Runtime Analysis of Different Approaches to Select Conflicting Auxiliary Objectives in the Generalized OneMax Problem

Arina Buzdalova, Irina Petrova, Maxim Buzdalov
ITMO University
49 Kronverkskiy prosp.
Saint-Petersburg, Russia, 197101
Email: abuzdalova@gmail.com, irenepetrova@yandex.com, mbuzdalov@gmail.com

***Abstract*— It has been shown that single-objective optimization may be improved by introducing auxiliary objectives. Ideally, auxiliary objectives should be designed to be independent. However, in practice, this is not always easy or possible and the objectives may be conflicting. Particularly, this happens in the Job-Shop Scheduling problem and in certain search-based software engineering problems.**

We theoretically analyse different approaches of using auxiliary objectives on the general ONEMAX problem with conflicting auxiliary objectives ONEMAX and ZEROMAX. In most of the considered methods, the optimized objectives are selected dynamically. We also consider a method, where the same objectives are optimized during the whole run. We show that dynamic selection of conflicting auxiliary objectives combined with explicit optimization of the target objective has the best runtime of $O(n \log n)$ among all the considered methods.

I. INTRODUCTION

The approaches of using auxiliary objectives can be divided into three groups [19]. One group contains methods that treat constraints as auxiliary objectives and transform a constrained problem into an unconstrained one [7], [18]. Other methods consider diversity as an objective [19]. We consider the so called *multiobjectivization* methods, that use auxiliary objectives correlated with the target objective. It has been shown both theoretically and practically that multiobjectivization may improve efficiency of single-objective optimization [2], [11], [14], [15], [19]. The rest of the paper considers auxiliary objectives used in the context of multiobjectivization.

Auxiliary objectives may be designed as parts of the decomposition of the original *target* objective [11]. The optimum of the target objective should belong to the Pareto front formed by the resulting auxiliary objectives. In this case, the auxiliary objectives are optimized simultaneously using a multi-objective evolutionary algorithm in order to find the optimum of the target objective.

There are some different approaches as well, when the auxiliary objective, which should be used at the current step of optimization, is selected dynamically [5], [10]. In this case, it is not required that the auxiliary objectives comprise a decomposition of the target objective. Therefore, dynamic selection is helpful when objectives are generated automatically and it is hard to know their properties in advance [3].

Most theoretical works consider multiobjectivization by decomposition and investigate when it may outperform single-objective optimization [2], [9], [14], [15]. However, to the best of our knowledge, only few theoretical works consider dynamic selection of objectives [4].

In the mentioned theory papers auxiliary objectives are usually designed to be independent. So there seem to be no theoretical results regarding the case of conflicting auxiliary objectives. However, such objectives may arise in practical applications. For example, consider the following Job Shop Scheduling problem [13]. A predefined set of jobs and machines is given. A job consists of operations, each operation should be processed by a specified machine and takes a specified processing time. No two operations of a job can be processed simultaneously, each machine can process only one operation at time, all machines run at the same speed, no preemption allowed. The target objective is the total flowtime of the schedule. This problem may be solved my multiobjectivizing the total flowtime into individual flowtimes needed to complete a certain job. In this case optimization of the flow time to complete one job may result in the increasing of the flow time for some other job [13]. Thus, the auxiliary objectives are conflicting.

Another example of potentially conflicting auxiliary objectives comes from search-based software engineering problems, such as worst-case execution time test generation. When solving this kind of problems, the auxiliary objectives may be generated automatically [3], so it is also hard to ensure that there are no conflicts.

It is known from experimental studies that for the above problems the best solutions are usually obtained using dynamic selection of auxiliary objectives combined with the simultaneous explicit optimization of the target objective [6], [16]. Theoretical analysis of a problem with conflicting auxiliary objectives may help to better understand the reasons why different methods of objective selection are efficient or not for such problems.

Overall, the aim of this paper is to prove runtime estimations for the known methods of objective selection and to compare them on a model problem with conflicting objectives.

II. Preliminaries

In this section we define the model problem and a multi-objective evolutionary algorithm, which serves as the base for the most of the considered algorithms. We also prove some statements about this algorithm, which are intensively used in the rest of the paper.

A. Model problem

In general, it is needed to optimize the target objective t. Auxiliary objectives $h_1, h_2, \ldots, h_m$ may be used to find the optimum of t in a smaller number of fitness function evaluations. Objectives $t, h_1, \ldots, h_m$ form an optimization problem with auxiliary objectives. Notice that there is no need to find the optima of the auxiliary objectives, they are used just to enhance optimization of the target objective. Below we formulate a simple model problem used in this paper as an example of a problem with target and auxiliary objectives.

Definition 1 (OM$_d$ problem): The target objective OM$_d$ is calculated as the number of bits in an individual of length n that matches a given bit mask. The bit mask has d 0-bits and $n-d$ 1-bits. The auxiliary objective ONEMAX is calculated as the number of 1-bits in an individual, while another auxiliary objective ZEROMAX is calculated as the number of 0-bits.

In the rest of the paper, we say that an algorithm *solves the OM$_d$ problem* if it finds the optimum of OM$_d$ using the ONEMAX and ZEROMAX auxiliary objectives as defined in this algorithm.

Auxiliary objectives ONEMAX and ZEROMAX are not independent, because increasing of the ONEMAX value results in decreasing of the ZEROMAX value. Also the auxiliary objectives may conflict with the target objective. Let us explain why a similar situation may arise in the Job-Shop Scheduling problem [13]. Assume that the target objective is the total flow time, which should be minimized. Auxiliary objectives are flow times of particular jobs. Such objectives may be in conflict, because the same machines are used in different jobs. In other words, optimization of the flow time needed to complete one job may result in increasing of the flow time needed to complete some other job. What is more, the total flow time may increase, if an improper job was selected to be optimized.

B. Simple Evolutionary Multi-Objective Optimizer

Most of the algorithms considered in this paper deal with optimization of several objectives. We use an example multi-objective evolutionary algorithm, where the objectives are compared based on Pareto dominance and the non-dominated set of solutions is maintained. To describe this algorithm we need to define fitness in a multi-objective case.

Definition 2: Let $c_1, c_2, \ldots, c_m$ be the objectives to be optimized in a multi-objective evolutionary algorithm. The fitness of an individual x is defined as a vector $f = (c_1(x), c_2(x), \ldots, c_m(x))$.

As a multi-objective evolutionary algorithm, we consider the Simple Evolutionary Multi-Objective Optimizer (SEMO) [12], [14], [17]. This algorithm stores all non-dominated individuals in the population P. At each step of the algorithm, an individual x is chosen randomly from P and one bit is flipped to obtain a new individual x'. Then non-dominated individuals are selected from P and x' to form the new population P', which is used as P at the next step of the algorithm. In the original version of this algorithm proposed in [12], the mutated individual is not added to the population if an individual with the same fitness is already present. In the version considered in the present paper, the old individual with the same fitness is replaced by the newly mutated individual. The pseudocode of SEMO is presented in Algorithm 1).

Algorithm 1 SEMO algorithm [12]

```
Population P ← a randomly generated individual
while (Stopping criterion is not reached) do
    Randomly select individual x from P
    Individual x' ← mutate x (flip one bit)
    Select non-dominated individuals P' from P ∪ {x'}
    if ∃y ∈ P' : f(y) = f(x') and y ≠ x' then
        Remove y from P'
    end if
    P ← P'
end while
```

C. General Running Time of Auxiliary Objective Approaches Based on SEMO

In this section we give a general theorem that allows estimating running time of auxiliary objective approaches based on SEMO. We believe that results of this kind already belong to folklore, but nevertheless give the proof in the Appendix.

Theorem 1: Consider optimization of objectives $c_1, c_2, \ldots, c_m$ by the SEMO algorithm. Let T be the runtime of the random local search while optimizing some objective c_k, where $k \in \{1, \ldots, m\}$. Let F be the size of the current population. Then SEMO finds the optimum of c_k in expected number of fitness function evaluations of $O(E(F) \cdot E(T))$, where $E(F)$ and $E(T)$ are the expected values of F and T correspondingly.

Now, using Theorem 1, we can describe the general scheme for the most further analyses of different auxiliary objective approaches based on SEMO. First, we estimate the expected population size in a considered auxiliary objective approach. We also use the fact that the expected optimization time of the OM$_d$ objective using random local search is $\Theta(n \log n)$ [1], as the problem of optimizing OM$_d$ is just another instance of the coupon collector problem. Then the final result is obtained by substituting the expected running time to optimize OM$_d$ and the expected population size to Theorem 1.

III. Helper Objective Approach: Random Selection of Auxiliary Objectives

To start with, we analyse the dynamic objective approach proposed in [10]. In this approach, the target objective is optimized together with the dynamically selected auxiliary objectives. We consider the two-objective case, when the one

objective is the target one and the other objective is selected from a randomly ordered set of auxiliary objectives. This case was reported in [10] to be the most efficient in practice.

In order to estimate the population size in the algorithms considered in this section, we need two lemmas presented below.

Lemma 1: Assume that two objectives are optimized by SEMO and each objective may be of n different values. Then the size of population while optimizing these two objectives is at most n individuals.

Proof: Denote a population as P and the first objective as h. Assuming that there are $n+1$ individuals in P, we get that there exist two individuals x and $x' \neq x$ such that $h(x) = h(x')$, as there are no duplicates. However, P consists of individuals which do not dominate each other, which means that $h(x) \neq h(x')$. The contradiction proves the lemma. ■

Lemma 2: Assume that the auxiliary objective h (ONEMAX or ZEROMAX) was selected at the previous iteration, and then the opposite criterion $h' \neq h$ (ZEROMAX or ONEMAX correspondingly) is selected at the current iteration. Then the Pareto front at the current iteration consists of at most two individuals.

Proof: Denote the Pareto front from the previous iteration as P and the Pareto front from the current iteration as P'. Consider the individual x from P with maximal value of the target objective. Then for every individual $y \in P$ it holds that $t(x) \geq t(y)$ and $h(x) \leq h(y)$. When the opposite objective h' is selected, for each individual $y \in P$ it holds that $t(x) \geq t(y)$ and $h'(x) = n - h(x) \geq n - h(y) = h'(y)$. So each individual from P is dominated by x. Therefore, the only point from P that is kept in P' is x. One more point may be added to P' in the case of the successful mutation. So P' consists of at most two individuals. ■

In the original approach presented in [10], each auxiliary objective is selected just once and is optimized for the same number of iterations. So the total number of iterations is needed to calculate the number of iterations for a particular objective. However, the total number of iterations is unknown in the case when the stopping criteria is finding the optimum. So we denote the number of iterations given for one objective as k and only require that k is at least one and does not exceed the total number of iterations. In this case, a particular objective may be optimized more than once.

We consider two variants of the dynamic objective approach. The first variation (Theorem 2) is closer to the original version, as the objectives are selected according to a fixed order, which was earlier generated at random. In the second variation (Theorem 3), an auxiliary objective is just selected randomly.

Theorem 2: Consider an algorithm that starts from a randomly chosen auxiliary objective and then switches the auxiliary objective every k iterations (the auxiliary objective is being optimized simultaneously with the target objective by SEMO). This algorithm solves the OM_d problem in $O(\min(n,k) \cdot n \log n)$ fitness function evaluations in expectation.

Proof: The algorithm starts with one individual in the population. At each iteration, the current population may increase at most by one. According to Lemma 2, there are at most two individuals in the population after the auxiliary objective has been changed. Thus, the number of individuals in a population is bounded by the length of the period when the same objective is used, which is k.

Therefore, we get that the size of a population is $O(k)$. On the other hand, according to Lemma 1, the size of a population is less than or equal to n. Thus, the expected size of a population P is bounded by $O(\min(n,k))$ at any iteration of the algorithm: $E(|P|) = O(\min(n,k))$.

By substitution of $E(|P|)$ into Theorem 1, we get the expected number of iterations to find the optimum of OM_d, which is $O(\min(n,k) \cdot n \log n)$. ■

Theorem 3: Consider the algorithm that randomly chooses a helper objective every k iterations and optimizes it simultaneously with the target objective by SEMO. This algorithm solves the OM_d problem in $O(\min(n,k) \cdot n \log n)$ fitness function evaluations in expectation.

Proof: The proof of this theorem is analogous to the proof of Theorem 2. The only difference is that we do not know exactly the length of the period when the same objective is used, because the auxiliary objectives are switched randomly. So let us estimate the length of this period, in other words, the expected number of iterations until the currently selected auxiliary objective is switched to the other auxiliary objective:

$$E(I) = \frac{1}{2}k + \frac{1}{2^2}2k + \ldots + \frac{1}{2^t}tk = \sum_{i=1}^{t} \frac{ik}{2^i} = k \sum_{i=1}^{t} \frac{i}{2^i} = \Theta(k).$$

This result together with Lemma 1 and Theorem 1 proves the statement. ■

As we can see, for both considered variations of the dynamic objective approach, the expected runtime to solve the model problem is $O(\min(n,k) \cdot n \log n)$. Thus, for relatively small fixed values of k, the expected runtime of $O(n \log n)$ may be obtained, which is as efficient as the optimization of the target objective without auxiliary objectives.

IV. REINFORCEMENT BASED SELECTION OF AUXILIARY OBJECTIVES

Consider the dynamic objective approach, where auxiliary objectives are selected using reinforcement learning (RL) [20]. Such an approach was proposed in [5]. In RL, an *agent* applies an action to an *environment*, then the environment returns some representation of its *state* and a numerical *reward* to the agent, and the process repeats. Initially, a single objective evolutionary algorithm was used. The corresponding method is called EA+RL. In the EA+RL method, an evolutionary algorithm is treated as an environment. Each action of the agent corresponds to selection of an objective to be optimized at the current generation. The agent selects an objective from the set of auxiliary objectives and the target objective. The target objective is implicitly taken into account in the reward

optimized by RL, so it is possible to select only one objective to be explicitly optimized at each iteration.

As explained above, EA+RL optimizes the target objective implicitly. However, explicit optimization of the target objective together with the dynamically selected auxiliary objective seems to still be more efficient in the presence of objectives which may conflict with the target objective. The corresponding method is called MOEA+RL. The only difference of MOEA+RL from EA+RL is that a multi-objective evolutionary algorithm instead of a single-objective one is used, so the selected auxiliary objective is optimized simultaneously with the target objective. In this section we analyse both methods, starting from the single-objective one.

A. General Principles of Objective Selection by Reinforcement Learning

The general pseudocode of EA/MOEA + RL is presented in Algorithm 2. Instead of EA, we use RLS, and we use SEMO as MOEA. For RL, the Q-learning algorithm is used. So the considered implementations are named RLS+Q-learning and SEMO+Q-learning correspondingly. In Q-learning, the efficiency of selecting an objective h in a state s is measured by the value $Q(s,a)$, which is updated dynamically after each selection as shown in line 14 of the pseudocode.

Algorithm 2 RLS/SEMO + Q-learning Algorithm

1: Population $P \leftarrow$ a randomly generated individual
2: Define target objective t
3: Define set of auxiliary objectives H
4: $Q(s,h) \leftarrow 0$ for each state s and action $h \in H$
5: Calculate fitness for each individual in P
6: **while** (Stopping criterion is not reached) **do**
7: $s \leftarrow$ best value of t in P
8: Randomly select individual x from P
9: Individual $x' \leftarrow$ mutate x (flip one bit)
10: Select objective h: $Q(s,h) = \max_{h' \in H} Q(s,h')$
11: Update P using x' and h
12: $s' \leftarrow$ best value of t in P
13: $r \leftarrow s' - s$
14: $Q(s,h) \leftarrow (1-\alpha)Q(s,h) + \alpha(r + \max\limits_{h' \in H} Q(s',h'))$
15: **end while**

In RLS + Q-learning, the set of auxiliary objectives (line 3) consists of the target objective OM_d and two auxiliary objectives OneMax and ZeroMax. In SEMO + Q-learning, the set of auxiliary objectives consists only of two auxiliary objectives.

In RLS + Q-learning, the individual with the higher value of the selected objective h (x or x') is selected for the next population P in line 11. In SEMO + Q-learning, non-dominated individuals from $P \cup \{x'\}$ are selected as new P. In this case, the target objective and the selected objective h take part in the comparison according to Pareto dominance.

Notice that there is always one individual in a population of RLS+Q-learning, while in MOEA+Q-learning a population may consist of several individuals.

B. EA+RL: Single-Objective Approach

In this section we analyse the method, where RL is used to select an objective — the target objective or one of the auxiliary objective, which is optimized in the current iteration of EA. As RLS is used instead of EA, the population consists of a single individual.

Theorem 4: Consider the RLS+Q-learning algorithm. The states correspond to the current target fitness value, the reward is defined as the difference of target fitness values in two consequent iterations. The RLS+Q-learning algorithm does not solve the OM_d problem with a nonzero probability in the case of $2 \leq d \leq n-2$.

Proof: Possible states of the algorithm are presented in Fig. 1. The vertical axis corresponds to possible values of t, and, as a consequence, to states of reinforcement learning algorithm. The horizontal axis corresponds to possible values of h_1. Let us show a possible run of the algorithm in which the problem is not solved.

Assume that in the optimal solution in the position k_1 there is a 0-bit, and in the position k_2 there is a 1-bit. Assume that the algorithm obtained an individual of all ones except positions k_1 and k_2. Then $t = n-d$ and $h_1 = n-2$ (point A in Fig 1). The corresponding RL state is $n-d$. Assume that it is the first time when the algorithm turned to the state $n-d$. So the agent does not have any knowledge about efficiency of objectives in this state.

Assume that the agent randomly selects the objective h_1 and obtains individual with the flipped bit at the position k_2. So $t = n-d+1$ and $h = n-1$ (the point B in Fig. 1). The agent has achieved a positive reward for the action h_1 in the state $n-d$. Since the agent has no experience in the state $n-d+1$, it can select the action h_1 just at random. Assume that the algorithm obtained the new individual with the flipped k_1 bit, which is the optimum of h_1 (the point C in Fig. 1). The algorithm has moved to the state $n-d$. The agent has already learned that in the state $n-d$ the h_1 objective gives positive reward, so the agent chooses h_1. However, the mutated individual is worse than the optimum of h_1 and it will not be selected for the next generation. The obtained reward is zero, so the estimation of objective efficiency does not change

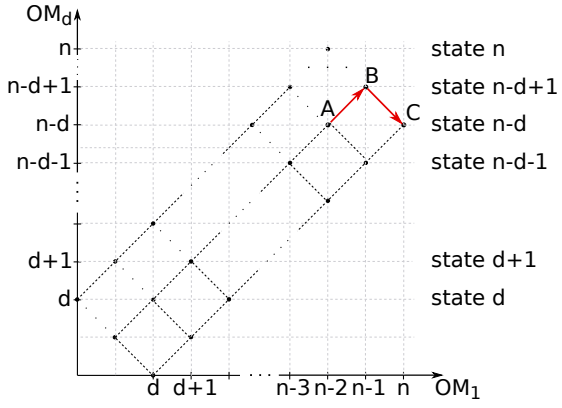

Fig. 1. Markov chain of RLS+Q-learning

according to the formula in line 14 of Algorithm 2. Therefore, the algorithm will never move from the current state. ■

C. MOEA+RL: *Multi-Objective Approach*

In this section we analyse the method which optimizes the target objective simultaneously with an auxiliary objective selected by RL. We use SEMO as the multi-objective algorithm. Compared to a single-objective evolutionary algorithm, the reward and states should be redefined for the case of a population based algorithm.

Consider the individual that has the highest value of the target objective in the current population. We denote it as the *current best individual* and use this individual to define states and reward in the same way as in the single-objective case (see Section IV-B).

Theorem 5: Consider the SEMO+Q-learning algorithm. The states correspond to the target fitness value of the current best individual, the reward is defined as the difference of target fitness values of the current best individuals in two consequent iterations. The considered algorithm solves the OM_d problem in $O(n \log n)$ fitness function evaluations in expectation.

Proof: Notice that the target fitness of the best individual may not decrease, because this individual can not be dominated by any individual with lower value of the target objective. Therefore, the agent would never move to a state that corresponds to a lower value of the target objective.

Once the agent gets to a state, it chooses any of the two auxiliary objectives equiprobably, since the state is visited for the first time and there is no information about the efficiency of the objectives. If an individual with a higher target fitness is generated by mutation and accepted by SEMO, the agent moves to a new state and again selects one of the two auxiliary objectives equiprobably. In the opposite case, the agent gets zero reward and stays in the same state. Since the reward is zero, no new information about the objectives is obtained. Therefore, the agent equiprobably selects one of the two auxiliary objectives in this case as well.

As shown above, an auxiliary objective is selected randomly at each iteration. So we have the same situation as in Theorem 3 for $k = 1$. Therefore, the expected number of iterations to find the optimum of OM_d is $O(n \log n)$. ■

To sum up the analysis of the RL based dynamic objective approaches on the OM_d problem, explicit optimization of the target objective used in the MOEA+RL algorithm efficiently works with conflicting auxiliary objectives, while the single-objective EA+RL algorithm gets stuck in presence of these objectives.

V. Simultaneous Optimization of Auxiliary Objectives

Previously, we studied different dynamic objective approaches, namely random and RL based. In this section, we consider a static approach, where no selection of objectives is performed.

Theorem 6: Consider the algorithm that optimizes all auxiliary objectives (OneMax and ZeroMax) and the target objective simultaneously by SEMO until the optimum of the target objective OM_d is found. Then, if initialized by a random individual, this algorithm solves the OM_d problem in $\Theta(n^2 \log n)$ fitness function evaluations in expectation.

Proof: For every two individuals x and x' in the population, it holds that if $\textsc{OneMax}(x) = \textsc{OneMax}(x')$ then $\textsc{ZeroMax}(x) = \textsc{ZeroMax}(x')$. Due to this fact, $\mathrm{OM}_d(x) = \mathrm{OM}_d(x')$, because otherwise one of the individuals dominates another one, so both of them cannot coexist in the same population. The selection we use does not permit two individuals with all the same fitness values, so at most one individual with a given $\textsc{OneMax}(x)$ survives. If $\textsc{OneMax}(x) \neq \textsc{OneMax}(x')$, then either $\textsc{OneMax}(x) < \textsc{OneMax}(x')$ (then $\textsc{ZeroMax}(x) > \textsc{ZeroMax}(x')$) or vice versa, which means that such individuals never dominate each other.

For the upper bound, consider an individual x with the maximum value of OM_d, which is $k < n$. It is selected with probability $p_0 \geq \frac{1}{n}$, and the mutation which increases its $\mathrm{OM}_d(x)$ value happens with $p_1 = \frac{n-k}{n}$. When this happens, the new individual x' will have the maximum possible value of OM_d among the population, which, due to the reasoning above, makes the SEMO selection accept this individual. Thus, when looking at the maximum OM_d values, the OneMax process is modeled. By Theorem 1, the running time is at most $O(\frac{n \log n}{p_0}) = O(n^2 \log n)$.

For the lower bound, note that the population is initialized by a single individual and only single-bit mutations are used. Due to these observations, the population forms a contiguous segment in all three objectives (OneMax, ZeroMax, OM_d). Assume the algorithm starts with $\mathrm{OM}_d(x) = v_0$. If $n - v_0 = \Theta(n)$, then as soon as the value $\mathrm{OM}_d(x) = \frac{n+v_0}{2}$ is reached, the algorithm needs $\Omega(n^2 \log n)$ iterations to optimize OM_d: the probability of selecting the individual with the maximum OM_d is $O(\frac{1}{n})$, and the running time to traverse the last $\Theta(n)$ levels of OneMax is $\Theta(n \log n)$. If, finally, the initial individual is sampled at a distance of $o(n)$ from the optimum, the probability of this to happen is at most $1/2$, so the result for this case does not influence the lower bound, as it is not greater than $O(n^2 \log n)$. ■

VI. Conclusion

We analyzed different approaches of using auxiliary objectives on a model problem with conflicting objectives. The target objective was the generalized OneMax problem, the auxiliary objectives were standard OneMax and ZeroMax.

Dynamic selection of an auxiliary objective, as well as simultaneous optimization of all auxiliary objectives were analyzed. In the case when the target objective is optimized simultaneously with the auxiliary ones, it was shown that dynamic selection allows solving the problem in $O(\min(n, k) \cdot n \log n)$ fitness function evaluations, where n is the problem size and k is the period of optimizing the same objective. Particularly, dynamic selection based on reinforcement learning (the MOEA+RL method) solves the problem in $O(n \log n)$ fitness function evaluations. Simultaneous optimization yields $\Theta(n^2 \log(n))$ evaluations in this case. To obtain the above

results, we proposed a general theorem that is useful for analyzing different auxiliary objective methods. In the case when the target objective is not optimized explicitly, reinforcement based dynamic selection (the EA+RL method) fails to solve the problem. Therefore, it seems that preserving the individual with high value of the target objective is important to mitigate the influence of selection of an improper auxiliary objective.

It is also important to notice that when the conflicting auxiliary objectives were dynamically selected, this often led to significant decrease of the current population size. For the considered problem, such behaviour allowed to solve the problem as efficiently as if there were no conflicting objectives, i.e. in $O(n \log n)$ fitness function evaluations. One may speculate that similar processes may arise while solving the Job-Shop Scheduling problem, where objectives may also be in conflict. This fact together with the explicit usage of the target objective could explain why in practice the MOEA+RL method finds better schedules than EA+RL and simultaneous optimization do.

In the future work, to get a fuller picture, it would be good to analyse the static method, where only the auxiliary objectives are optimized simultaneously without the target one. The stopping criterion for such method is, though, whether the target optimum is found. However, the target value does not influence the rest of the search process at all. This method is a model of the pioneer method of multiobjectivization [11]. This approach was initially proposed for the auxiliary objectives which comprise a decomposition of the target objective, unlike in the considiered problem. We have a hypothesis that for the considered problem this method should not be very efficient, because in our case it has to perform somewhat random search of the proper optimal individual.

VII. Acknowledgements

Arina Buzdalova and Irina Petrova were supported by RFBR according to the research project No. 16-31-00380 mol_a. Maxim Buzdalov was supported by the Government of Russian Federation, Grant 074-U01.

References

[1] A. Auger, A. Auger, and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2011.

[2] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler. On the effects of adding objectives to plateau functions. *IEEE Transactions on Evolutionary Computation*, 13(3):591–603, 2009.

[3] M. Buzdalov, A. Buzdalova, and I. Petrova. Generation of tests for programming challenge tasks using multi-objective optimization. In *Proceedings of Genetic and Evolutionary Computation Conference Companion*, pages 1655–1658. ACM, 2013.

[4] 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.

[5] A. Buzdalova and M. Buzdalov. Increasing efficiency of evolutionary algorithms by choosing between auxiliary fitness functions with reinforcement learning. In *Proceedings of the International Conference on Machine Learning and Applications*, volume 1, pages 150–155, 2012.

[6] 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*, number 8084 in Lecture Notes in Computer Science, pages 300–305. Springer, 2013.

[7] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[8] B. Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14(3):502–525, 1982.

[9] J. Handl, S. C. Lovell, and J. D. Knowles. Multiobjectivization by decomposition of scalar cost functions. In *Parallel Problem Solving from Nature – PPSN X*, number 5199 in Lecture Notes in Computer Science, pages 31–40. Springer, 2008.

[10] M. T. Jensen. Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation: Evolutionary computation combinatorial optimization. *Journal of Mathematical Modelling and Algorithms*, 3(4):323–347, 2004.

[11] J. D. Knowles, R. A. Watson, and D. Corne. Reducing local optima in single-objective problems by multi-objectivization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, pages 269–283. Springer-Verlag, 2001.

[12] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 44–53. Springer-Verlag, London, UK, UK, 2002.

[13] D. F. Lochtefeld and F. W. Ciarallo. Helper-objective optimization strategies for the Job-Shop scheduling problem. *Applied Soft Computing*, 11(6):4161–4174, 2011.

[14] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[15] F. Neumann and I. Wegener. Can single-objective optimization profit from multiobjective optimization? In *Multiobjective Problem Solving from Nature*, Natural Computing Series, pages 115–130. Springer Berlin Heidelberg, 2008.

[16] I. Petrova, A. Buzdalova, and M. Buzdalov. Improved helper-objective optimization strategy for Job-Shop scheduling problem. In *Proceedings of the International Conference on Machine Learning and Applications*, volume 2, pages 374–377. IEEE Computer Society, 2013.

[17] C. Qian, Y. Yu, and Z.-H. Zhou. An analysis on recombination in multi-objective evolutionary optimization. *Artif. Intell.*, 204:99–119, Nov. 2013.

[18] C. Qian, Y. Yu, and Z.-H. Zhou. Subset selection by pareto optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 1774–1782, Cambridge, MA, USA, 2015. MIT Press.

[19] C. Segura, C. A. C. Coello, G. Miranda, and C. Léon. Using multi-objective evolutionary algorithms for single-objective optimization. *4OR*, 3(11):201–228, 2013.

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

Appendix

Consider some auxiliary objective approach, where several objectives, including the target one, are optimized by SEMO. We consider SEMO as a general process Y. There is also an underlying process X optimizing the target objective. A step of this process appears when the individual with the best target objective value is selected to be mutated by SEMO. Notice that the underlying process X in SEMO is essentially the random local search (RLS) on the target objective. Lemma 3 allows estimating the expected running time of the general process provided that it stops when the underlying process finds the optimum of the target objective.

Lemma 3: Let $E(X)$ be the expected number of steps in a random process X. Let Y be a random process that stops simultaneously with X. On each step of Y, a step of X is

performed with probability $p \geq p_0$. Then the following holds: $E(Y) \leq \frac{E(X)}{p_0}$, where $E(Y)$ is the expected number of steps in the process Y.

Proof: $E(X) = \sum_{t=0}^{\infty} t \cdot p_x(t)$, where $p_x(t)$ is the probability that X finishes in t steps. Let Y_i be the number X steps performed in i steps of Y. Then $Y_0 = 0$ and with probability p $Y_i = Y_{i-1} + 1$, otherwise $Y_i = Y_{i-1}$. Assume that X finishes in t_0 steps. Let us estimate the value of $E(t|Y_t = t_0)$. Consider $Y'_i = t_0 - Y_i$. Notice that $Y'_i = 0$ if X has performed t_0 steps. Also $E(Y'_i - Y'_{i+1}) = p \geq p_0$ because $Y'_i - Y'_{i+1} = 1$ with probability p and is zero otherwise. By additive drift theorem [8]:

$$E(t|Y'_t = 0) = E(t|Y_t = t_0) \leq \frac{t_0}{p_0}$$

From this we get that $E(Y) = \sum_{t=0}^{\infty} E_{Y|t} \cdot p_x(t) \leq \sum\limits_{t=0}^{\infty} \frac{t}{p_0} \cdot p_x(t) = \frac{E(X)}{p_0}$. ■

Lemma 3 estimates the expected running time of the general process using the expected running time of the underlying process and the probability that the underlying process was iterated. Lemma 4 allows estimating this probability using the expected population size in SEMO.

Lemma 4: Consider optimization of objectives $c_1, c_2, \ldots, c_m$ by the SEMO algorithm. Let F be the size of the current population. Then the expected value of F is $E(F) = \sum_{i=1}^{n} i \cdot p(i)$, where i is a population size and $p(i)$ is the probability that the size of the current population is i. Let q be the probability that an individual with the best value of c_k is selected from the current population. Then $q = \sum_{j=1}^{n} \frac{p(j)}{j}$, where j is a population size and $p(j)$ is the probability that the population size is j. Then the following holds: $E(F) \cdot q \geq 1$.

Proof: From $i^2 - 2ij + j^2 = (i-j)^2 \geq 0$ follows that $\frac{i}{j} + \frac{j}{i} \geq 2$, which gives that:

$$\begin{aligned} E(F) \cdot q &= \left(\sum_{i=1}^{n} i \cdot p(i)\right) \cdot \sum_{j=1}^{n} \frac{p(j)}{j} = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{i \cdot p(i)p(j)}{j} \\ &= \sum_{i=1}^{n} \left(\frac{i \cdot p(i)p(i)}{i} + \sum_{j=i+1}^{n} p(i)p(j)\left(\frac{i}{j} + \frac{j}{i}\right)\right) \\ &\geq \sum_{i=1}^{n} \left(p(i)^2 + \sum_{j=i+1}^{n} 2p(i)p(j)\right) = \left(\sum_{i=1}^{n} p(i)\right)^2 \\ &= 1. \end{aligned}$$

■

Finally, we can formulate the theorem that allows obtaining the expected running time of an auxiliary objective approach using the expected population size and the expected optimization time of the target objective.

Proof of Theorem 1: Denote SEMO as process Y. Then the underlying process X is the random local search on c_k. Consider the target objective as c_i. From Lemma 3 and Lemma 4 we get that $E(Y) \leq \frac{E(T)}{p_0} \leq \frac{E(T)}{(E(F))^{-1}} = E(T) \cdot E(F)$. ■