, but are outperformed by random local search (RLS) as optimization progresses [7]. Moreover, these two groups of algorithms have noticeable differences in mutation operators: AIS mutations are global, while the flip one bit mutation used in RLS is local. Therefore, investigating different methods that combine AIS algorithms and RLS may be an interesting subject for a comparative study.

Design and analysis of methods combining mutation operators in one algorithm is an actively developing research area. For example, in memetic algorithms, combination of evolutionary algorithms and local search is used [8]. Local search operators in memetic algorithms may be selected adaptively from the predetermined set of operators [13, 10]. Hyper-heuristics are worth mentioning as well [2]. A hyper-heuristic may be described as a new heuristic created by combining and adapting several simpler heuristics.

Advantages of different algorithms may also be combined by *hybridisation*. In paper [4], hybridisation was used to combine mutation operators of AIS and EAs in one algorithm. In this approach, selection between mutation operators is performed with constant probability. Therefore, this approach does not make adjustments during optimization. We propose a method of fitness-dependent hybridisation of AIS and RLS, where probability of selection of mutation operator depends on the fitness value of a current individual.

Selection between several mutation operators can also be done adaptively with reinforcement learning (RL) [11, 3]. RL agent takes a set of possible operators as input and learns what operators are efficient for the considered problem at the current stage of optimization. To the best of our knowledge, the selection between AIS and RLS mutation operators with reinforcement learning has never been studied before.

The rest of the paper is organized as follows. Firstly, we describe some approaches of operator combination: reinforcement learning, hybridisation and memetic algorithms. Secondly, we take the best performing algorithms from the previous sections and conduct a comparative research of these methods. Finally, some conclusions about the considered methods are given.

2 Preliminaries

In this section we describe the model problems, present a general algorithmic scheme used in our research and give general settings for the experiments described in the next sections. We also compare RLS and AIS algorithms which are combined further in the paper.

2.1 Considered Optimization Problems

We considered two problems: OneMax and LeadingOnes. OneMax and LeadingOnes are well-known model problems in the field of evolutionary computation and artificial immune systems. What is more, these problems have different computational complexities [9, 1, 15]. For both problems, the search space consists of all possible bit vectors of length n. One needs to maximize the fitness function. In the OneMax problem the fitness function of a bit vector is the number of bits set to one. In the LeadingOnes problem the fitness function is the length of the maximal prefix consisting of bits set to one.

2.2 Algorithmic Scheme

The general scheme of algorithms considered in this paper is (1+1) Algorithm according to a similar scheme presented in [4]. At every step of optimization this algorithm has one individual in a generation and creates a single offspring using some mutation operator. This offspring can be selected for the next generation if its fitness function value is greater than or equal to the fitness function value of the parent. Algorithm stops as soon as it reaches the optimum.

Depending on the mutation operator, (1+1) Algorithm may represent RLS or an AIS algorithm. For the AIS algorithms, we consider the Clonal Selection Algorithm (CLONALG) and the B-cell Algorithm (BCA) [15].

2.3 General Experimental Settings

Each algorithm was run 1000 times, and then the results were averaged. Since a generation is always comprised of a single individual, the algorithms could be compared based on the number of generations needed to reach the problem optimum. The number of fitness function evaluations was limited by 10^6. For both problems, the search space consists of all bit vectors of length n. Due to different computational complexities of the considered problems for the selected algorithms, it was decided to use individuals of length $n = 1000$ for the OneMax problem and individuals of length $n = 100$ for the LeadingOnes problem. In all the figures presented in the paper, the logarithmic scales with the base of 10 are used for the both axes. Using this scale we can show the beginning of the optimization process in more detail.

2.4 Separate Performance of BCA, CLONALG and RLS

The plots in Fig. 2 demonstrate algorithms BCA, CLONALG and RLS solving OneMax and LeadingOnes problems. Overall, in the beginning the average fitness value is increasing faster with the help of AIS algorithms, whereas RLS demonstrates more stable increase during the whole optimization process. At the same time, RLS is less efficient than AIS algorithms in the beginning.

Among AIS algorithms, CLONALG is the most efficient one on the OneMax problem. In the beginning, CLONALG achieves the large increase of fitness by applying global mutations. However, after a number of fitness evaluations the efficiency of CLONALG decreases compared to RLS. As it can be seen from Table 1, RLS is the most efficient algorithm among all the considered algorithms.

Another AIS algorithm BCA increases the value of the fitness function very fast on LeadingOnes problem. In the beginning, the AIS algorithms are more efficient than RLS. At the same time, unlike in the case of the OneMax problem, in the LeadingOnes problem CLONALG increases fitness very slowly, whereas BCA shows steady growth of fitness. This growth can be explained by the fact that BCA mutation has block structure and in LeadingOnes a block of ones should be obtained. There is an interval from 10^2 to almost 10^3 fitness evaluations when BCA is better than CLONALG and RLS. It can also be observed from Table 1 that RLS needs less fitness evaluations to optimize LeadingOnes than any other considered algorithms.

Based on the presented separate results, we assume that it is best to combine RLS with CLONALG while solving OneMax, whereas for the LeadingOnes problem RLS should be combined with BCA.

Table 1: Fitness evaluations needed to optimize OneMax and LeadingOnes with simple algortihms

	OneMax		LeadingOnes	
Algorithms	Generations	Deviation	Generations	Deviation
RLS	6.76×10^3	1.27×10^3	5.01×10^3	8.89×10^2
CLONALG	1.68×10^4	3.38×10^3	6.23×10^5	1.73×10^5
BCA	–	–	2.64×10^4	1.12×10^4

3 Considered Methods of Combination

In this section we describe the considered methods and present separate experiment results for each method.

3.1 Selection of Operators with Reinforcement Learning

Operator selection can be done with reinforcement learning (RL), including multi-armed bandits [5, 6] and Q-learning [3]. In such methods, the RL agent selects operators to be used in an optimization algorithm. Based on the information returned from the optimization algorithm, the agent adapts its strategy and selects the next operator to be used. In this section we select between AIS and RLS operators using RL and experimentally analyse this approach.

3.1.1 RL(O): Method Description

Consider RL(O) operator selection method presented in Fig. 1 [3]. The agent selects an operator to be used in the current generation. The optimization algorithm forms the next generation using this operator and returns some numerical reward to the agent. The reward is based on the growth of the fitness function. The agent learns to select efficient operators that maximize the reward.

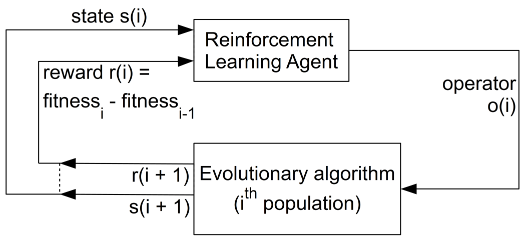

state s(i)
Reinforcement Learning Agent
reward r(i) = fitness_i - fitness_i-1
operator o(i)
r(i + 1)
s(i + 1)
Evolutionary algorithm (i^th population)

Figure 1: RL(O) method: selection of operators with reinforcement learning

We implemented RL(O) using Q-learning as the RL algorithm, and the (1+1) Algorithm as the optimization algorithm. The pseudocode of the resulting method is shown in Algorithm 1. As mutation operators, we used AIS mutation operators from CLONALG and BCA, as well as the flip one bit mutation operator from RLS.

Algorithm 1 RL(O) Algorithm

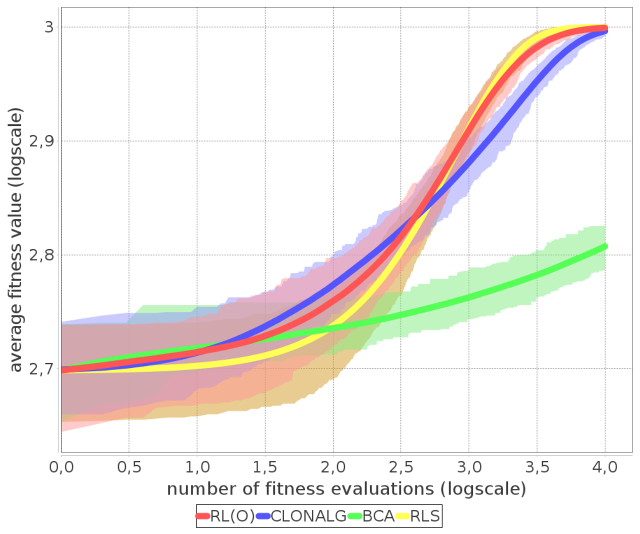
```
1: Current individual $x \leftarrow$ a randomly generated individual
2: Define set of operators $H = \{$CLONALG mutation, BCA mutation, RLS mutation$\}$
3: $Q(h) \leftarrow 0$ for each action $h \in H$
4: while (stopping criterion is not reached) do
5:    Store individual $y \leftarrow x$
6:    Select operator $h$: $Q(h) = \max_{h' \in H} Q(h')$
7:    Individual $x' \leftarrow$ mutate $x$ using the selected operator $h$
8:    if $f(x') \geq f(x)$ then
9:       $x \leftarrow x'$
10:   end if
11:   $r \leftarrow f(x) - f(y)$
12:   $Q(h) \leftarrow Q(h) + \alpha(r + \max_{h' \in H} Q(h') - Q(h))$
13: end while
```

3.1.2 RL(O): Experiment Results

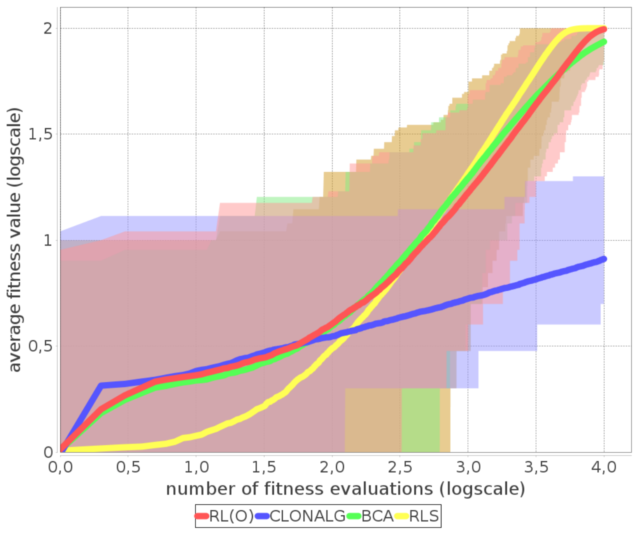
All experiments were conducted with the following parameters of reinforcement learning: the learning rate $\alpha = 0.3$, the discount factor $\gamma = 0.02$ and the probability of exploration $\alpha = 0.1$.

The plots shown in Fig. 2 illustrate optimization of OneMax and LeadingOnes by RLS, CLONALG, BCA separately and RL(O) which selects one of the corresponding operators. The computational budget of up to 10000 fitness function evaluations is used. In these plots, as in all the following plots, the dark coloured lines are the average fitness function values and the light coloured areas around these lines are the space between maximum and minimum values of the fitness function reached in the corresponding step during 1000 runs.

Runtime results for this method are demonstrated in Table 2. As we can see from Table 2, Table 1 and plots shown in Fig. 2 this method has average efficiency. It is always slightly worse than the best separate algorithm



(a)



(b)

Figure 2: CLONALG, BCA, RLS and RL(O) solving OneMax (a) and LeadingOnes (b) problems.

in every stage of optimisation due to the fact the RL agent needs some time to learn which algorithm is the most efficient at the current stage of optimization.

3.2 Operator Combination with Hybridisation

Mutation operators can be applied according to a certain condition. The hybridisation approach proposed in the paper [4] selects one mutation operator with a fixed probability p or the other one with the probability of $1 - p$. Further we call this approach *constant probability hybridisation*.

3.2.1 Method of Fitness-Dependent Hybridisation

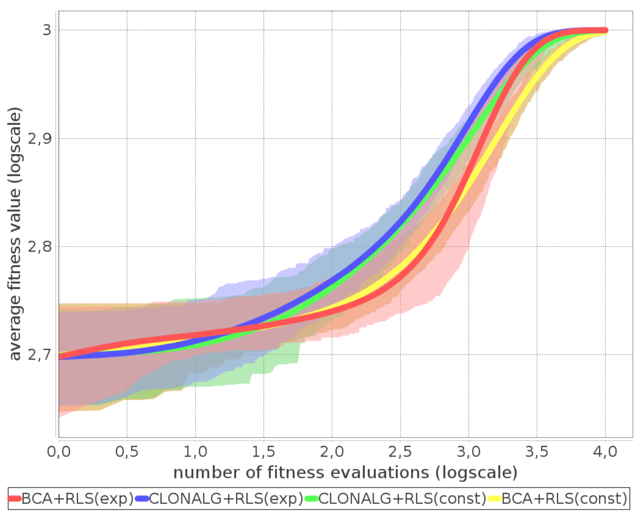
We proposed to use a hybrid algorithm which combines two mutation operators in one algorithm and selects one of them according to probability function which depends exponentially on the current fitness (see Algorithm 2). This approach helps shift preference between mutation operators from one operator to another during the optimization process. We call this hybridisation technique *fitness-dependent hybridisation*.

In this study, we were guided by the following idea. AIS algorithms demonstrate higher efficiency in the beginning of optimisation, so we can use mutation operators from AIS at the beginning and then gradually switch to EA mutation operators. To do this, the fitness-dependent probability function should be set to one in the beginning of optimization, this way we can maximize the number of calls to AIS mutation at this stage. Then the fitness-dependent probability function should decrease to a sufficiently small value in the end of optimization.

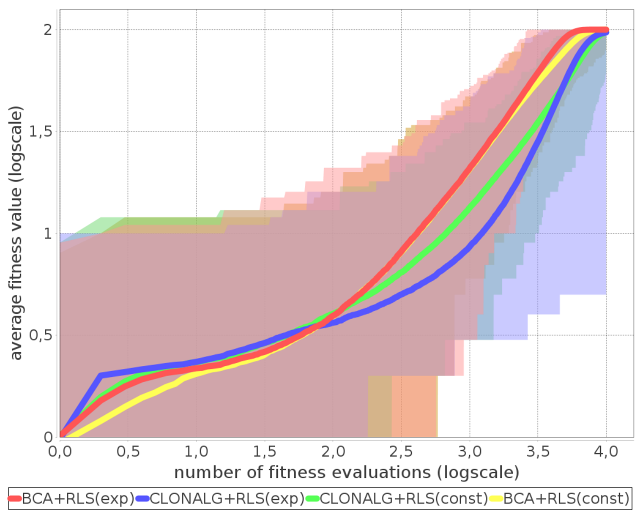
Algorithm 2 (1+1) Hybrid Algorithm

```
1: $x \leftarrow$ random bit string of length n
2: $v_0 \leftarrow f(x)/\max(f)$ normalized fitness of the initial individual, $max(f)$ — upper bound of fitness value for
   the considered problem
3: **while** (optimum is not found or limit of iterations is not reached) **do**
4:    $v \leftarrow f(x)/\max(f)$ normalized fitness of the current individual
5:    with probability of $n^{(-v+v_0)}$
6:       $x' \leftarrow Mutation1(x)$ **or**
7:    With probability of $1 - n^{(-v+v_0)}$
8:       $x' \leftarrow Mutation2(x)$
9:    **if** $f(x') \geq f(x)$ **then**
10:      $x \leftarrow x'$
11:   **end if**
12: **end while**
```

3.2.2 Hybridisation: Experiment Results



(a)



(b)

Figure 3: Hybrids solving OneMax (a), LeadingOnes (b) problems.

In accordance with the conclusions of Section 2.4, we make the assumption that for the OneMax problem it will be useful to combine CLONALG and RLS mutation operators. For the LeadingOnes problem, it can be concluded that hybridisation of BCA and RLS would be the best choice. Runtime results for hybrids are presented in Table 2. These results confirm our assumption and the claimed efficiency of fitness-dependent hybridisation method. As we can see, for the OneMax problem, fitness-dependent hybridisation of CLONALG and RLS is the most efficient. For LeadingOnes, the most efficient method is the fitness-dependent hybrid of BCA and RLS.

On fixed budget, we can see a similar situation (see Fig. 3). Constant probability hybrid algorithm with the fixed probability of selection of mutation operators is not as efficient as fitness-dependent probability hybrid which behaves more like AIS in the beginning of optimization. So we choose CLONALG+RLS(exp) for the general comparison of the combination methods (Section 4) on the OneMax problem and BCA+RLS(exp) on the LeadingOnes problem.

3.3 Combining AIS and Local Search in Memetic Algorithms

Memetic algorithms combine global and local search in one algorithm [14]. In each iteration, a memetic algorithm uses global search mutation and applies local search with some probability. We consider the algorithm where local search is called on each iteration, which is denoted as iterated local search.

3.3.1 Memetic Algorithms: Method Description

In Algorithm 3 we use AIS mutation operator (Mutation1) for global search and RLS mutation operator (Mutation2) for local search. When we make a comparison of the efficiency of algorithms based on the number of fitness function evaluations, we also consider the fitness calculations when the local search is called.

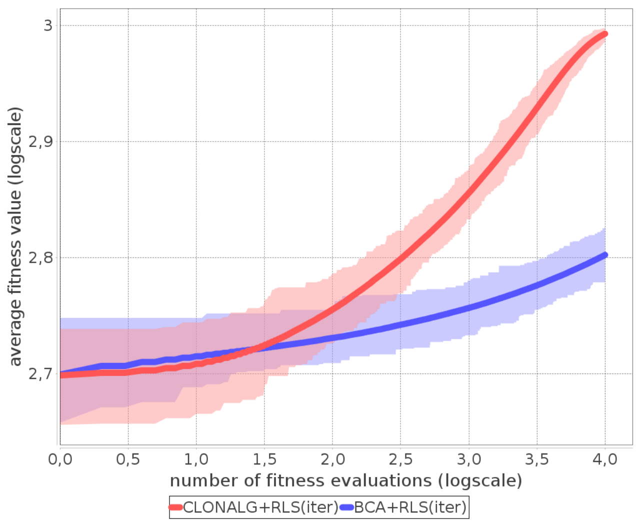
3.3.2 Memetic Algorithm: Experiment Results

In Table 2 the number of fitness function evaluations needed to reach the optimum by memetic algorithms is demonstrated. On the OneMax problem the memetic algorithm which uses BCA with local search still can not solve the problem during 10^6 function evaluations. Memetic algorithm with CLONALG is more efficient. On the LeadingOnes problem the situation is inversed, the memetic algorithm with BCA is more efficient than the memetic algorithm with CLONALG. This may be explained by the fact that mutation in BCA has block structure, and the value of LeadingOnes grows when a block of one-bits is obtained at a certain position.

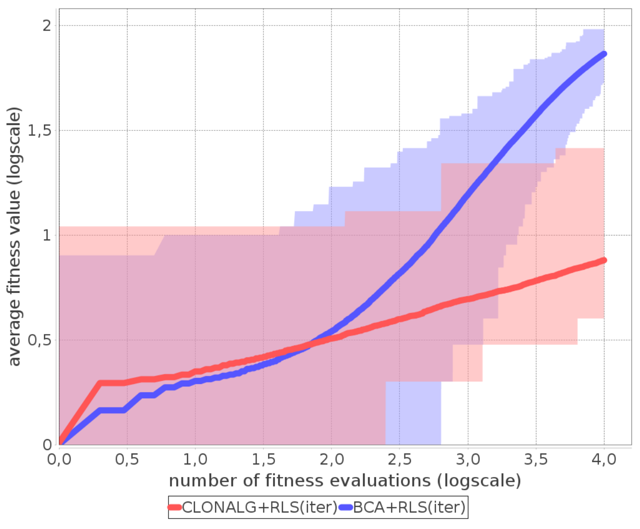
In Fig. 4 we present the results of applying iterated local search in memetic algorithm on fixed budget. In accordance with the conclusions of Section 2.4 and these plots, we choose the memetic algorithm with CLONALG on OneMax problem and the memetic algorithm with BCA on LeadingOnes problem for the general comparison of the combination methods (Section 4).

Algorithm 3 (1+1) Memetic Algorithm

```
1: $x \leftarrow$ random bit string of length n
2: while (optimum is not found or limit of iterations is not reached) do
3:     $x' \leftarrow Mutation1(x)$
4:     $x'' \leftarrow Mutation2(x')$
5:     if $f(x'') \geq f(x')$ then
6:         $x' \leftarrow x''$
7:     end if
8:     if $f(x') \geq f(x)$ then
9:         $x \leftarrow x'$
10:    end if
11: end while
```



(a)



(b)

Figure 4: Memetics solving OneMax (a), LeadingOnes (b) problems.

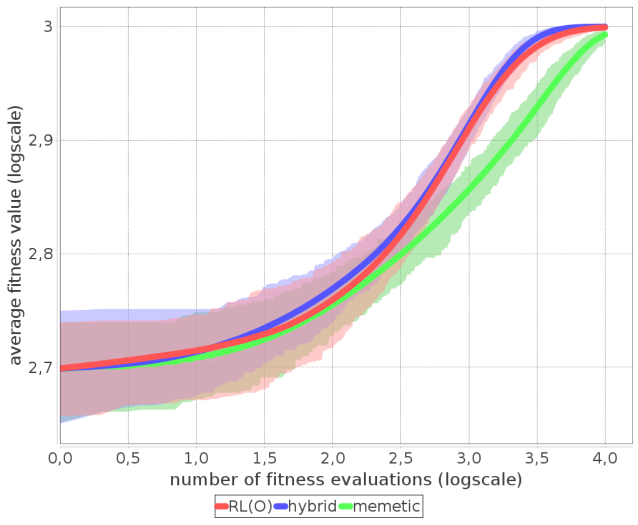
4 General Efficiency Comparison of The Combination Methods

For our comparative study we decided to take the best performing algorithm from each considered approach and then compare them with each other.

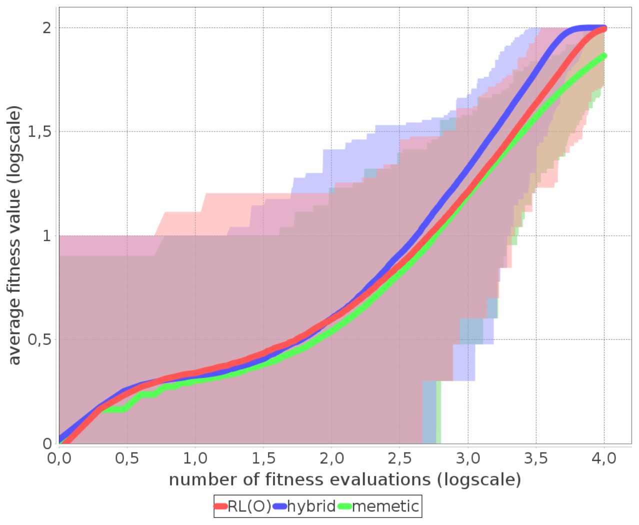
Firstly, we can observe from Fig. 5 that for both problems on fixed budget hybrid algorithms demonstrate the best efficiency. This can be explained by the fact that among other described methods hybridisation uses information about efficiency of different mutation operators on considered problems. It was noticed that AIS mutation operators are more efficient in the beginning of optimisation, and RLS is more efficient in the end when small local changes left to be made. The hybridisation approach explicitly uses this insight. Two remaining methods are not so specified. Reinforcement learning needs to discover efficiency of operators during optimisation. The memetic algorithm does not switch operators, so it uses both operators together, the efficient operator and the inefficient one at the current stage of optimization.

Secondly, in Table 2 the number of fitness function evaluations needed to optimize OneMax and LeadingOnes is presented. Standard deviation is given in the right column for each considered algorithm. As we can see, hybrids are still the best among other methods. Memetics demonstrate the worst results because they never stop using AIS mutation operator, including the cases when AIS mutation is inefficient like in the end of optimisation.

In order to investigate the results for statistical significance, the unpaired Wilcoxon test was applied with the level of significance of $\alpha = 0.05$. The numbers of generations needed to reach the optimum were used as input to the test. The test was performed with the stats::wilcox.test() function from the R language [12]. We applied Holm correction to compare each algorithm with the rest of algorithms. To verify the results for statistical significance, we applied the Wilcoxon rank sum test followed by the Holm-Bonferroni correction. Algorithms in every possible pair are statistically distinguishable from each other since the obtained p-values were less than 0.05.



(a)



(b)

Figure 5: Comparative study of Reinforcement Learning, Hybridisation and Memetics on OneMax (a) and LeadingOnes (b) problems.

Table 2: Number of fitness evaluations needed to optimize OneMax and LeadingOnes with hybrids, memetics and reinforcement learning

Description of Combination Methods			OneMax		LeadingOnes	
Method	Operators	Selection function	Generations	Deviation	Generations	Deviation
RL(O)	RLS,CLONALG, BCA	Q	1.14×10^4	2.29×10^3	8.16×10^3	1.53×10^3
hybrid	BCA,RLS	constant	1.34×10^4	2.58×10^3	6.89×10^3	1.24×10^3
hybrid	BCA,RLS	exponential	7.48×10^3	1.31×10^3	5.21×10^3	9.08×10^2
hybrid	CLONALG,RLS	constant	9.57×10^3	1.87×10^3	8.73×10^3	1.57×10^3
hybrid	CLONALG,RLS	exponential	6.91×10^3	1.32×10^3	7.58×10^3	2.96×10^3
memetic	BCA,RLS	iterated	–	–	3.83×10^4	1.34×10^4
memetic	CLONALG,RLS	iterated	1.91×10^4	3.50×10^3	1.12×10^6	3.44×10^5

5 Conclusion

We conducted comparative study of different methods of combination of mutation operators of AIS and RLS using OneMax and LeadingOnes problems. It was previously confirmed that AIS algorithms may be more efficient at the beginning of optimization, while RLS is able to solve an optimization problem in less number of fitness function evaluations. So, in our study, we could use mutation operators from AIS at the beginning of optimization and then gradually switch to RLS mutation operator. This knowledge turned to be essential while constructing an efficient algorithm from several mutation operators.

The hybridisation method demonstrated the best results. This may be explained by the fact that the hybridisation approach explicitly uses the above observation and applies selection between mutation operators according to fitness-dependent probability function. It seems that using AIS mutation is efficient only in a small period at the beginning of optimization, and further the probability of an AIS mutation should be reduced drastically. Therefore, the exponential probability function used in one of the considered hybrids demonstrated higher efficiency than the constant function.

Reinforcement learning showed slightly worse results. It had to learn about algorithm efficiency on different stages of optimisation, which is associated with some difficulties due to the fact that the efficiency of the considered algorithms changes during the optimization process. The memetic algorithm with iterated local search demonstrated the worst result among other methods because it had to use the considered operators together. All the obtained results were tested for statistical significance. These results may be useful for further research of combination of AIS with other search heuristics.

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

References

[1] Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In: Parallel Problem Solving from Nature – PPSN XI, no. 6238 in Lecture Notes in Computer Science, pp. 1–10. Springer (2010)

[2] Burke, E.K., Gendreau, M., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. JORS **64**(12), 1695–1724 (2013)

[3] Buzdalova, A., Kononov, V., Buzdalov, M.: Selecting evolutionary operators using reinforcement learning: Initial explorations. In: Proceedings of Genetic and Evolutionary Computation Conference (Companion), pp. 1033–1036 (2014)

[4] Corus, D., He, J., Jansen, T., Oliveto, P.S., Sudholt, D., Zarges, C.: On easiest functions for somatic contiguous hypermutations and standard bit mutations. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 1399–1406 (2015)

[5] Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In: Learning and Intelligent Optimization, Third International Conference, LION 3, Trento, Italy, January 14-18, 2009. Selected Papers, pp. 176–190 (2009)

[6] Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Ann. Math. Artif. Intell. **60**(1-2), 25–64 (2010)

[7] Jansen, T., Zarges, C.: Reevaluating immune-inspired hypermutations using the fixed budget perspective. IEEE Trans. Evolutionary Computation **18**(5), 674–688 (2014)

[8] Neri, F., Cotta, C., Moscato, P. (eds.): Handbook of Memetic Algorithms, *Studies in Computational Intelligence*, vol. 379. Springer (2012)

[9] Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing **4**(3), 281–293 (2007)

[10] Ong, Y., Lim, M., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man, and Cybernetics, Part B **36**(1), 141–152 (2006)

[11] Pettinger, J.E., Everson, R.M.: Controlling Genetic Algorithms with Reinforcement Learning. In: Proceedings of Genetic and Evolutionary Computation Conference, p. 692. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002)

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

[13] Smith, J.E.: Self-adaptative and coevolving memetic algorithms. In: Handbook of Memetic Algorithms, pp. 167–188 (2012)

[14] Sudholt, D.: Memetic algorithms with variable-depth search to overcome local optima. In: Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008, pp. 787–794 (2008)

[15] Zarges, C.: Theoretical foundations of artificial immune systems. Ph.D. thesis, Universität Dortmund (2011)