Costs and Benefits of Tuning Parameters of Evolutionary Algorithms

Volker Nannen, S.K. Smit, and A.E. Eiben

Vrije Universiteit Amsterdam
vnannen@gmail.com,{sksmit,gusz}@few.vu.nl

Abstract. We present an empirical study on the impact of different design choices on the performance of an evolutionary algorithm (EA). Four EA components are considered—parent selection, survivor selection, recombination and mutation—and for each component we study the impact of choosing the right operator, and of tuning its free parameter(s). We tune 120 different combinations of EA operators to 4 different classes of fitness landscapes, and measure the cost of tuning. We find that components differ greatly in importance. Typically the choice of operator for parent selection has the greatest impact, and mutation needs the most tuning. Regarding individual EAs however, the impact of design choices for one component depends on the choices for other components, as well as on the available amount of resources for tuning.

1 Introduction

Evolutionary Algorithms (EA) form a class of search methods that work by incrementally improving the quality of a set of candidate solutions by variation and selection [5]. The most important *components* of EAs are thus recombination and mutation (umbrella term: variation), parent selection, and survivor selection. To obtain a working EA, each component needs to be instantiated by a specific *operator*, e.g., the one-point crossover operator for the recombination component. Furthermore, an EA has *parameters* that need to be instantiated by a specific *parameter value*, e.g., 0.5 for the crossover rate. In this paper we maintain the distinction between components and parameters and say that the instantiation of EA components by concrete operators specifies a particular EA, e.g., uniform crossover, bit-flip mutation, random uniform parent selection and k-tournament survivor selection. Further details regarding the parameters do not lead to a different EA, only to variants of the one defined by the operators.[1] A complete EA design includes the definition of an EA (operators for its components) and the specification of a particular variant of it (values for its parameters).

Setting EA parameters is commonly divided into two cases, parameter tuning and parameter control [3,4]. In case of parameter control the parameter values are changing during an EA run. This requires initial parameter values and suitable

---

[1] Alternatively, components & operators could also be called symbolic parameters & values, and we could say these values only define different EA variants.

control strategies, which in turn can be deterministic, adaptive or self-adaptive. The problem of parameter tuning is hard because for any given application there is a large number of options, but only little knowledge about the effect of EA parameters on EA performance. EA users mostly rely on conventions (mutation rate should be low), ad hoc choices (why not use uniform crossover), and experimental comparisons on a limited scale (testing combinations of three different crossover rates and three different mutation rates). Here we address the problem of parameter tuning. Our main research questions are:

1. How does the choice of operator for each component contribute to EA performance? To this end we compare the absolute performance achieved with different combinations of operators.
2. The parameters of which EA component need the most tuning? For this question we measure the amount of information needed to tune the free parameter(s) of each operator (e.g., crossover rate or tournament size).

For a systematic exploration of the space of EA configurations we use exhaustive search for the combination of operators, and Relevance Estimation and Value Calibration (REVAC) to tune the free (numeric) parameters. REVAC is an Estimation of Distribution Algorithm [14] that tunes an EA by optimizing marginal probability distributions over the free parameters [16,15]. Starting from a set of uniform distributions and an initial drawing of 100 vectors of random parameter values, REVAC iteratively generates new marginal distributions of increasing expected EA performance by *drawing* a new vector of parameter values from the current distributions, *evaluating* the vector by measuring the performance of the EA with these values, *updating* all marginal distributions based on this evaluation, and *smoothing* the updated distributions. Smoothing is a unique feature of REVAC that forces all marginal distributions to approach the maximum Shannon entropy distribution for a given EA performance. This maximized Shannon entropy is independent from the computational cost of any particular tuning method and can be used as a general estimator of the minimum amount of information required to reach a certain level of EA performance. Hence, it can be regarded as a general indicator of how difficult it is to tune a certain EA parameter, and how relevant it is to overall EA performance.

Related work includes the general discussion of EA design [2] and parameter setting [12], in particular within parameter tuning as defined in [3,1,4]. Throughout the relevant literature we find that the costs of tuning parameters is largely ignored. Notable exceptions are the theoretical considerations of [17] and [9], as well as the systematic parameter sweeps of [11,21,20] and the statistical analysis of parameters by [6]. In the general field of experimental design, a paradigm shift that emphasizes a low cost of tuning over the performance of optimal parameter values was due to [22]. In our field, [7] proposes a meta-GA approach in which both EA components and EA parameters are tuned and shows the importance of the right choice for the GA operators. [20] shows how parameter sweeps can be used for robustness and correlation analysis. [18] embed sequential parameter optimization in a wider framework of experimental EA design.

2   Experimental Setup

For a clear discussion we distinguish three different layers in the analysis of an
EA: the problem/application (here: fitness landscapes created by a generator),
the problem solver (here: an EA), and the method for tuning the problem solver
(here: REVAC). For an unbiased study we use independent software implementa-
tions for each layer and combine them through simple interfaces. For the problem
layer we use a generator of real-valued fitness landscapes that are formed by the
max-set of Gaussian curves in high dimensional Cartesian spaces [8]. Where a
Gaussian mixture model takes the average of several Gaussians, a max-set takes
their enveloping maximum, giving full control over the location and height of all
maxima. For the implementation we followed [19] on rotated high dimensional
Gaussians, and used 10 dimensions, 100 Gaussians, and the same distributions
over height, location, and rotation of these Gaussians as specified in the exem-
plary problem sets 1–4 of [8]. These sets offer an increasing amount of exploitable
structure to the EA. Set 1 has the least structure, with peaks of different height
scattered at random, while set 4 is the most structured, with peaks that get
higher the closer they get to the origin. For each set, different landscapes are
created by passing a different random seed to the generator. Initialization of all
EA populations is uniform random in the domain of the fitness landscapes. The
optimal fitness value is 1 on each problem instance and the condition for success-
ful termination is defined as "fitness > 0.9999" or 10,000 fitness evaluations".

For the EAs we use the Evolutionary Computation toolkit in Java (ECJ)
[13], which allows the specification of a fully implemented EA through a simple
parameter file, including the choice of operator for each component and the
values for the free parameters. The ECJ offers several operators for each EA
component, cf. Table 1. For any given EA, the population size parameter is
always present. Most operators have zero or one free parameter. One operator
has 2 free parameters—Gaussian(σ, p) with parameters σ for step size and p
for mutation probability, which takes the value 1 in case of Gaussian($\sigma, 1$). Due
to technical details of the ECJ, only 10 different combinations of parent and
survivor selection operators are possible.[2] With 4 operators for recombination
and 3 operators for mutation, we have 120 combinations of operators, of which
6 with 2, 33 with 3, 53 with 4, 25 with 5, and 3 with 6 free parameters.

The performance of an EA with a given set of parameter values is measured in
three different ways: SR (Success Rate, percentage of runs with fitness > 0.9999),
MBF (Mean Best Fitness of all runs), and AES (Average number of Evaluations
to Solution of successful runs; undefined when SR = 0). Each EA is tuned 5
times on each of the 4 problem sets. During each tuning session on a given set
REVAC generates 1,000 different vectors of parameter values. Each vector of
values is written to the ECJ configuration file, together with the specification of
the operators and the problem generator. The resulting EA is evaluated on 10
different instances of the problem set, generated by different random seeds.

---

[2] Arguably, (μ, λ) and $(\mu + \lambda)$ define both parent *and* survivor selection. Here we clas-
sify them under survivor selection because that is what the parameter λ influences.

Table 1. EA components, operators, and parameters used in this study

Component	Operator	Parameter(s)
		population size μ
parent	tournament	parent tournament size
selection	best selection	number n of best
	random uniform	-
	fitness proportional	-
survivor	generational	-
selection	tournament	survivor tournament size
	random uniform	-
	(μ, λ)	λ
	$(\mu + \lambda)$	λ
recombination	none	-
	one-point	crossover probability
	two-point	crossover probability
	uniform	crossover probability
mutation	reset (random uniform)	mutation probability
	Gaussian$(\sigma, 1)$	step size
	Gaussian(σ, p)	step size, mutation probability

Notes. We follow the naming convention of the ECJ.

For each REVAC tuning session and each EA, the performance after n evaluations is the best performance measured after evaluating n vectors of parameter values. The average performance after n evaluations is averaged over multiple tuning sessions on the same EA. We define *near best performance* as the average performance after 1,000 evaluations minus 5%. If n is the lowest number of vectors for which the average performance exceeds this value, then we say that REVAC needs n evaluations to tune the EA to near best performance. Section 3 uses this to study the impact of choosing an operator for each component.

In Section 4 we analyze the cost and benefits of tuning per EA component. REVAC continuously maximizes the Shannon entropy of the marginal distributions that it optimizes during a tuning session. This maximized Shannon entropy provides a generic information-theoretic measure of the minimum amount of information needed per parameter to reach a given performance level. The differential Shannon entropy H of a probability density function D over the continuous interval $[a, b]$ is commonly defined as

$$H(D_{[a,b]}) = -\int_a^b D(x) \log_2 D(x) \, dx.$$

The sharper the peaks of a probability density function, the lower its Shannon entropy. In order to compare the entropy of distributions that are defined over different parameter intervals in a meaningful way, we normalize all parameter intervals to the interval $[0, 1]$ before calculating the Shannon entropy. In this way the initial uniform distribution has a Shannon entropy of zero, and any other distribution has a negative Shannon entropy $H(D_{[0,1]}) < 0$.
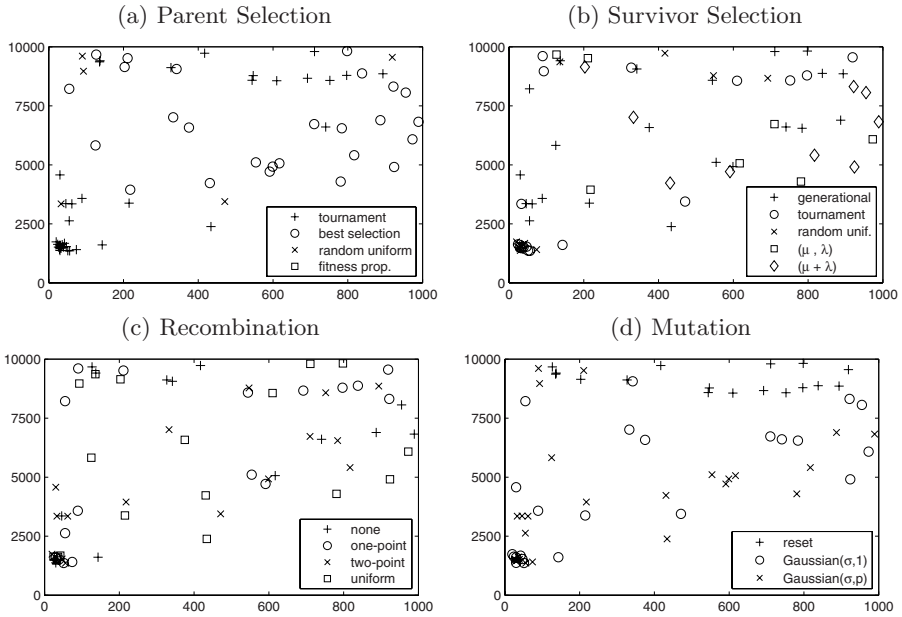
Fig. 1. Near best performance in AES against cost of tuning, by EA component
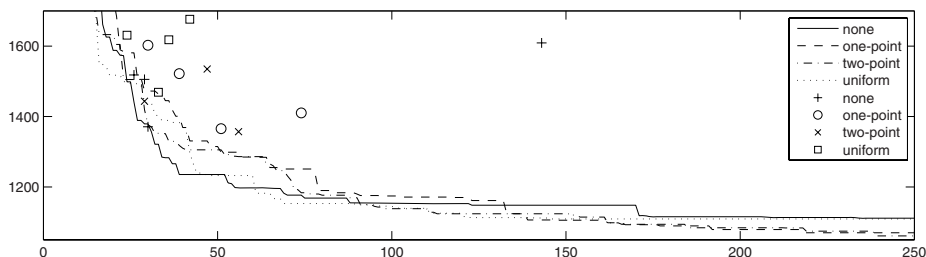
3 How Does the Choice of Operator Per Component Contribute to Performance?

Due to space limitations we only present data on one problem set (no. 4) and one performance measure. We choose to report on the AES, because it only yields 67 data points (those 67 EAs with SR > 0 for which the AES could be calculated). MBF and SR require 120 data points, making the plots less transparent. The four scatter plots in Figure 1 show the performance of these 67 EAs after tuning, and the cost of tuning, averaged over 5 tuning sessions per EA. The y-axes show the near best performance in AES. The x-axes show the number of REVAC evaluations needed to tune the EA to this performance. Each plot shows the same EAs but labels them according to the operator choice for a different component. To read the full specification of an EA, one needs to look at the same location in all four plots. Table 2 shows the near best performance in AES per operator, averaged over those EAs that have this operator and terminated with success.

The choice of operator for the parent selection component has the strongest effect on EA performance. The 16 EAs that are clustered together in the lower left of each plot of Figure 1 display the best performance and the lowest number of evaluations needed to reach this performance. These EAs all use tournament selection for parent selection, either tournament selection or random uniform selection for survivor selection, any recombination operator, and either Gaussian(σ, p) or Gaussian($\sigma, 1$) for mutation. On the other hand, those 53 EAs that never terminated with success share one common feature, namely a lack

**Table 2.** Average near best performance in AES per operator

parent selection		survivor selection		recombination		mutation	
random unif.	9581	random unif.	7039	none	7994	Gaussian(σ, p)	6056
tournament	4514	tournament	6332	one-point	7736	Gaussian$(\sigma, 1)$	6891
best select.	7661	generational	8299	two-point	7053	reset	9633
fitness prop.	-	(μ, λ)	7943	uniform	7325		
		$(\mu + \lambda)$	7386				

Fig. 2. Impact of recombination operators on AES and cost of tuning

of selection pressure. In particular, EAs with random uniform or fitness proportional selection for parent selection almost never terminate with success unless combined with strong survivor selection pressure.

Of the two variation components, the choice of mutation operator has the stronger effect on EA performance, as can be seen from the differences in Table 2. On this problem set reset mutation is the worst mutation operator, and non-standard Gaussian(σ, p) mutation is superior to Gaussian$(\sigma, 1)$ both in terms of performance and cost of tuning. The latter may come as a surprise, since the additional free parameter for mutation probability increases the parameter search space. We conclude that the tuning cost of different operators is not additive, and that the tuning cost of an operator can only be evaluated in the context of the overall EA composition.

While choosing the recombination operator has the least effect on EA performance, it demonstrates how the choice of operator can depend on the available resources for tuning. Figure 2 enlarges the lower left corner of Figure 1c, overlaid by four graphs that show the evolution of the average performance of 4 EAs with tournament selection for both parent and survivor selection, Gaussian(σ, p) mutation, and four different recombination operators. 20 tuning sessions were used for each graph. While any recombination operator eventually outperforms no recombination, an EA with no recombination consistently outperforms EAs with recombination after about 30–40 parameter vectors have been evaluated, and it has at least average performance for anything under 100 evaluated parameter vectors. Recombination behaves similar over a wide range of operator choices for the other components and over all 4 problem sets. All in all, for recombination, the choice of operator can clearly depend on the amount of effort that can be invested in tuning.
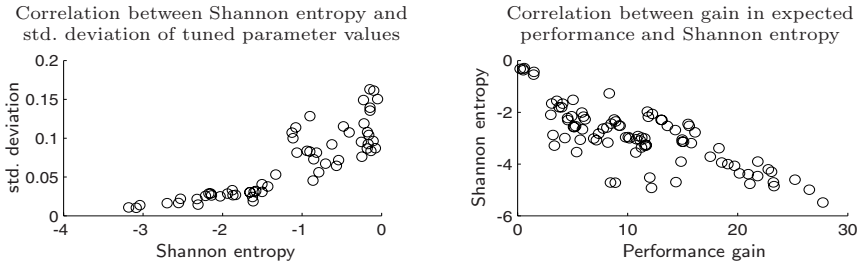
Fig. 3. Correlations with Shannon entropy

4   Which EA Component Needs the Most Tuning?

The previous section related the performance of the near best parameter vector to the number of REVAC evaluations needed to find this vector and to achieve this performance. This section takes a rather unconventional approach based on the average performance when parameter values are drawn from a probability distribution, namely those created by REVAC after 500 evaluations. To calculate the performance gain achieved by tuning, this average performance is compared to the average EA performance when parameter values are drawn from the uniform distribution. All results are averaged over 5 REVAC tuning sessions of an EA on each of the 4 problem sets, 20 tuning sessions per EA. In order to extend our analysis to all 120 EAs, we use the Mean Best Fitness that an EA achieves at termination (successful or not), rather than the AES.

Shannon entropy measures the amount of information that a probability distribution provides on its random values. By definition, the lower the Shannon entropy of the maximum entropy distribution that achieves a given expected EA performance, the finer the parameter value has to be tuned in order to achieve that expected performance. This is demonstrated in Figure 3. The left scatter plot shows the correlation between the Shannon entropy of the marginal distribution over the mutation probability and the standard deviation of the best found parameter values. The x-axis shows the Shannon entropy as estimated by REVAC. The y-axis shows the average of the standard deviation of the 5 best found values for each set. The correlation coefficient is 0.8. The point here is that if the maximum entropy distribution has a higher Shannon entropy, there is less certainty on the precise parameter value, something that can otherwise be expensive to assess.

The right scatter plot of Figure 3 shows a clear correlation between a gain in expected MBF and the Shannon entropy of the maximum entropy distributions that REVAC has estimated after 500 evaluations. The x-axis shows the average performance gain in percent. The y-axis shows the Shannon entropy of the estimated distributions, summed over all tuneable parameters of the EA. Note that no EA lies above the main diagonal, which shows that there is a minimum information cost for every percent point of increase in expected performance,

Table 3. Entropy per EA component & population size aggregated over all EAs

Component & pop. size	Correl. with MBF gain		Shannon Entropy			Median Sha. Entropy per Component & pop. size
	Correl.	p-value	max	median	min	
1) pop. size	-0.3	0.002	0	-0.8	-1.5	
2) parent sel.	-0.3	0.069	0	-0.7	-3.7	
3) surviv. sel.	-0.5	0.002	0	-0.3	-1.2	
4) recombin.	-0.3	0.004	0	-0.1	-1.0	
5) mutation	-0.6	0	-0.2	-1.5	-4.6	
entire EA	-0.8	0	-0.3	-2.9	-5.1	
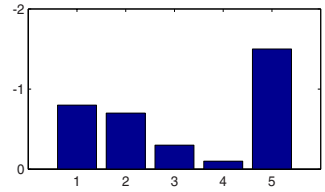
regardless of the EA specifications. Of those EAs that lie significantly below the diagonal, most use tournament selection for both parent and survivor selection. By 500 REVAC evaluations, their MBF had long been maximized. Further tuning only improved their AES, distorting their performance gain-entropy ratio.

Does the strong correlation between total Shannon entropy and the gain in expected performance carry over to individual EA components? The first two numeric columns of Table 3 show the correlation coefficient for each component and its p-value, i.e., the probability to observe this value if the true coefficient is zero. Only EAs with a tunable operator were considered for the respective component. The correlation is generally weaker, with coefficients up to -.3. In other words, the question which component needs tuning in order to improve the performance of a particular EA depends much on the EA in question.

With respect to the average Shannon entropy per component, we see that not all components require the same amount of tuning. The right numeric columns in Table 3 show the maximum, median, and minimum Shannon entropy that we observed for each component (and the population size) when instantiated with an operator that needs tuning. The bar diagram to the right of Table 3 allows a visual comparison of this average median Shannon entropy. Such a skewed distribution of a need for tuning is commonly known as *sparcity of effects*.

Typically, mutation requires the highest amount of tuning, and recombination the least. This rule has many exceptions, as can be concluded from the low correlation coefficients. While the relative order of Shannon entropy per component depends much on the EA in question, consistent patterns can be detected for small groups of EAs. Take for example the two EAs with tournament selection for both parent and survivor selection, Gaussian$(\sigma, 1)$ mutation and either one-point, two-point or uniform crossover. We find that the Shannon entropy for mutation has the unusually high Shannon entropy of around -.2, while the parent selection operator has a low Shannon entropy below -3. When combining the same selection operators with other recombination or mutation operators, we find that the Shannon entropy for parent selection is back to normal levels, while it is still comparatively high for mutation. Another example is recombination, which only exhibits a low Shannon entropy for uniform crossover in combination with either $(\mu + \lambda)$, or (μ, λ). Such irregular patterns are consistent

over different problem sets and seem to be inherent to specific combinations of
EA components.

5   Conclusions and Further Work

This paper introduces a novel approach to EA design that emphasizes the cost
of tuning. To understand how this cost depends on the choice of operator per EA
component, we combined an enumerative search over operators with REVAC for
tuning their parameters. Our experiments revealed a number of notable insights.

Our tests confirmed the common wisdom that the choice of operator for one
EA component depends on the choice of operator for the other components.
Of all components, the choice of operator for parent selection has the biggest
impact on EA performance. Furthermore, EAs differ greatly in the amount of
tuning needed to reach a given performance, and this tuning cost depends on
the overall setup of the EA, rather than the number of free parameters. With
regard to recombination, we found that the best EA setup depends on the time
and effort one can permit to tune the EA.

To measure the need for tuning per component we use the Shannon entropy
of maximum entropy distributions as estimated by REVAC, which expresses
the minimum amount of information that is needed to achieve a given expected
EA performance. It is a generic information-theoretic measure that is indepen-
dent of any particular tuning algorithm. Inspired by theoretical considerations,
it was validated by a strong correlation with the standard deviation of best solu-
tions found during multiple tuning sessions. Based on this measure we observed
that the need for tuning follows a skewed distribution, and that while total
Shannon entropy is strongly correlated with performance gain, the correlation
per component is weak. The question which component needs the most tuning
depends on the precise composition of an EA and can not be answered on a
general level. It needs to be addressed by the operational analysis of individual
EAs. Also, we recommend that a scientific discussion of individual operators ad-
dresses its effect on the overall tunability of an EA and on the need for tuning per
component.

Regarding the scope of our results, an empirical study can only use a limited
set of test problems, and strictly speaking our findings are only proven for our
test problems. However, we consider it unlikely that the complex picture that
has emerged here is an artefact of the test problems. What remains to be studied
is whether the way in which the need for tuning per component depends on the
choice of operators is different on other complex fitness functions.

Last but not least, this paper serves as a demonstration of an open source
tool kit that can be used to analyze the need for tuning of EA parameters
on a given application. Further documentation, Matlab implementations and
graphical demonstrations of REVAC are available on the web sites of the
authors[3].

---

[3] http://www.few.vu.nl/~volker/revac and http://www.few.vu.nl/~gusz

References

1. Birattari, M.: The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. PhD thesis, Université Libre de Bruxelles (2004)
2. Czarn, A., MacNish, C., Vijayan, K., Turlach, B.A., Gupta, R.: Statistical Exploratory Analysis of Genetic Algorithms. IEEE Trans. Evol. Comp. 8(4), 405–421 (2004)
3. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Trans. Evol. Comput. 3(2), 124–141 (1999)
4. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter Control in Evolutionary Algorithms. In: Lobo, et al. (eds.) [12], pp. 19–46
5. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
6. François, O., Lavergne, C.: Design of Evolutionary Algorithms—A Statistical Perspective. IEEE Trans. Evol. Comput. 5(2), 129–148 (2001)
7. Friesleben, B., Hartfelder, M.: Optimization of Genetic Algorithms by Genetic Algorithms. In: Albrecht, R.F., Reeves, C.R., Steele, N.C. (eds.) Artificial Neural Networks and Genetic Algorithms, pp. 392–399. Springer, Heidelberg (1993)
8. Gallagher, M., Yuan, B.: A General-Purpose Tunable Landscape Editor. IEEE Trans. Evol. Comput. 10(5), 590–603 (2006)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Boston (1989)
10. Grefenstette, J.J.: Optimization of Control Parameters for Genetic Algorithms. IEEE Trans. Syst. Man Cybernet. 16(1), 122–128 (1986)
11. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
12. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence. Springer, Heidelberg (2007)
13. Luke, S., et al.: A Java-based Evolutionary Computation Research System, http://www.cs.gmu.edu/~eclab/projects/ecj/
14. Mühlenbein, H., Höns, R.: The Estimation of Distributions and the Minimum Relative Entropy Principle. Evolutionary Computation 13(1), 1–27 (2005)
15. Nannen, V., Eiben, A.E.: Efficient Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: IEEE Congress on Evolutionary Computation (CEC), Piscataway, NJ, USA. IEEE Press, Los Alamitos (2007)
16. Nannen, V., Eiben, A.E.: Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: Veloso, M.M., et al. (eds.) Proc. of the 20th Int. Joint Conf. on Artif. Intell., IJCAI 2007, pp. 1034–1039. AAAI Press, Menlo Park (2007)
17. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In: Grefenstette, J.J. (ed.) Proc. of the 2nd Int. Conf. on Genetic Algorithms on Genetic algorithms and their application, pp. 224–230. L. E. Associates (1987)
18. Preuss, M., Bartz-Beielstein, T.: Sequential Parameter Optimization Applied to Self-adaptation for Binary-coded Evolutionary Algorithms. In: [12], pp. 91–119
19. Rudolph, G.: On Correlated Mutations in Evolution Strategies. In: Männer, R., Manderick, B. (eds.) Proc. of the 2nd Conf. on Parallel Problem Solving from Nature, pp. 107–116. Springer, Heidelberg (1992)

20. Samples, M.E., Byom, M.J., Daida, J.M.: Parameter Sweeps for Exploring Parameter Spaces of Genetic and Evolutionary Algorithms. In: [12], pp. 161–184
21. Schaffer, J.D., Caruana, R.A., Eshelman, L.J., Das, R.: A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In: Schaffer, J.D. (ed.) Proc. of the 3rd Int. Conf. on Genetic Algorithms, pp. 51–60. Morgan Kaufmann, San Francisco (1989)
22. Taguchi, G., Wu, Y.: Introduction to Off-Line Quality Control. Central Japan Quality Control Association, Nagoya, Japan (1980)