Why Parameter Control Mechanisms Should Be
Benchmarked Against Random Variation

Giorgos Karafotias
Computer Science Department
VU University
Amsterdam
Email: g.karafotias@vu.nl

Mark Hoogendoorn
Computer Science Department
VU University
Amsterdam
Email: m.hoogendoorn@vu.nl

A.E. Eiben
Computer Science Department
VU University
Amsterdam
Email: a.e.eiben@vu.nl

Abstract—Parameter control mechanisms in evolutionary algorithms (EAs) dynamically change the values of the EA parameters during a run. Research over the last two decades has delivered ample examples where an EA using a parameter control mechanism outperforms its static version with fixed parameter values. However, very few have investigated why such parameter control approaches perform better. In principle, it could be the case that using different parameter values alone is already sufficient and EA performance can be improved without sophisticated control strategies raising an issue in the methodology of parameter control mechanisms’ evaluation. This paper investigates whether very simple random variation in parameter values during an evolutionary run can already provide improvements over static values. Results suggest that random variation of parameters should be included in the benchmarks when evaluating a new parameter control mechanism.

I. INTRODUCTION

When setting up an evolutionary algorithm (EA) one aspect that needs to be addressed is defining appropriate values for the various parameters of the algorithm. In case inappropriate values are chosen the performance of the EA can be severely degraded. The question whether a certain value of a parameter is appropriate is far from trivial as different phases in an evolutionary run could require different values. In fact, there are two principal options to set such values [4]: (1) trying to find fixed parameter values that seem to work well across the entire evolutionary run (parameter tuning), and (2) finding a suitable control strategy to adjust the parameter values during a run (parameter control). Furthermore, we can distinguish three forms of parameter control: (a) deterministic parameter control, which uses a fixed control scheme without using any input from the state of the process; (b) adaptive parameter control, utilizing information from the state of the process to determine good parameter values, and (c) self-adaptive whereby the parameter values are part of the evolutionary process itself. In the literature a variety of evolutionary algorithms equipped with sophisticated parameter control strategies have been shown to outperform their static counterparts (see e.g. [12], [3] and [18]), and many have acknowledged that dynamically adjusting parameter values is a very good idea (see e.g. [14]).

In the majority of work presenting parameter control mechanisms, the value of the controller is assessed only by comparing its performance to the static version of the EA that keeps parameter values fixed. The motivation of this paper is based on the idea that such performance benefits observed when using parameter control mechanisms over using static parameter values might be in fact a result of simply the variation of the parameter and not the intelligent strategy itself. Some authors have made some weak hints in this direction, see the next Section, however, none have performed a rigorous analysis. If it is possible that variation on its own (without some intelligent strategy) might improve performance, a methodological issue is raised: when evaluating a parameter control mechanism the actual contribution of the intelligent strategy to the performance gain should not be taken for granted but should be explicitly assessed by also including ‘naive variation’ in the set of benchmarks used.

The goal of this paper is to investigate whether (non-intelligent) ‘variation’ alone might indeed improve EA performance as compared to keeping parameter values fixed. To this end, we implement a few simple random methods to vary parameter values during the run of an EA and investigate their impact on a set of standard test problems. In particular, we use a uniform distribution and a Gaussian distribution and compare the resulting EAs with an EA whose parameter values are fixed (by a powerful tuning algorithm) and with an EA whose parameters change by a sine wave based schedule (enabling increasing and decreasing the values).

This paper is organized as follows. Section II explains the motivation in more detail and provides some related work. Thereafter the experimental setup is discussed in Section III whereas the results are presented in Section IV. Finally, Section V concludes the paper and presents avenues for future work.

II. MOTIVATION AND RELATED WORK

It is widely accepted in EC that parameter control is preferable over static parameter values because different parameter values are needed at different stages of the evolutionary process (e.g. switching from global to local search). Additionally, information about the fitness landscape that is accumulated during the search can be used to improve parameter values in the later phases of the process [2]. Several parameter control methods for evolutionary algorithms have been suggested, some literature reviews can be found in [4], [16], [7] and [13].

In many of the studies introducing these parameter control strategies, performance comparisons between the EA using the control mechanism and the equivalent EA with static parameter values are presented as a proof of the controller’s value.
However, it is the usual case that no further investigation is carried out as to how exactly the parameters are varied and to what extent the performance gain is a result of the specific control strategy or the mere fact that parameters simply change during the run, i.e. just adding some variation in the parameter values already brings added value.

The idea that simply changing the values of a parameter, regardless of how that change is done, can result in better performance has been hinted in some previous work. In [17], Spears experiments with self-adaptive operator selection for genetic algorithms. Results show that the GA with random operator selection has similar performance with the self-adaptive GA meaning that it is just the availability of multiple operators that improves performance and not self-adaptation. Randomized values are purposefully used in [5] to set the parameters of different islands for a distributed EA with the rationale that, at each moment during the search process, there will be at least one parameter configuration that will be favorable for further advance.

In this paper we attempt to answer the question whether a random variation of parameter values by itself (with no intelligence, purpose or strategy) can have a positive effect on the performance of an evolutionary algorithm as compared to keeping parameter values fixed. Though theoretical studies on optimal parameter values or ideal control strategies do exist (see e.g. [6], [9], [8] and [11]), we believe that such a theoretical approach here would be impossible or greatly oversimplifying. For this reason we prefer an experimental approach as will be described in the following section.

III. EXPERIMENTAL SETUP

As was explained in the previous sections, the purpose of the experiments presented here is to determine if the mere variation of parameter values (with no particular method or strategy) can result in performance gain for the evolutionary algorithm when compared to keeping parameters fixed. In order to assess the effect of parameter variation isolated from the effect of an “intelligent” control method or strategy we use the most naive parameter variation approach possible, i.e. random variation of parameter values. Keeping all other factors identical, we compare the performance of an evolutionary algorithm when its parameter values are kept fixed during the whole search and when its parameter values vary according to some random distribution. To show the difference between the random variation and a non-random (but certainly not sophisticated) variation approach, an additional approach to vary the parameter values, i.e. a sine-based function, is used which facilitates sequences of increase and decrease of such values.

Before describing the experimental setup, two important points must be emphasized here. First, we are not trying to establish as a general truth that parameter variation will by itself lead to better performance but rather to determine if it can be possible to observe better performance as a result of only the availability or application of multiple parameter values regardless of any control strategy. Second, we do not propose random variation as a parameter control method. The performance comparison between search with static parameter values and with randomly varying parameters aims only at exploring the effect of parameter variation and not in designating a winner.

A. The evolutionary algorithm and the test functions

As an evolutionary algorithm we use a \((\mu + \lambda)\) Evolution Strategy with n-point crossover, Gaussian mutation and tournament selection for both parent and survivor selection. The parameters used in these experiments are the following six:

- population size \(\mu\)
- generation gap \(g\) (\(g\) is the ratio of the number of offspring to the population size)
- number of crossover points \(n\)
- mutation step size \(\sigma\)
- parent selection tournament size \(k_p\)
- survivor selection tournament size \(k_s\)

For test problems we use a set of seven standard continuous optimization test functions (see Table I). All functions are to be minimized in 10-dimensions. All are multimodal except one \((f_3\) Rosenbrock). A single EA run is 10000 function evaluations.

B. Comparison approach

We use the following workflow to facilitate the desired comparison, in step (2)-(4) experimental results are generated and comparisons are made:

1) Tune the parameters values of the ES using a dedicated parameter tuner, resulting in a set of basic parameter values.
2) Add variation to the basic parameter values found under (1) using a Gaussian and uniform distribution with fixed variation values.
3) Instead of using fixed variation values as in (2), try to find the best variation values using a parameter tuning approach given the basic values found under (1).
4) Tune all parameter values (both basic and variation values) at the same time, also include a non-randomized parameter value generator which can express basic sequences of increasing and/or decreasing values.

Each of these steps is discussed more elaborately below.

1) Tuning the ES: As a first step, the ES is tuned for every test function separately (all six parameters are tuned concurrently with one tuning process per problem). For tuning we use Bonesa [15], which is a state-of-the-art parameter tuning method for tuning real valued parameters. This step results in seven parameter vectors \(\vec{p}_i\), one for each problem \(f_i\), \(i = 1...7\), with good static values for each parameter. The ranges and the results of the tuning process are shown in Table II. A single Bonesa tuning run was given a budget of 10000 algorithm tests.
Several width coefficients $d$ are tried, $d = 0.01, 0.02, 0.03, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5, 0.8$. Separate runs are made with each parameter varied alone and all parameters varied together. For every setting (i.e. combination of parameter, distribution and $d$), the ES is run 30 times to derive statistically reliable results.

3) Experiment 2: “Tuning” the range of the variation:
The above process attempts to determine whether adding some variance around the parameter values can result in improved performance, however, only a small number of hand-picked ranges (defined by the values of $d$) are tested. For a more thorough and rigorous test, the rationale of experiment 1 is maintained but we use Bonesa as a search algorithm to find good values for the standard deviation of the Gaussian distribution. Thus now, for problem $i$, values for parameter $j$ are drawn from a normal distribution $N(\bar{p}_i(j), d \cdot \bar{p}_i(j))$ with every $\sigma_i^j$ being derived through a search process by Bonesa (one tuning process per problem was executed that concurrently tuned the deviations of all six parameters). If the tuning process of Bonesa for a $\sigma_i^j$ converges to a non-zero value, that would indicate that some random variation is indeed beneficial. A much longer (25000 algorithm tests) tuning process was used for this experiment to increase the reliability of the results.

Due to time limitations, this experiment was performed only for function $f_1$.

4) Experiment 3: “Tuning” all the settings of the variation:
As a final test we make a fair comparison between the performance of the ES using static parameter values and its performance using varying values. Since the static values were derived through a tuning process, in order to make the comparison fair, the settings that determine the varying values must also be calibrated in equal terms. Thus, an identical tuning process (using Bonesa with the same budget of 10000 algorithm tests) is used. Here, except for the normal and uniform random distributions employed previously, we also use an approach based on a sine wave which is able to generate sequences of increasing and/or decreasing parameter values. For each variation mechanism, the following settings are tuned:

- **Gaussian**: for each problem $i$, a tuning process calibrates for each parameter $j$ the mean $m_i^j$ and standard deviation $\sigma_i^j$ of the normal distribution. Thus, each tuning process tunes 12 values.

- **uniform**: for each problem $i$, a tuning process calibrates for each parameter $j$ the minimum $l_i^j$ and the width $w_i^j$ of the range from which values are drawn. Thus each tuning process tunes 12 values.

- **sine**: for each problem $i$, a tuning process calibrates for each parameter $j$ the amplitude $A_i^j$, frequency $f_i^j$, angular frequency $\omega_i^j$ and phase $\phi_i^j$ that define a sine wave used as a deterministic schedule. Thus, each tuning process tunes 24 values.

After the tuning is complete, for each problem and variation setting combination, the ES is run 30 times to derive statistically reliable results.
IV. Results and Analysis

The results of experiment 1 are presented in Table III. The table shows the performance of the three ES variants that have been run in this first experiment: with tuned static parameter values, with values drawn from a Gaussian distribution (for various values of \(d\) defining the standard deviation) and with values drawn from a uniform distribution (for various values of \(d\) defining the width). Emphasized numbers indicate a significant improvement over fixed parameters.

These results suggest that only varying the value of the parameter around the static value, without any strategy or purpose, may possibly lead to better performance. For 4 out of 7 problems and for 9 out of 49 combinations of problem and parameter, there exists some kind of variation that may significantly improve performance. The Gaussian distribution appears more often, perhaps indicating that mild noise is preferable, however, there are also cases where drawing values from a uniform distribution is beneficial when compared to keeping parameters fixed. An important observation is that, in most cases where changing parameter values can be beneficial, performance improves as the range of the change becomes wider, with the best results achieved when the range is 80% of the center value. Figure 2 shows some examples where performance improves with variation and how this performance gain is influenced by the variation width \(d\).

The parameter that has most often a positive response is the mutation step size \(\sigma\) but there are also cases where varying the population size and number of crossover points may result in improvement. Finally, for function \(f_2\) varying all parameter values significantly increases performance while varying each parameter independently does not.

The results of experiment 2 are shown in Table IV. The best three vectors resulting from the tuning process are presented, each vector defining the standard deviations of the Gaussian distributions from which parameter values are drawn (the results only concern \(f_1\)). For all parameters, except \(g\), the tuning process converged to deviation values far from zero, indicating that the existence of variation (non-zero deviation) was preferred by the tuning process. Using the best vector of deviations (and tuned vector \(\mu_i\) for mean values), the ES was run 30 times with Gaussian variation of all parameters. A comparison with keeping the parameters static (to tuned values \(\mu_i\)) is shown in Figure 1. Also, the two best (for this problem) cases of experiment 1 are considered, namely the case with variation of all parameters with a width of \(d = 0.3\) and varying only \(\sigma\) with a width of \(d = 0.8\). The tuned deviations of this experiment produce better results than static and than varying all parameter values with \(d = 0.3\) (experiment 1). However, they are not better than varying only \(\sigma\). This may be due to the fact that varying one of the parameters except \(\sigma\) has a very detrimental effect (for this problem and EA) but the tuning process was not able to set the parameter’s deviation to zero.

The results of experiment 3 are presented in Table V, showing the performance of the ES using static parameter values, and completely tuned variations (Gaussian and uniform distributions as well as a completely tuned sine function). Underlined values denote the best average performance and bold values indicate performance not significantly worse than the best. We can again see that varying the parameter values can result in better performance in some cases. However, tuning the settings of the random variations (Gaussian and uniform) did not produce any improvement compared to the results acquired by experiment 1 with handpicked \(d\) values (see Table III). For function \(f_5\), the performance of the tuned Gaussian is much worse than the performance acquired simply by setting the deviation of all parameters to \(0.8 \cdot \mu_i(j)\). Furthermore, for functions \(f_2\) and \(f_4\), while experiment 1 showed improvement when varying all parameters, here we see worse performance. It might be that the task of the tuning process is too tough when tuning the settings of the variation mechanisms due to the number of values tuned: for the Gaussian and uniform distributions there are double the settings compared to tuning static values (two settings per parameter) while for the sine wave the factor is four. Consequently, though the same tuning effort was spent for static values and variation mechanisms, the outcomes are unbalanced.

The tuned sine wave variation performs the best with problem \(f_2\); the corresponding parameters variation is shown in Figure 3. Except for \(\mu\), the variation of all other parameters is just a very fast oscillation within a certain range, showing that tuning resulted in a process also resembling random instead of a more “meaningful” schedule that could be expressed with a sine wave.

![Table IV](image-url)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Vector 1</th>
<th>Vector 2</th>
<th>Vector 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mu)</td>
<td>17.925</td>
<td>19.616</td>
<td>25.785</td>
</tr>
<tr>
<td>(g)</td>
<td>0.025</td>
<td>0.06</td>
<td>0.03</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>0.862</td>
<td>1.049</td>
<td>0.897</td>
</tr>
<tr>
<td>(N)</td>
<td>2.692</td>
<td>1.064</td>
<td>2.771</td>
</tr>
<tr>
<td>(k_p)</td>
<td>25.464</td>
<td>21.966</td>
<td>25.662</td>
</tr>
<tr>
<td>(k_s)</td>
<td>39.978</td>
<td>22.644</td>
<td>38.916</td>
</tr>
</tbody>
</table>

![Fig. 1](image-url)

Fig. 1. Experiment 2: A comparison of the performance when keeping all parameter values with tuned static parameter values (ST), when varying all parameters with a Gaussian distribution with a tuned standard deviation (TD), when varying all parameter values with a Gaussian distribution with \(d = 0.3\) (VA) and when varying only \(\sigma\) with a Gaussian distribution with \(d = 0.8\). The function used is \(f_1\) (Ackley), which is a minimization problem.
| $|\mu|$ | $g$ | $\sigma$ | $N$ | $k_p$ | $k_s$ | $f_2$ | $f_1$ (Ackley) | $f_3$ (Rastrigin) | $f_4$ (Griewank) | $f_5$ (Shkel) | $f_6$ (Bohachevsky) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8.64 | 7.27 | 8.73 | 6.87 | 7.51 | 7.68 | 8.22 | 7.69 | 8.18 | 7.85 | 9.41 | 10.30 | 11.25 | 10.25 | 11.19 |
| 8.30 | 7.53 | 7.51 | 8.05 | 7.74 | 8.34 | 8.46 | 8.37 | 9.34 | 13.99 | 8.77 | 7.45 | 7.41 | 8.60 | 8.20 |
| 7.32 | 7.32 | 7.32 | 6.83 | 6.33 | 6.19 | 7.69 | 8.26 | 12.37 | 7.32 | 7.32 | 6.18 | 6.18 | 5.71 | 5.94 |
| 7.32 | 7.12 | 7.32 | 7.32 | 7.31 | 7.25 | 6.69 | 8.85 | 8.25 | 7.52 | 7.32 | 7.32 | 7.32 | 7.32 | 7.32 |
| 7.32 | 7.32 | 7.84 | 7.51 | 7.21 | 7.66 | 7.53 | 7.66 | 8.05 | 8.90 | 7.42 | 7.42 | 7.42 | 7.42 | 7.42 |
| 8.38 | 7.71 | 7.28 | 6.98 | 5.71 | 5.75 | 6.28 | 7.22 | 15.53 | 40.55 | 8.09 | 8.44 | 8.70 | 9.59 | 7.50 |
| 9.82 | 7.27 | 8.27 | 8.73 | 5.71 | 7.73 | 6.78 | 8.22 | 7.69 | 8.18 | 7.85 | 9.41 | 10.30 | 11.25 | 10.25 |
| 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 |
| 7.56 | 7.56 | 9.99 | 7.73 | 7.67 | 8.04 | 10.09 | 7.96 | 10.53 | 8.99 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 |
| 7.56 | 7.56 | 7.56 | 7.70 | 7.70 | 7.96 | 7.96 | 7.61 | 7.54 | 9.52 | 7.56 | 7.56 | 7.56 | 7.56 | 7.56 |
| 8.22 | 10.69 | 8.71 | 8.13 | 7.72 | 8.21 | 7.80 | 7.39 | 7.12 | 7.05 | 8.09 | 8.48 | 8.16 | 8.11 | 7.83 |

**TABLE III.** Results of experiment 1. The left part is for the Gaussian distribution and the right for the uniform. There is a suitable for every function and distribution combination. For each subject, in every line it is denoted which parameter is varied. The first column of each suitable shows the performance when the parameter is kept static and the subsequent columns show the performance when the parameter is varied with the corresponding value of $d$. All numbers are averages over 30 runs. Emphasized values show performance that is significantly better than static (with 0.95 confidence). All functions are to be minimized.
Fig. 2. Four cases from the results of experiment 1. Each subgraph shows the performance when varying a parameter (or all) according to a random distribution. The x-axis is the width $d$ of the distribution. The horizontal dashed line shows the performance when keeping the parameter values static to the tuned values. The caption of each subgraph lists the test function, which parameter is varied and the type of the random distribution. Lower values are better for all functions.

Fig. 3. The parameter values over time when using the sine wave with the tuned settings from experiment 3 with $f_7$. Each subgraph shows the values of a parameter over the generations.
TABLE V. RESULTS OF EXPERIMENT 3. FOR EACH PROBLEM, THE PERFORMANCES USING THE VARIATION METHODS WITH TUNED SETTINGS AND THE PERFORMANCE KEEPING THE VALUES FIXED TO THE TUNED VALUES ARE SHOWN. LOWER VALUES ARE BETTER. FOR EACH FUNCTION, UNDERLINED VALUES DENOTE THE BEST AND BOLD VALUES DENOTE PERFORMANCE NOT SIGNIFICANTLY WORSE THAN THE BEST.

<table>
<thead>
<tr>
<th></th>
<th>f1</th>
<th>f2</th>
<th>f3</th>
<th>f4</th>
<th>f5</th>
<th>f6</th>
<th>f7</th>
</tr>
</thead>
<tbody>
<tr>
<td>gaussian</td>
<td>1.74</td>
<td>14.57</td>
<td>9.58</td>
<td>15.88</td>
<td>14.28</td>
<td>6.18</td>
<td>-1.26</td>
</tr>
<tr>
<td>uniform</td>
<td>3.08</td>
<td>12.17</td>
<td>13.28</td>
<td>18.34</td>
<td>21.76</td>
<td>3.54</td>
<td>-2.24</td>
</tr>
<tr>
<td>static</td>
<td>3.21</td>
<td>48.37</td>
<td>38.76</td>
<td>14.13</td>
<td>15.40</td>
<td>17.14</td>
<td>-3.08</td>
</tr>
<tr>
<td>sine</td>
<td>2.67</td>
<td>7.32</td>
<td>7.56</td>
<td>11.51</td>
<td>16.16</td>
<td>0.87</td>
<td>-1.97</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS AND FUTURE WORK

In this paper we have investigated the effect of randomly changing the values of an evolutionary algorithm’s parameters. To be specific, we put forward the assumption that random variation, without intelligence or strategy, can improve EA performance, compared to keeping parameters fixed, simply by making multiple parameter values available to the evolutionary process. To test this hypothesis we performed three separate experiments where the effect of randomly varying the parameter values was examined. All three experiments showed that it is indeed possible to significantly improve the performance of an evolutionary algorithm by randomly changing its parameter values.

The results of this paper raise an important issue in methodology. It is common practice in literature that presents parameter control mechanisms to evaluate the controller by performing a comparison to the equivalent EA with fixed parameters. However, as the results of this paper show, observing an improvement in such a comparison does not necessarily show that the controller is good as it is not shown whether the observed improvement is a consequence of the intelligent control strategy itself or merely the variation of the values. We believe that a complete evaluation of a control mechanism should also include an analysis of how the parameters are varied during a run and we suggest that a “naive” variation scheme for the same parameters should be included in the baseline benchmarks.

Future work will focus on making a comparison between sophisticated parameter control approaches and the non-sophisticated random variation approach presented in the experimental part of this paper to investigate the differences between the two in terms of performance.

REFERENCES


