2014 13th International Conference on Machine Learning and Applications

A Switch-and-Restart Algorithm with Exponential Restart Strategy for Objective Selection and its Runtime Analysis

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

***Abstract*—There exist optimization problems with the target objective, which is to be optimized, and several extra objectives, which may or may not be helpful in the optimization process. This paper considers the case when it is possible to find an optimum of the target objective by optimizing either the target objective or a single extra objective.**

An algorithm is presented that uses a single instance of an underlying single-objective optimization algorithm to optimize different objectives at different iterations and restarts the optimization algorithm between optimizing different objectives. This algorithm has the expected running time of at most $4K \min_O T_O$ until an optimum of the target objective is found, where T_O is the expected running time of the underlying optimization algorithm to find an optimum of the target objective by optimizing the objective O. An impact of not using restarts between iterations is also discussed.

I. Introduction

Single-objective optimization can often benefit from multiple objectives [1]–[4]. Different approaches are known from the literature. Some researchers introduce additional objectives to escape from the plateaus [5]. Decomposition of the primary objective into several objectives also helps in many problems [2], [3], [6]. Additional objectives may also arise from the problem structure [7].

There is an open question: can the expected number of iterations when using extra objectives be smaller than the expected number of iterations without using extra objectives? The algorithm presented in this paper has the following property: if for at least one extra objective the target optimum can be found in an expected number of iterations that is asymptotically smaller than for the target objective itself, the same will hold for the presented algorithm.

The rest of the paper is structured as follows. Section II discusses the related work. In Section III, the switch-and-restart algorithm for objective selection is described with some basic runtime analysis. Section IV concentrates on the exponential restart strategy and gives stronger upper and lower bounds. In Section V, it is discussed what happens when one does not restart the underlying optimization algorithm when swtching between optimizing different objectives. Section VI concludes.

II. Related Work

In this section, some previous work is presented that concentrates on constructing and applying objectives other than the target objective to speed up optimization. The idea of exponential restarts, which is used in this paper, also appeared in some preceding works, although it addresses something different. Finally, a folklore idea of running many algorithms in parallel and choosing the best result is also discussed.

A. Multi-Objectivization and Helper-Objectives

Different approaches may be applied to a problem with the "original" objective, which can be called the *target* objective, and some extra objectives. The *multi-objectivization* approach is to optimize *all* extra objectives at once using a multi-objective optimization algorithm [3], [6]. One of the examples where this approach provably works is the H-IFF optimization problem [3]: one can naturally decompose it into two components and optimize them simultaneously, which leads to discovery of all optima of the original problem.

The *helper-objective* approach is to optimize simultaneously the target objective and some (not necessarily all, in some cases, only one is preferable) extra objectives, switching between them from time to time. The famous job-shop scheduling problem is a good example of a problem where it is possible to introduce helper-objectives based on the structure of the problem [4], [7].

B. The EA+RL Method

Multi-objectivization and helper-objective approaches are designed in the assumption that the extra objectives are crafted to help optimizing the target objective. However, this is not always true, especially when the extra objectives are generated automatically [8], or their properties are unknown. In fact, the extra objectives may support or obstruct the process of optimizing the target objective.

978-1-4799-7415-3/14 $31.00 © 2014 IEEE
DOI 10.1109/ICMLA.2014.27

The EA+RL method was developed to cope with such situations [9]. The idea of this method is to use a single-objective evolutionary optimization algorithm and switch between the objectives (which include the target one and the extra ones). To find the most suitable objective for the optimization, reinforcement learning algorithms are used [10]–[12]. In paper [9], the EA+RL method was introduced and experimentally evaluated on some simple benchmark problems. In [8], it was applied to the problem of worst-case execution time test generation. In papers [13], [14], first theoretical results about this method were presented. The EA+RL method can also be adapted to select the second objective for the helper-objective approach [8].

C. Exponential Restarts

It is not a very new idea to restart an iterative optimization algorithm after exponentially increasing number of iterations. For example, in [15] Jansen studied the effect of applying additive and multiplicative restarts (the latter is similar to the considered case) to simple evolutionary algorithms which optimize specific classes of functions. Another similar idea known in the literature is the iterative-deepening depth-first search [16] which, when run on a tree with fixed or bounded number of children per node, essentially visits an exponentially increasing number of nodes per iteration.

D. Algorithm Portfolios

The problem of objective selection, in some cases, can be treated as the problem of algorithm selection. This problem is not new [17], however, it was mainly applied to combinatorial problem solvers [18]. An algorithm portfolio approach [19] was applied recently to the field of evolutionary computation [20] and to continuous optimization [21]. These approaches, however, are rarely treated from the theoretical or worst-case point of view. What is more, the algorithms belonging to portfolios are optimizing the same function, which makes the progress of different algorithms observable and comparable, unlike the current research.

E. Common Sense: Running Algorithms in Parallel

It is a common idea to run several optimization algorithms in parallel. When one of them finds an optimum, all the algorithms are terminated. Let there be K algorithms, and let T_i be the running time of the i-th algorithm until an optimum is found, then the total running time of the algorithm family is $K \min_i T_i$. The same result actually holds for the expected total running time if the algorithms are randomized and T_i is the *expected* running time until an optimum is found.

This way of running several algorithms requires all of them to be loaded in the memory simultaneously. Sometimes, when K is large, or when an algorithm needs big amounts of memory, this behavior is undesirable. This paper addresses this issue — the switch-and-restart algorithm requires only one instance of the optimization algorithm to be loaded in the memory, while the total running time is only $O(1)$ times worse.

```
1:  function SARA(X, K)
2:      X — the optimization algorithm
3:      K — the number of objectives, target or extra
4:      A — the list of meta-iteration sizes
5:      O ← an instance of X
6:      j ← 1 — the meta-iteration number
7:      loop
8:          for i ← 1 ... K do
9:              INITIALIZE(O)
10:             SETOBJECTIVE(O, i)
11:             for t ← 1 ... A_j do
12:                 MAKEITERATION(O)
13:                 best ← BEST(O)
14:                 if ISTARGETOPTIMUM(best) then
15:                     return best
16:                 end if
17:             end for
18:         end for
19:         j ← j + 1
20:     end loop
21: end function
```

Fig. 1. The general scheme of the switch-and-restart algorithm

III. General Switch-and-Restart Algorithm

Running algorithms in parallel has a single drawback — it needs K instances of the optimization algorithm to be loaded into memory. This can be a major issue when K is large, or when the algorithm needs significant amount of memory to work.

One of the possible solutions is to have a single instance of the optimization algorithm and to optimize different objectives for different time slots. For the sake of symmetry, it makes sense to run optimization in rounds, or *meta-iterations*. During a single meta-iteration with the number i, the algorithm will try to optimize each objective for a certain number of iterations, say, A_i.

Between rounds of optimization of different objectives, the optimization algorithm may or may not be reset to some initial, probably randomized, state. This is a non-trivial question whether restart should be done. The consequences of this choice will be discussed late. The algorithm being described in this section *does* restart the optimization algorithm.

The described algorithm, which is called SaRA for "switch-and-restart algorithm" is presented at Fig. 1.

A. Runtime Analysis of SaRA

It is assumed the *running time* of the SaRA algorithm to be equal to the number of iterations of the underlying optimization algorithm. This can be quite a natural measure if each iteration of the underlying optimization algorithm takes roughtly the same time.

Theorem 1. *Let K be the number of objectives, O be an arbitrary objective, and $P_O(x)$ be the probability for the underlying optimization algorithm to find an optimum of the*

target objective by optimizing the objective O in x iterations. Let A_i be the size of the i-th meta-iteration of the SaRA algorithm, and T be the expected running time (number of iterations) of this algorithm. Then the following holds:

$$T \leq K \sum_{y=1}^{\infty} \left(A_y \prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{A_i} P_O(j) \right) \right).$$

Proof: For convenience, let's define prefix sums:

$$S_i = \sum_{j=1} i A_i.$$

Assume the optimum of the target objective is found at the x-th iteration of optimizing the objective O during the y-th meta-iteration. The number of iterations of the SaRA algorithm before and including this iteration is:

$$D(x, y) \leq S_{y-1} K + A_y (K - 1) + x \leq S_y K.$$

The probability of this to happen is:

$$E_O(x, y) = P_O(x) \prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{A_i} P_O(j) \right).$$

Then the upper bound on T is:

$$\begin{aligned}
T &= \sum_{y=1}^{\infty} \sum_{x=1}^{A_y} (D(x, y) E_O(x, y)) \\
&\leq K \sum_{y=1}^{\infty} \left(S_y \sum_{x=1}^{A_y} E_O(x, y) \right) \\
&= K \sum_{y=1}^{\infty} \left(S_y \sum_{x=1}^{A_y} P_O(x, y) \prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{A_i} P_O(j) \right) \right) \\
&= K \sum_{y=1}^{\infty} \left(S_y \left(\prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{A_i} P_O(j) \right) \right) \sum_{x=1}^{A_y} P_O(x, y) \right).
\end{aligned}$$

Let's define $Z_O(x) = 1 - \sum_{i=1}^{A_x} P_O(i)$. The upper bound above can be rewritten as:

$$\begin{aligned}
T &\leq K \sum_{y=1}^{\infty} \left(S_y \left(\prod_{i=1}^{y-1} Z_O(i) \right) (1 - Z_O(y)) \right) \\
&= K \sum_{y=1}^{\infty} \left(S_y \left(\prod_{i=1}^{y-1} Z_O(i) - \prod_{i=1}^{y} Z_O(i) \right) \right) \\
&= K \sum_{y=1}^{\infty} \left(A_y \prod_{i=1}^{y-1} Z_O(i) \right) \\
&= K \sum_{y=1}^{\infty} \left(A_y \prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{A_i} P_O(j) \right) \right),
\end{aligned}$$

which completes the proof. ■

IV. Exponential Restart Strategy

The formulation of Theorem 1 considers the most general version of the SaRA algorithm. Due to this fact, it is difficult to use its results in the above specified form. To construct a statement that is more usable, one needs to restrict the values of A_i. In this section, the *exponential restart strategy* for A_i is analyzed. In other words, it is required that $A_{i+1}/A_i \to \alpha$ for some constant $\alpha > 1$ when i grows. For simplicity, it is assumed that $A_i = \lfloor \alpha^{i-1} \rfloor$, such that $A_1 = 1$ and the ratio of consecutive A_i really approaches α.

A. Upper Bound

Theorem 2. *Let K be the number of objectives, O be an arbitrary objective, and T_O be the expected number of iterations for the underlying optimization algorithm to find an optimum of the target objective by optimizing the objective O. Let $A_i = \lfloor \alpha^{i-1} \rfloor$ be the size of the i-th meta-iteration of the SaRA algorithm, and T be the expected running time (number of iterations) of this algorithm. Then the following holds:*

$$T \leq K \frac{\alpha^2}{\alpha - 1} T_O.$$

Proof: Using the bound from Theorem 1 and the values of A_i, one can write the following bound:

$$T \leq K \sum_{y=1}^{\infty} \left(\lfloor \alpha^{y-1} \rfloor \prod_{i=1}^{y-1} \left(1 - \sum_{j=1}^{\lfloor \alpha^{i-1} \rfloor} P_O(j) \right) \right).$$

By ignoring all multipliers in the product except for the last one, the right part is not decreased:

$$T \leq K \sum_{y=1}^{\infty} \left(\lfloor \alpha^{y-1} \rfloor \left(1 - \sum_{j=1}^{\lfloor \alpha^{y-2} \rfloor} P_O(j) \right) \right).$$

Note that even for $y = 1$ the expression involving $\lfloor \alpha^{y-2} \rfloor$ remains correct. The bound is further rewritten to complete the proof:

$$\begin{aligned}
T &\leq K \sum_{y=1}^{\infty} \left(\lfloor \alpha^{y-1} \rfloor \left(1 - \sum_{j=1}^{\lfloor \alpha^{y-2} \rfloor} P_O(j) \right) \right) \\
&= K \sum_{y=1}^{\infty} \left(\lfloor \alpha^{y-1} \rfloor \sum_{j=\lfloor \alpha^{y-2} \rfloor + 1}^{\infty} P_O(j) \right) \\
&= K \sum_{y=1}^{\infty} \left(\lfloor \alpha^{y-1} \rfloor \sum_{t=y-1}^{\infty} \sum_{j=\lfloor \alpha^{t-1} \rfloor + 1}^{\lfloor \alpha^{t} \rfloor} P_O(j) \right) \\
&= K \sum_{y=0}^{\infty} \left(\lfloor \alpha^{y} \rfloor \sum_{t=y}^{\infty} \sum_{j=\lfloor \alpha^{t-1} \rfloor + 1}^{\lfloor \alpha^{t} \rfloor} P_O(j) \right) \\
&= K \sum_{t=0}^{\infty} \sum_{y=0}^{t} \left(\lfloor \alpha^{y} \rfloor \sum_{j=\lfloor \alpha^{t-1} \rfloor + 1}^{\lfloor \alpha^{t} \rfloor} P_O(j) \right)
\end{aligned}$$

$$
\begin{aligned}
&= K\sum_{t=0}^{\infty}\left(\left(\sum_{j=\lfloor\alpha^{t-1}\rfloor+1}^{\lfloor\alpha^t\rfloor} P_O(j)\right)\left(\sum_{y=0}^{t}\lfloor\alpha^y\rfloor\right)\right)\\
&\le K\sum_{t=0}^{\infty}\left(\left(\sum_{j=\lfloor\alpha^{t-1}\rfloor+1}^{\lfloor\alpha^t\rfloor} P_O(j)\right)\frac{\alpha^{t+1}-1}{\alpha-1}\right)\\
&< K\frac{\alpha^2}{\alpha-1}\sum_{t=0}^{\infty}\left(\left(\sum_{j=\lfloor\alpha^{t-1}\rfloor+1}^{\lfloor\alpha^t\rfloor} P_O(j)\right)\alpha^{t-1}\right)\\
&< K\frac{\alpha^2}{\alpha-1}\sum_{t=0}^{\infty}\left(\sum_{j=\lfloor\alpha^{t-1}\rfloor+1}^{\lfloor\alpha^t\rfloor} P_O(j)j\right)\\
&= K\frac{\alpha^2}{\alpha-1}\sum_{j=1}^{\infty}P_O(j)j = K\frac{\alpha^2}{\alpha-1}T_O.
\end{aligned}
$$

■

B. Lower Bound

Theorem 2 states the upper bound on the expected running time of the SaRA algorithm. Now it is shown that this upper bound is, in a sense, reachable.

Theorem 3. *For a fixed $\alpha > 1$ and an arbitrary $\varepsilon > 0$ there exist a set of K objectives such that:*

$$\frac{T}{T_0} > K\frac{\alpha^2}{\alpha-1}(1-\varepsilon),$$

where T is the expected running time of the SaRA algorithm and T_0 is the smallest expected running time to optimize the target objective using one of K objectives.

Proof: Consider a problem with K objectives where for all objectives O except for the last one $T_O = \infty$, while for the K-th objective the target optimum is reached exactly at the iteration $T_0 = \lfloor\alpha^N\rfloor + 1$ for some integer N. Then the running time for the SaRA algorithm (which is not a random value anymore) is exactly:

$$
\begin{aligned}
T &= \sum_{i=1}^{N+1}\left(K\lfloor\alpha^{i-1}\rfloor\right) + (K-1)\lfloor\alpha^{N+1}\rfloor + \lfloor\alpha^N\rfloor + 1\\
&= \sum_{i=1}^{N+2}\left(K\lfloor\alpha^{i-1}\rfloor\right) - \lfloor\alpha^{N+1}\rfloor + \lfloor\alpha^N\rfloor + 1.
\end{aligned}
$$

A lower bound for T/T_0 can be achieved from a lower bound on T and an upper bound on T_0. The latter one can be immediate: $T_0 = \lfloor\alpha^N\rfloor + 1 \le \alpha^N + 1$. The former one can be constructed as follows:

$$
\begin{aligned}
T &= K\sum_{i=1}^{N+2}\lfloor\alpha^{i-1}\rfloor - \lfloor\alpha^{N+1}\rfloor + \lfloor\alpha^N\rfloor + 1\\
&> K\sum_{i=1}^{N+2}\left(\alpha^{i-1}-1\right) - \lfloor\alpha^{N+1}\rfloor + \lfloor\alpha^N\rfloor + 1\\
&= K\frac{\alpha^{N+2}-1}{\alpha-1} - K(N+2) - \lfloor\alpha^{N+1}\rfloor + \lfloor\alpha^N\rfloor + 1\\
&> K\frac{\alpha^{N+2}-1}{\alpha-1} - K(N+2) - \alpha^{N+1} + \alpha^N\\
&= \frac{K\alpha^{N+2} - K - K(N+2)(\alpha-1) - \alpha^N(\alpha-1)^2}{\alpha-1}\\
&> K\frac{\alpha^{N+2}}{\alpha-1}\left(1 - \frac{1}{K}\left(1-\frac{1}{\alpha}\right)^2 - \frac{K(N+3)}{\alpha^{N+1}}\right).
\end{aligned}
$$

Using the lower and upper bounds, one can get that:

$$\frac{T}{T_0} > K\frac{\alpha^2}{\alpha-1}\left(\frac{1-\frac{1}{K}\left(1-\frac{1}{\alpha}\right)^2 - \frac{K(N+3)}{\alpha^{N+1}}}{1+\frac{1}{\alpha^N}}\right),$$

where the fraction in big parenthesis can be arbitrarily close to 1 from below by appropriate choices of N and K. ■

C. Optimal Exponent Base

Theorems 2 and 3 show that the performance of the SaRA algorithm with exponential restart strategy is proportional to $\alpha^2/(\alpha-1)$ where α is the exponent base for the meta-iteration size. If $\alpha > 1$, the single minimizer for this expression is $\alpha = 2$, which suggests that the optimal choice for the exponential restart strategy is to have the size of the i-th meta-iteration equal to $A_i = 2^{i-1}$. For this size, the upper bound on the expected running time of the SaRA algorithm is

$$4K\min_O T_O$$

where T_O is the expected running time of the underlying optimization algorithm to find an optimum of the target objective by optimizing the objective O.

V. The Impact of Restarts

In the beginning of Section III it was mentioned that it is not a simple problem to decide whether it is needed to perform algorithm restarts when the objectives are switched. The general idea is that restarts allow estimating the running time judging only about the considered algorithms and not their initializations. In other words, the algorithm that performs restarts does not care about what was remained from the previous iteration round.

When the algorithm does not perform restarts, things may get either worse or better depending on the properties of underlying optimization algorithms and of the objectives. First, an artificial problem is described for which it is provably better to not use restarts.

Theorem 4. *Consider the situation when the optimization algorithm always find an optimum of the target objective O at the 2^N-th iteration. Consider the problem with K identical objectives equal to the objective O. Then, the running time of the SaRA algorithm with $\alpha = 2$ is $K(2^N-1)+2^N$, while the same algorithm without restarts has the running time of 2^N.*

Proof: The version with restarts has to complete N meta-iterations without any success, and the optimum is found only at the end of the $(N+1)$-th meta-iteration. The version without restarts continues optimizing effectively the same objective uninterrupted. ■

A case has just been considered where using restarts make things worse up to a degree of K. However, there exist

situations where not using restarts leads to the results that are significantly worse.

Theorem 5. *Consider a problem with two objectives. The domain is a set of bit strings of length N. Let us define* LeadingOnes(x) *to be the length of the maximum prefix of the bit-string x consisting of 1-bits, and* ZeroMax(x) *to be the number of 0-bits in the bit-string x.*

The objectives for the considered problem are defined as follows:

- *The target objective:*

$$O_1(x) = \begin{cases} N & \text{if } x = 0^N; \\ N+1 & \text{if } x = 1^N; \\ \textsc{LeadingOnes}(x) & \text{otherwise.} \end{cases}$$

- *The secondary objective:* $O_2(x) = \textsc{ZeroMax}(x)$.

The optimization algorithm is the randomized local search algorithm, which does the following:

1) $X \leftarrow$ *a randomly generated bit string of length N;*
2) $Y \leftarrow X$ *with exactly one randomly selected bit flipped;*
3) *if* $f(Y) \geq f(X)$*, then* $X \leftarrow Y$*;*
4) *go to step 2.*

Then, the expected running time of the SaRA algorithm is $O(N^2)$, while the expected running time of the same algorithm without restarts is infinite.

Proof: If the optimizer comes to the situation where $X = 0^N$, it stucks there until the next restart happens whatever the objective selection algorithm does, because if O_1 is selected, any single mutation decreases this objective, and 0^N is already an optimum for the O_2 objective.

This means that, in the case of no restarts, the algorithm stucks at the local optimum with high probability, because ZeroMax is faster to optimize using the randomized local search algorithm than LeadingOnes ($\Theta(N \log N)$ [22] versus $\Theta(N^2)$ [23]). So the expected running time of the algorithm without restarts is infinite.

The probability for O_1 to stuck at the local optimum after random initialization is very small: the initial value can be O^N with the probability of 0.5^N, or it can be initialized to $X = 10\ldots$ and then mutated to $00\ldots$, which can be 0^N — the very imprecise upper bound to the probability of this to happen is $0.25/N$. This means that, when the size of the meta-iteration is large enough, the O_1 objective will not be optimized with the probability of at most $0.25/N + 0.5^N$. Due to the results from [23], for every $\delta > 0$ there exists a constant c such that the probability for LeadingOnes not to be optimized in cN^2 iterations is at most δ. Let us fix $\delta = 0.1$ and find m such that $2^{m-1} < cN^2 \leq 2^m$. Then, at each meta-iteration of the SaRA algorithm starting with the $(m+1)$-th one, the probability for O_1 not to be optimized is at most $0.1 + 0.9(0.25/N + 0.5^N)$, which is at most 0.2875 if $N \geq 3$. The expected running time of the SaRA algorithm is thus at most:

$$\begin{aligned} T &\leq \sum_{i=m+1}^{\infty} \left(2(2^i - 1)(1 - 0.2875)0.2875^{i-m-1}\right) \\ &< 0.7125 \sum_{i=m+1}^{\infty} \left(2^{i+1} 0.2875^{i-m-1}\right) \\ &= 0.7125 \sum_{i=0}^{\infty} \left(2^{m+2+i} 0.2875^{i}\right) \\ &= 0.7125 \cdot 2^{m+2} \sum_{i=0}^{\infty} 0.575^i \\ &= \frac{0.7125}{1 - 0.575} \cdot 2^{m+2} < 14 \cdot 2^{m-1} < 14cN^2, \end{aligned}$$

that is, $O(N^2)$, which completes the proof. ■

To sum up this section, using an algorithm with restarts ensures that the overall performance will be at most $4K$ multiplied by the performance of the optimization algorithm on the target objective. An algorithm without restarts may save up some iterations, but in some cases the performance degrades drastically.

VI. Conclusion

An objective selection algorithm SaRA, for "switch-and-restart algorithm", is presented to control a single-objective optimization algorithm for solving problems with a target objective, which one needs to optimize, and several extra objectives, which can or cannot help optimizing the target one. This algorithm consists of meta-iterations, in the i-th meta-iteration the objectives are optimized one at turn for 2^{i-1} iterations. When switching between different objectives, the underlying optimization algorithm is restarted from scratch.

If one considers, for each objective O, the expected running time, measured in iterations of the underlying optimization algorithm, needed to find an optimum of the target objective by optimizing the objective O, then the expected running time of the presented algorithm is at most $4K \min_O T_O$, where K is the number of objectives.

It is also shown that the given upper bound is asymptotically tight by presenting, for each $\varepsilon > 0$, a problem which requires at least $4K \min_O T_O(1 - \varepsilon)$ iterations to find the optimum.

The impact of restarts needed by this algorithm is discussed. Two problems are presented: for the first one, the version of the algorithm without restarts is K times faster, while for the second one, the version without restarts, in expectation, cannot find the optimum of the target objective.

The author believes that a stronger proof can be constructed: one may take not the expected minima over performances of single objectives, but the minima over performances with high probability. In a sense, a possibility of this is demonstrated in Theorem 5, but a more rigorous proof is required for the general case.

The presented algorithm can be applied not only for objective selection, but for choosing the best of virtually any discrete algorithm variations, or even of any finite number of algorithms. The properties of the algorithm, like the expression for the upper bound on the expected running time, are preserved in any case. However, if the number of options to select grows with the problem size, the asymptotic behavior

of the presented algorithm may no longer coincide with the asymptotic behavior of the best option.

VII. Acknowledgments

This work was partially financially supported by the Government of Russian Federation, Grant 074-U01. The author thanks Arina Buzdalova for fruitful discussions about objective selection algorithms.

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] ——, "Minimum Spanning Trees Made Easier via Multi-objective Optimization," *Natural Computing*, vol. 5, no. 3, pp. 305–319, 2006.

[3] 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.

[4] 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.

[5] 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.

[6] J. Handl, S. C. Lovell, and J. D. Knowles, "Multiobjectivization by Decomposition of Scalar Cost Functions," in *Parallel Problem Solving from Nature PPSN X*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 5199, pp. 31–40.

[7] D. F. Lochtefeld and F. W. Ciarallo, "Deterministic Helper-Objective Sequences Applied to Job-Shop Scheduling," in *Proceedings of Genetic and Evolutionary Computation Conference*. ACM, 2010, pp. 431–438.

[8] A. Buzdalova, M. Buzdalov, and V. Parfenov, "Generation of Tests for Programming Challenge Tasks Using Helper-Objectives," in *5th International Symposium on Search-Based Software Engineering*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 8084, pp. 300–305.

[9] 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.

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

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

[12] A. Gosavi, "Reinforcement Learning: A Tutorial Survey and Recent Advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[13] 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*, vol. 1. IEEE Computer Society, 2013, pp. 203–208.

[14] 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.

[15] T. Jansen, "On the analysis of dynamic restart strategies for evolutionary algorithms," in *Parallel Problem Solving from Nature PPSN VII*, ser. Lecture Notes on Computer Science. Springer, 2002, vol. 2439, pp. 33–43.

[16] R. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intelligence*, vol. 27, pp. 97–109, 1985.

[17] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[18] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," *AI Magazine*, 2014.

[19] C. P. Gomes and B. Selman, "Algorithm portfolios," *Artificial Intelligence*, vol. 126, no. 1, pp. 43–62, 2001.

[20] S. Y. Yuen, C. K. Chow, and X. Zhang, "Which Algorithm Should I Choose at Any Point of the Search: An Evolutionary Portfolio Approach," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2013, pp. 567–574.

[21] P. Baudiš and P. Pošík, "Online Black-Box Algorithm Portfolios for Continuous Optimization," in *PPSN XIII*, ser. Lecture Notes on Computer Science, 2014, vol. 8672, pp. 40–49.

[22] C. Witt, "Optimizing Linear Functions with Randomized Search Heuristics – the Robustness of Mutation," in *Proceedings of the 29th Annual Symposium on Theoretical Aspects of Computer Science*, 2012, pp. 420–431.

[23] S. Böttcher, B. Doerr, and F. Neumann, "Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem," in *Parallel Problem Solving from Nature PPSN XI*, ser. Lecture Notes in Computer Science. Springer, 2010, vol. 6238, pp. 1–10.