CHAPTER 2

RELEVANCE ESTIMATION AND VALUE CALIBRATION OF EVOLUTIONARY ALGORITHM PARAMETERS

Abstract

Evolutionary algorithms (EAs) form a rich class of stochastic search methods that use the Darwinian principles of variation and selection to incrementally improve a set of candidate solutions (Eiben and Smith, 2003; Jong, 2006). Both principles can be implemented from a wide variety of components and operators, many with parameters that need to be tuned if the EA is to perform as intended. Tuning however requires effort, both in terms of time and computing facilities.

When resources are limited we are interested to know how much tuning an EA requires to reach an intended performance, and which parameters are most relevant in the sense that tuning them has the biggest impact on EA performance. Likewise, when designing an EA to simulate a real evolutionary process we would like to minimize the dependency of our simulation results on specific parameter values. In this case the amount of tuning required until the simulation behaves as intended indicates how plausible and realistic the simulation really is.

To measure the amount of tuning that is needed in order to reach a given performance, we introduce the REVAC method for Relevance Estimation and Value Calibration. While tuning the EA parameters in an automated and systematic manner, the method provides an information-theoretic measure on how much the tuning of each parameter contributes to overall EA performance. We evaluate its reliability and efficiency empirically on a number of test cases that reflect the typical properties of EA parameter spaces, as well as on evolutionary agent-based simulations. Finally we compare it to another tuning method, meta-GA.

Parts of this chapter have been published in Nannen and Eiben (2007b,a); de Landgraaf et al. (2007).

2.1 Background

One of the big challenges in evolutionary computing is the design and control of evolutionary algorithm (EA) parameters (Eiben et al., 1999). Without exaggeration, one could state that one of the canonical design problems is how to chose the operators for an evolutionary algorithm to ensure good performance. For instance, the question whether crossover is a relevant operator is still open, or rather, the answer depends on the application at hand (Eiben and Smith, 2003). A related issue is the relevance of free EA parameters. Depending on the EA and the problem it is applied to, tournament size can be a highly relevant parameter whose value must be chosen well for good performance, while mutation rate could be less relevant in the sense that its values do not affect EA performance too much. When designing an evolutionary algorithm to model a real evolutionary system, for example in evolutionary economics, one often has to deal with non-standard evolutionary mechanisms. These can include domain specific features of which it is altogether unknown whether the system behavior depends on their correct parameterization.

While the tuning of relevant evolutionary algorithm (EA) parameters is essential to good EA performance, current practice in EA tuning is based on ill-justified conventions and ad hoc methods. In particular, studies on confidence intervals for good parameter values and sensitivity analyzes for parameter robustness are allmost non-existent. Part of the problem lies in the fact that most EAs are non-deterministic and path-dependent, in the sense that small changes to the initial conditions can lead to highly divergent results. This makes it difficult to obtain a reliable estimate of EA performance on a given problem. The standard statistical method to reduce variance and improve measurement reliability is measurement replication. With measurement replication, a set of parameter values is chosen, the EA is executed several times with these values on the same problem, and an aggregate performance measure is taken. A classical example of this approach is Analysis of Variance (ANOVA), which provides a clear set of rules how to optimally combine a number of carefully chosen parameter values, how to calculate the number of replications needed to decide whether one combination of values has a significantly better performance than another, and how to infer parameter interaction. An exhaustive overview of how to use ANOVA to tune an EA is given by Czarn et al. (2004).

This approach has a number of disadvantages, particularly when it is applied to an EA with several sensitive parameters. First, the choice of parameter values for the analysis is far from trivial and experiments in this vain often allow for no other conclusion than that a given choice was wrong. Second, the variance of an EA can easily be so high and its distribution so bad-behaved that the number of replications needed to produce significant results is not feasible. Third, there is disagreement in the statistical community on how to treat non-numerical results, for example when an EA does not find an acceptable solution within given computational constraints. Fourth, replications divert computational resources that could otherwise be used to obtain a better cover of the parameter space. This is a serious drawback, since it is virtually impossible to infer from a small number of measurements in a multi-dimensional search space, reliable as they might be, important measures of robustness like sensitivity to small changes and the range of values for which a certain EA performance can be achieved.

Here we propose to use an Estimation of Distribution Algorithm (EDA) to control

the parameters of an evolutionary algorithm: REVAC, which stands for Relevance Estimation and Value Calibration. REVAC is designed to a) tune or calibrate the parameters of an EA in a robust way and b) quantify the minimum amount of information that is needed to tune each parameter. Like a meta-GA (Grefenstette, 1986), it is an evolutionary method, a meta-EDA, that explores the parameter space of an evolutionary algorithm dynamically. Unlike meta-GA, it tunes an evolutionary algorithm on the basis of probability density functions over parameter values, rather than individual parameter values. Starting from a wide distribution over all possible parameter values, REVAC iteratively evaluates and improves the distribution such that it increases the probability of those parameter values that result in good EA performance. To avoid the pitfalls of bad-behaved distributions and non-numerical results, REVAC only uses rank based statistics to decide where to zoom in. Also, instead of investing valuable computational resources in measurement replications, REVAC uses them to get a better cover of the parameter space.

The estimated distributions over each parameter can be used to estimate the relevance of that parameter in an intuitive way. Broadly speaking, a distribution with a narrow peak indicates a highly relevant parameter whose values largely influence EA performance, while a broad plateau belongs to a less relevant parameter whose values do not matter too much. In terms of information theory, the Shannon entropy of a distribution expresses the average amount of information that is needed to specify a value that was drawn from the distribution Shannon (1948). The sharper the peaks of a continuous probability density function, the lower its Shannon entropy, and the less information is needed to specify the values drawn from the distribution. If a distribution over parameter values has maximum entropy for a given level of expected EA performance, then this maximum entropy can be used to calculate the minimum amount of information that is needed to achieve that performance. REVAC forces the distribution it finds to approximate the maximum entropy distribution for a given level of performance by continuously smoothing them between updates, so that their Shannon entropy can be used to estimate the minimum amount of information needed to reach this level of performance. In these terms the objectives of REVAC can be formulated as follows:

- The Shannon entropy of the distribution is as high as possible for a given level of performance,
- The expected performance of the EA in question is as high as possible for a given level of Shannon entropy.

Related work includes meta-GA as an early attempt to automate the tuning of genetic algorithms (Grefenstette, 1986), and Eiben et al. (1999) who established parameter control in EAs as one of the big challenges in evolutionary computing. Czarn et al. (2004) discuss current problems in EA design and use polynomial models of a performance curve to estimate confidence interval for parameter values. François and Lavergne (2001) estimate performance curves for EAs across multiple test cases to measure generalizability. Bartz-Beielstein et al. (2005) uses a Gaussian correlation function to dynamically build a polynomial regression model of the response curve.

The groundwork of statistical experimental design was laid by R. A. Fisher in the 1920s and 1930s. The use of sequential sampling to search for optimal parameter val-

ues were introduced by Box and Wilson (1951). A paradigm shift that emphasizes the robustness of a solution is due to Taguchi and Wu (1980). Estimation of Distribution Algorithms, in particular those based on univariate marginal distributions, to which the present type belongs, were pioneered by Mühlenbein (1997). The relationship between Shannon entropy and EDAs is discussed extensively in Mühlenbein and Höns (2005).

A detailed description of REVAC is given in Section 2.2. In Section 2.3 we use abstract objective functions and a simple genetic algorithm (GA) to verify that REVAC can indeed estimate how much tuning the parameters of an EA need. Section 2.4 uses the same simple GA to evaluate whether REVAC uses the available evaluations of candidate solutions efficiently, without the need for measurement replication. The same section also tests REVAC on an agent-based simulation from evolutionary economy. Section 2.5 compares REVAC to other tuning methods, namely hand tuning and meta-GA. A summary and conclusions can be found in Section 2.6.

2.2 The algorithm

2.2.1 Approaching the maximum entropy distribution

Formally, the tuning of EA parameters to a specific application is itself an optimization problem where the value of an objective function $r = f(\vec{x})$ is maximized. The domain of this objective function are the possible combinations of parameter values $\vec{x}$ for the EA. Its value r, which is also called the response, is the expected performance of the EA on the application problem when executed with these parameter values.

Since the domain of many EA parameters is continuous, the choice of suitable EDAs to tune them is limited. The present algorithm is a steady state variant of the Univariate Marginal Distribution Algorithm (Mühlenbein, 1997). For efficiency, only a single parameter vector is evaluated between every update of the distributions. Given an EA with k parameters, REVAC defines a joint distribution $\mathscr{D}(\vec{x})$ over the space of possible parameter vectors $\vec{x} = \{x_1, \ldots, x_k\}$. This joint distribution is composed from a set of independent marginal density functions $\mathscr{D}(\vec{x}) = \langle \mathscr{D}(x_1) \ldots \mathscr{D}(x_k) \rangle$ over each parameter. Their Shannon entropy can be used to estimate how much information is needed per parameter to achieve the expected performance of the joint distribution. Let a probability density function $\mathscr{D}$ be defined over a continuous interval $[a, b]$. Its differential Shannon entropy h can be calculated as

$$h(\mathscr{D}_{[a,b]}) = -\int_a^b \mathscr{D}(x) \log_2 \mathscr{D}(x)\, dx. \tag{2.1}$$

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 unit interval $[0,1]$ before calculating the Shannon entropy. In this way the uniform distribution has a Shannon entropy of zero, and any other distribution has a negative Shannon entropy $h(\mathscr{D}_{[0,1]}) < 0$.

Starting with the uniform distribution, REVAC iteratively refines the distribution by drawing random vectors of parameter values from it, measuring the performance of the EA with these parameter values, and increasing the probability of those regions of the

Table 2.1: Two views on a table of parameter vectors

	$\mathscr{D}(x_1)$	$\cdots$	$\mathscr{D}(x_i)$	$\cdots$	$\mathscr{D}(x_k)$
$\vec{x}^1$	$\{x_1^1$	$\cdots$	x_i^1	$\cdots$	$x_k^1\}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$		$\vdots$
$\vec{x}^j$	$\{x_1^j$	$\cdots$	x_i^j	$\cdots$	$x_k^j\}$
$\vdots$	$\vdots$		$\vdots$	$\ddots$	$\vdots$
$\vec{x}^n$	$\{x_1^n$	$\cdots$	x_i^n	$\cdots$	$x_k^n\}$

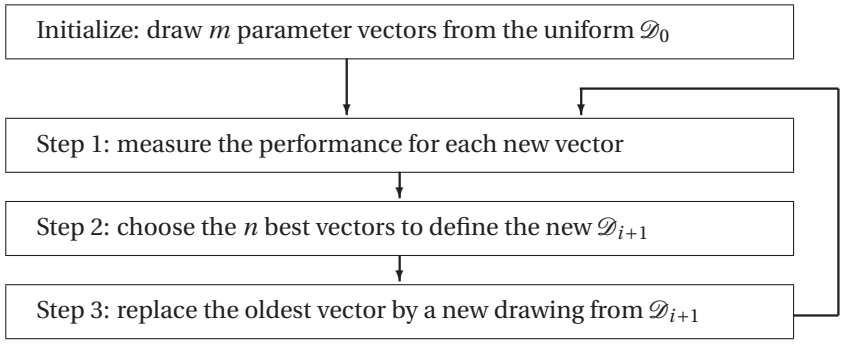
parameter space where a higher EA performance is measured. In this way, new distributions are built on estimates of the response surface that were sampled with previous distributions, each iteration increasing the expected performance of parameter vectors that are drawn from the distribution. To reduce the variance of stochastic measurements and to prevent premature convergence, REVAC continuously smoothes the distribution. It is the unique combination of these two operators, increasing the probability of regions with high performance and smoothing out the resulting probability function, that allows REVAC to approach the maximum entropy distribution for a given level of EA performance.

2.2.2 Algorithm implementation

At each step in the tuning process, REVAC maintains a pool of m vectors of parameter values. From this pool the $n < m$ vectors with the highest measured performance are selected to define the current distribution and to create a single new parameter vector. The new parameter vector always replaces the oldest one in the pool. For a good understanding of how this is done it is helpful to distinguish two views on the n selected parameter vectors as shown in Table 2.1. Taking a *horizontal* view on the table, a row is a vector of parameter values and we can see the table as n of such vectors. Taking a *vertical* view on the table, the i^{th} column shows n values from the domain of parameter i. Each column of Table 2.1 defines a marginal density function and the whole table defines the joint density function.

As can be seen in the diagram of Figure 2.1, REVAC initializes the table of parameter vectors by drawing k vectors from the uniform distribution over the space of possible parameter values. The update process that creates a new table of parameter vectors consists of three basic steps: *evaluating parameter vectors:* Given a vector of parameter values, we can evaluate it by executing the EA with these parameter values and measuring its performance; *updating the probabilities:* Given a set of evaluated parameter vectors, we can calculate the probability that some regions of the parameter space have a higher expected performance than others; *generating parameter vectors:* Given a probability density function over the parameter space, we can draw new parameter vectors proportional to those probabilities.

Figure 2.1: Diagram of the update process



Step one is straightforward. As for step two and three, they can be described from both the horizontal and the vertical perspective of Table 2.1. Looking from the *horizontal* perspective, REVAC can be described as a population based evolutionary algorithm with operators for selection, recombination and mutation. This description of REVAC must not be confused with the EA we are tuning. The population consists of m parameter vectors. It is updated by selecting $n < m$ parent vectors from the old population, which are then recombined and mutated to obtain exactly one child vector every generation. The child vector always replaces the oldest vector in the population.

REVAC uses a deterministic choice for parent selection as well as for survivor selection. The n vectors of the population that have the highest measured performance are selected to become the parents of the new child vector. Recombination is performed by a multi-parent crossover operator, uniform scanning, that creates one child from n parents, cf. Eiben and Smith (2003). The mutation operator—applied to the offspring created by recombination—is rather complicated. It works independently on each parameter i in two steps. First, a mutation interval $[x_a^i, x_b^i]$ is calculated, then a random value is chosen uniformly from this interval. To define the mutation interval for mutating a given x_i^j all other values $x_i^1, \ldots, x_i^n$ for this parameter in the selected parents are also taken into account. After sorting them in increasing order, the begin point of the mutation interval or window can be specified as the w-th lower neighbor of x_i^j, while the end point of the interval is the w-th upper neighbor of x_i^j. The new value is drawn from this interval with a uniform distribution. As there are no neighbors beyond the upper and lower limits of the domain, we extend it by mirroring the parent values as well as the mutated values at the limits, similar to what is done in Fourier transformations.

From the *vertical* perspective we consider step two and three as constructing k marginal probability density functions from the columns of Table 2.1 and then drawing a new parameter vector from these distributions. To define a marginal density function $\mathscr{D}(x_i)$, the n values of column i are sorted and arranged such that together with the limits 0 and 1 (the domain of each parameter is scaled to the unit interval) they form $n+1$ non-overlapping intervals that cover the entire domain. The density over any such

interval $[x_i^a, x_i^b]$ can be defined as

$$\mathscr{D}(x_i) = \frac{1}{(m+1)(x_i^b - x_i^a)}, \tag{2.2}$$

which satisfies $\int_0^1 \mathscr{D}(x_i)\, dx_i = 1$. This definition of a density function can be extended to allow intervals to overlap, for example by defining intervals between values that are separated by one or two other values. To overcome the problem of missing neighbors at the limits we again mirror all defining parameter values as well as the chosen values at the limits. The further the values that define an interval are separated, the higher the Shannon entropy of the resulting distribution.

In this context, the rationale behind the complicated mutation operator of the horizontal view is that it heavily smoothes the density functions of equation 2.2. Like all evolutionary algorithms, an EDA is susceptible for converging on a local maximum. By continuously smoothing the probability density functions we force them to converge on a maximum of the response surface that lies on a broad hill, yielding robust solutions with broad confidence intervals. But smoothing does more: it allows REVAC to operate under very noisy conditions, it allows it to readjust and relax marginal distributions when parameters are interacting and the response surface has curved ridges, and it maximizes the entropy of the constructed distribution. Smoothing is achieved by taking not the nearest neighbor but the w-th neighbors of x_i^j when defining the mutation interval. Choosing a good value for w is an important aspect when using REVAC. A large w value can slow down convergence to the point of stagnation. A small w value can produce unreliable results. Based on our experience so far, we prefer $w \approx n/10$.

2.2.3 Interpreting the measurements

REVAC, like any EDA, is a random process. The final result is different with every run or tuning session of REVAC. Independently of whether REVAC uses measurement replication, REVAC results can be made more reliable by tuning an EA more than once, and by either choosing the tuned parameter values that resulted in the highest EA performance, or by averaging over the tuned parameter values of several runs of REVAC, as will be explained below. This is indeed a replication of measurements at a higher level. But unlike ordinary measurement replication, which can be too expensive to extract any useful information, REVAC can always provide a first approximation, which can then be refined by repeating the tuning process.

Because REVAC produces a sequence of distributions with slowly decreasing Shannon entropy we use the Shannon entropy of these distributions to estimate the minimum amount of information needed to reach a target performance level. This can be used in several ways. First, it can be used to choose between different sets of EA operators. A set of operators that needs less information to reach a given level of EA performance is easier to tune, more fault tolerant in the implementation, and robuster against changes to the problem definition. Second, it can be used to identify the critical components of an EA. A highly sensitive parameter typically has a sharp peak in the distribution and a low Shannon entropy. When an EA needs to be adjusted to a new problem, sensitive parameters need the most attention, and with this knowledge the practitioner can

concentrate on the critical components straight away. Third, it can be used to suggest values and confidence intervals for the best parameter values. Given a distribution that peaks out in a region of high probability (except for the early stage of the algorithms the marginal distributions have only one peak), we take the 50^{th} percentile (the median) to be the best tuned parameter values, and use the 25^{th} and the 75^{th} percentile of the distribution as confidence interval. That is, every value from this range leads to a high expected performance, under the condition that the other parameters are also chosen from their respective confidence interval. This confidence interval is also useful when we average over several runs of REVAC. As we only want to average over those runs that converged on the same optimum in the parameter space, we take the average of only those REVAC distributions where all medians lie within the 25^{th} and 75^{th} percentiles of the respective distributions of the REVAC run that achieved the best EA performance. REVAC runs that converged on values beyond these intervals are discarded.

Throughout the rest of this doctoral thesis REVAC will use a population of $m = 100$ parameter vectors, from which the best $n = 50$ are selected for being a parent. We smooth by extending the mutation interval over the $w = 5$ upper and lower neighbors. In each run or tuning session REVAC is allowed to evaluate 1,000 parameter vectors.

2.3 Assessing the reliability of REVAC estimates

A real in vivo assessment of REVAC requires that we tune an EA on a set of application problems where the objective function is known, and use this to evaluate the results obtained by REVAC. It is however not feasible to accurately model the response surface of an EA on any non-trivial application problem. According to Czarn et al. (2004), even when working with rather simple application problems and a simple genetic algorithm with only two free parameters, it is difficult to fit anything more sophisticated than a cubic curve to the measured performance. This leaves us with two alternatives: we evaluate REVAC on abstract objective functions with a predefined response surface that is representative for EA tuning problems. This has the added advantage that by abstracting both the application and the algorithm layer the run time of the tuning process is reduced enormously, and that the assessment of REVAC can be based on a large number of measurements. The second alternative is to use an EA and an application problem that have been studied in the literature and to evaluate the REVAC relevance estimates against existing results on the relevance of the EA parameters.

2.3.1 Assessing REVAC reliability on abstract objective functions

In general, we distinguish 3 layers in designing an EA, as shown in Table 1.1 on page 19. For the present assessment, these layers are implemented as follows:

Experimental setup of Section 2.3.1	
design tool	*REVAC*
evolutionary algorithm	*abstract*
application problem	*abstract*

To define abstract response surfaces that resemble the response surface of a typical EA we identify five essential properties: *1) Low dimensionality:* typically not more than ten parameters need to be tuned. *2) Non-linearity (epistasis, cross-coupling):* parameters interact in non-linear ways, which implies that the response curves are non-separable, and that values for one parameter depend on the values of other parameters. *3) Smoothness:* small changes in parameter values lead to small changes in EA performance. *4) Low multi-modality (moderate ruggedness):* the objective function has only one or few significant local optima, i.e., few regions of the parameter space have high EA performance. *5) Noise:* depending on the application, the performance of an EA can be highly variable and can follow a distribution that is significantly different from the normal distribution.

We present three experiments: two on the accuracy of the relevance estimates (without noise) and one on the resistance of the estimates to noise. In all experiments the abstract objective function simulates the performance of an EA with $k = 10$ parameters, each of which can take values from the range $[0, 1]$.

Experiment 1: hierarchical dependencies. In our first experiment we hardcode predefined dependencies between parameters, such that the optimal value of parameter i depends on the current value of $i-1$ and the utility of tuning parameter i depends on how well parameter $i-1$ is tuned. The abstract objective function $r = f(\vec{x})$, is defined as the sum

$$r = \sum_{i=1}^{10} r_i \tag{2.3}$$

over ten partial response values $r_1, \ldots, r_{10}$, one for each of the ten parameters. These are calculated as follows. Before a tuning session is started, a single target value t is chosen at random from the range $[0, 1]$ and kept constant throughout the tuning session. When evaluating a parameter vector $\vec{x} = \{x_1, \ldots, x_k\}$, the partial response r_1 of the first parameter value x_1 is one minus the distance to the target value,

$$r_1 = 1 - |x_1 - t|. \tag{2.4}$$

The partial response r_i of each consecutive parameter value depends on how close the value x_i is to that of parameter $i-1$,

$$r_i = r_{i-1}(1 - |x_i - x_{i-1}|). \tag{2.5}$$

We have $x_1 - t \leq 1$ and $x_i - x_{i-1} \leq 1$ for any i. Because r_{i-1} is a factor in the calculation of r_i, the inequality $r_i > r_{i+1}$ always holds, with the effect that the first parameter needs more tuning than the second one, and so forth.

Figures 2.2 and 2.3 shows typical results when using REVAC to tune the 10 parameters to this abstract objective function. Results are from a single run of REVAC. The bar diagram in Figure 2.2 shows the final Shannon entropy per parameter after evaluating 1,000 parameter vectors. The precoded liner relationship in the need for tuning is well approximated. In particular, the order of the first five parameters, which need the most tuning, is correctly identified. The upper left graph of figure 2.3 shows how the measured Shannon entropy of the REVAC distributions over parameter 1, 4, and 7 changes during the tuning session. The other three graphs of the figure show how the 25^{th}, 50^{th}

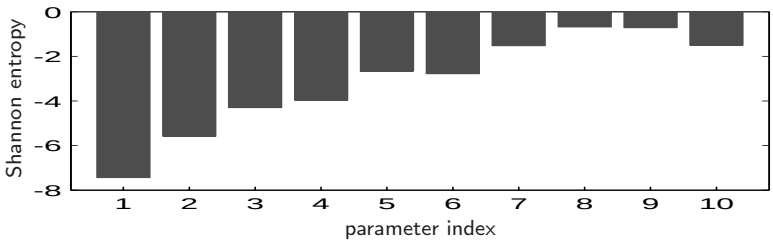


Figure 2.2: Final Shannon entropy per parameter

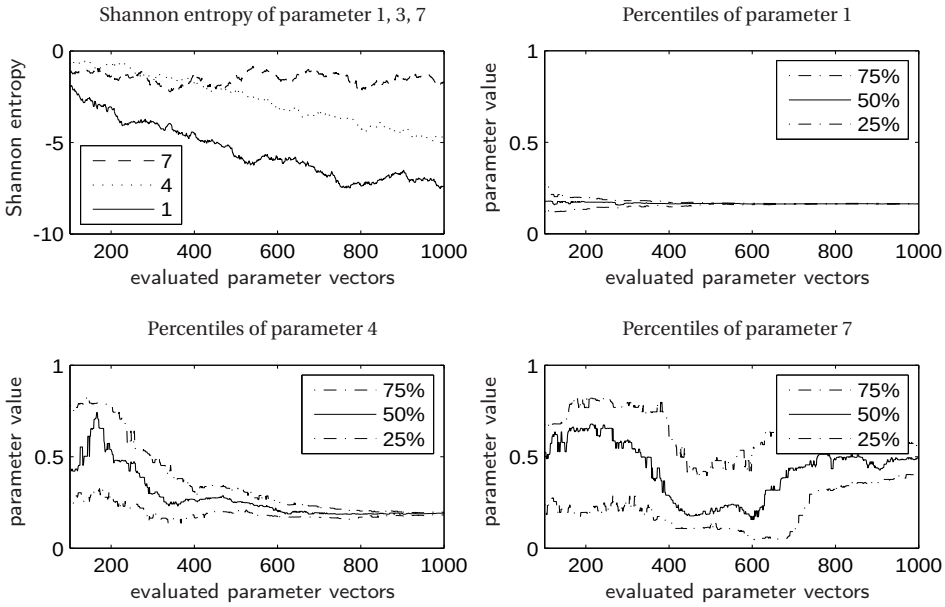


Figure 2.3: Shannon entropy and the percentiles of the REVAC distributions of parameter 1, 3, and 7 during tuning.

and 75^{th} percentile of the same distributions change during tuning. Note that the distance between the 25^{th} and 75^{th} percentile of the distributions behaves similar to the entropy of the same distribution.

Experiment 2: predefined relevance distribution. In this experiment we are interested to see whether REVAC can reveal arbitrary distributions over parameter relevance. To this end we create an abstract objective function with one peak which is placed at random in the 10-dimensional unit space. Again, total response is the sum of the partial response per parameter. This is calculated as one minus the distance of a parameter value x_i to the corresponding peak value t_i, weighted by a predefined vector of target

weights $w = \langle w_1, \ldots, w_{10} \rangle$,

$$r = \sum_{i=1}^{10} w_i \left[1 - (x_i - t_i)\right]. \tag{2.6}$$

In this way, tuning a parameter with a target weight close to zero has no impact on overall response, while tuning a parameter with a large target weight has a significant impact on the response. We use three sets of target weights to specify three elementary distributions over parameter relevance. The first distribution has two outliers of exceptionally low relevance, while the remaining target weights are equal (normalize w, with $w_1 = 0$, $w_2 = 1$, and $w_3, \ldots, w_{10} = 10$). The weights of the second distribution increase linearly over the parameters (normalize w, with $w_i = i$). The weights of the third distribution increase exponentially such that there are outliers with a high relevance (normalize w, with $w_i = i^{10}$). This last distribution represents what is known as *sparcity of effects* in the design of experiments and is the most typical situation when tuning a real EA.

Figure 2.4 shows the target weights (the black columns) together with what REVAC has estimated after 1,000 evaluated parameter vectors (the white columns). Results are from a single run of REVAC. To estimate how relevance is distributed over the parameters we normalize the Shannon entropy of the marginal REVAC distributions, which results in positive values that sum to one. As can be seen, REVAC approximately reproduces the hardcoded order of relevance, in particular with regard to the outliers, but has difficulties when the weights are too similar. When averaging over several REVAC runs (not shown here), the order of the estimated relevance per parameter converges to the order of the predefined target weights.

Experiment 3: measurement noise. In this experiment we study how the reliability of a relevance estimate changes with additive noise of increasing variance. For the abstract objective function we add a noise term η to the objective function 2.6 of experiment 2,

$$r = \sum_{i=1}^{10} w_i \left[1 - (x_i - t_i)\right] + \eta. \tag{2.7}$$

To simulate sparcity of effects, the weights $w_1, \ldots, w_{10}$ increase exponentially from parameter to parameter, cf. the bottom graph of Figure 2.4. Values for the noise term η are independent and identically distributed. They are drawn from a Pareto distribution

$$P(X > x) = cx^{-\gamma}, \qquad x > \log_\gamma c \tag{2.8}$$

with exponent $\gamma = 2.3$. The value c controls the variance σ^2 of the noise. Such a distribution is also called a power law distribution and can be found in many physical and social systems. It has frequent outliers, its variance converges rather slowly, and it is generally incompatible with statistical methods that require a normal distribution.

To measure the error of the REVAC estimates we use the mean squared distance between target weights and the corresponding normalized Shannon entropy after evaluating 1,000 parameter vectors. With $s_1, \ldots, s_{10}$ the normalized Shannon entropy, the error can be calculated as

$$error = \frac{1}{10} \sum_{i=1}^{10} (s_i - w_i)^2. \tag{2.9}$$

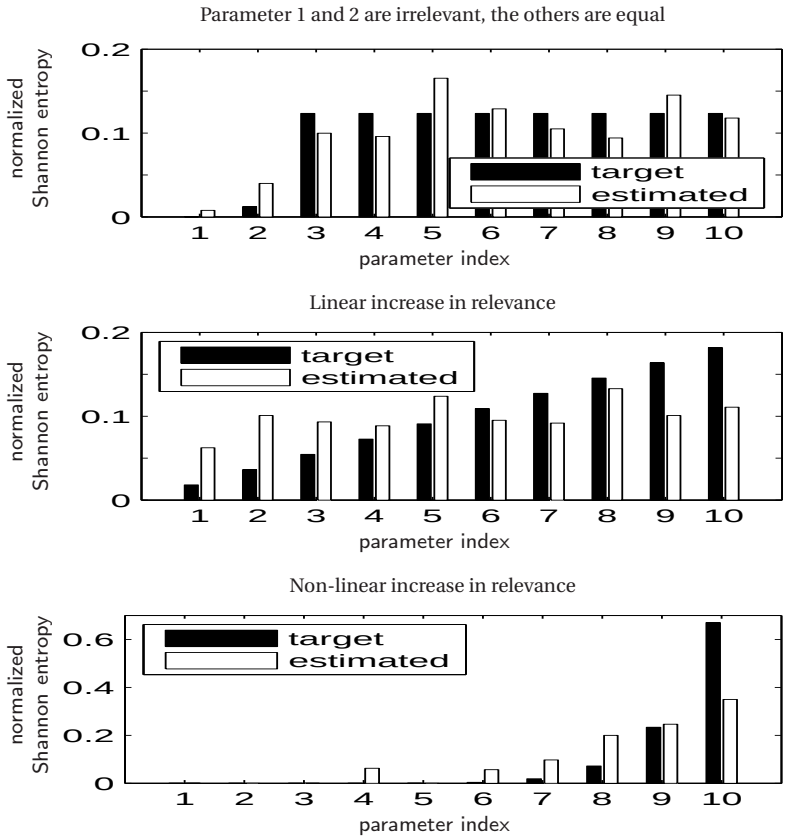


Figure 2.4: Comparing the normalized Shannon entropy per parameter as estimated by REVAC to the target weights of the abstract objective function.

Figure 2.5 plots the measured error for a single run of REVAC against the variance σ^2 of the noise. The mean squared error of the REVAC estimate increases roughly linearly with the variance. Note that the highest value for σ^2 is five, while the objective function itself only takes values from the range $[0,1]$. The mean squared error hardly exceeds the value 0.1. To compare, the mean squared error between the target weights and a 10-dimensional normalized random vector is 0.29.

The variance of independent and identically distributed noise can be reduced by measurement replication. As seen in Figure 2.6, REVAC estimates can also be improved by taking the average of several REVAC runs that were obtained without measurement replication. Both graphs compare the estimated normalized Shannon entropy after 1,000 evaluated parameter vectors (white columns) to the target weights (black columns). The variance of the noise is $\sigma^2 = 5$. The upper graph is based on a single run of REVAC and the lower graph on 10 runs. The mean squared error of the relevance estimate is 0.022 in the upper graph and 0.011 in the lower graph. This means that under noisy conditions a single run of REVAC without measurement replication can give a quick first approximation that can be refined by further runs if resources permit.

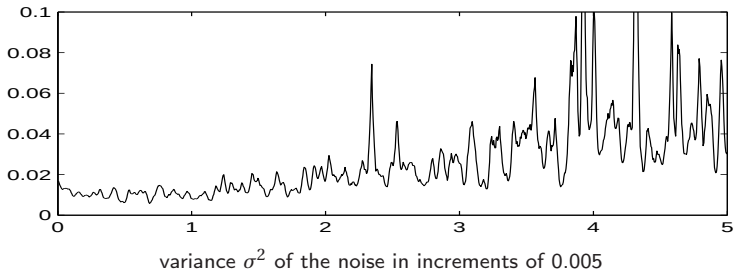


Figure 2.5: Impact of noise on the mean squared error of the normalized Shannon entropy after 1,000 evaluated parameter vectors. The graph is smoothed for readability.

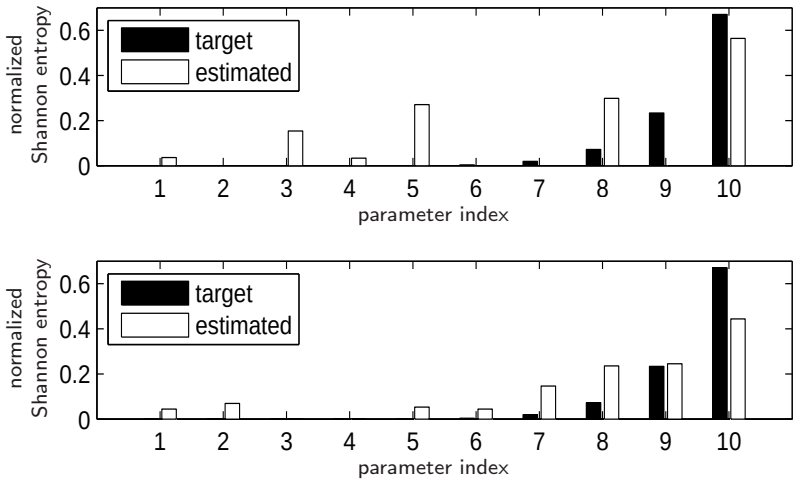


Figure 2.6: Relevance estimates with noise of variance $\sigma^2 = 5$. The upper graph shows a typical estimate from a single run. The lower graph shows the average of 10 typical runs.

2.3.2 Assessing REVAC reliability on a simple genetic algorithm

Experimental setup of Section 2.3.2

design tool	*REVAC*
evolutionary algorithm	*simple genetic algorithm*
application problem	*standard numerical optimization problems*

Here we present the results of tuning an EA on application problems that have been previously studied in the literature, as discussed at the beginning of this Section 2.3. For both the EA and the objective function we rely on Czarn et al. (2004), who use rigorous statistical exploratory analysis to tune a simple genetic algorithm and who compare their results to those of Jong (1975), Schaffer et al. (1989), Grefenstette (1986), and

Table 2.2: REVAC results after 1,000 evaluations

Function & parameters		Optimum value	Confidence interval (25^{th} and 75^{th} pctl.)	Shannon entropy	Normalized Shannon entropy
f_1	p_m	0.012	0.011 – 0.013	-8.6	0.82
	p_c	0.90	0.77 – 0.96	-1.9	0.18
f_2	p_m	0.0146	0.0143 – 0.0148	-9.4	0.82
	p_c	0.82	0.77 – 0.86	-2.1	0.18
f_3	p_m	0.0338	0.0334 – 0.0342	-9.0	0.72
	p_c	0.98	0.82 – 0.99	-3.5	0.28
f_6	p_m	0.0604	0.0635 – 0.0641	-6.9	0.86
	p_c	0.60	0.48 – 0.68	-1.1	0.14

Freisleben and Hartfelder (1993). Specifically, they study the effect of tuning the mutation parameter $p_m \in [0,1]$ and the crossover parameter $p_c \in [0,1]$ of a generational genetic algorithm (GA) with 22 bits per variable, Gray coding, probabilistic ranked-based selection, bit flip mutation, single point crossover, and a population of 50 chromosomes. The 4 objective functions for the application layer are standard benchmark problems from Jong (1975) and Schaffer et al. (1989): sphere (f_1), saddle (f_2), step (f_3), Schaffer's f_6. Their definitions are given in equation 2.10—2.13,

$$f_1(x) = \sum_{i=1}^{3} x_i^2, \qquad -5.12 \le x_i \le 5.12, \tag{2.10}$$

$$f_2(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \qquad -2.048 \le x_i \le 2.048, \tag{2.11}$$

$$f_3(x) = \sum_{i=1}^{5} \lfloor x_i \rfloor, \qquad -5.12 \le x_i \le 5.12, \tag{2.12}$$

$$f_6(x) = 0.5 + \frac{\left(\sin\sqrt{x_1^2+x_2^2}\right)^2 - 0.5}{\left(1+0.0001(x_1^2+x_2^2)\right)^2}, \qquad -100 \le x_i \le 100. \tag{2.13}$$

Table 2.2 shows the results per objective function after evaluating 1,000 parameter vectors. Figure 2.7 shows the Shannon entropy during tuning and the final distributions after tuning for Shaffer's f_6. The upper graph shows the Shannon entropy of p_m and p_c during tuning. Only the entropy of p_m decreases significantly, indicating its relevance. The two lower graphs show the final probability density function over the parameter values after evaluating 1,000 parameter vectors. Note the extremely sharp needle for p_m.

These results can be considered from two perspectives, compared with the "usual" GA settings, and with the work of Czarn *et al.* As Table 2.2 shows, the values found by REVAC are consistent with the conventions in evolutionary computing: p_m between 0.01 and 0.1, and p_c between 0.6 and 1.0. On the other hand, a direct comparison with Czarn et al. (2004) is difficult because of the different types of outcomes. As for the method, Czarn et al. use screening experiments to narrow down the space of feasible parameter settings, partition this space into equally spaced discrete levels, repeatedly measure

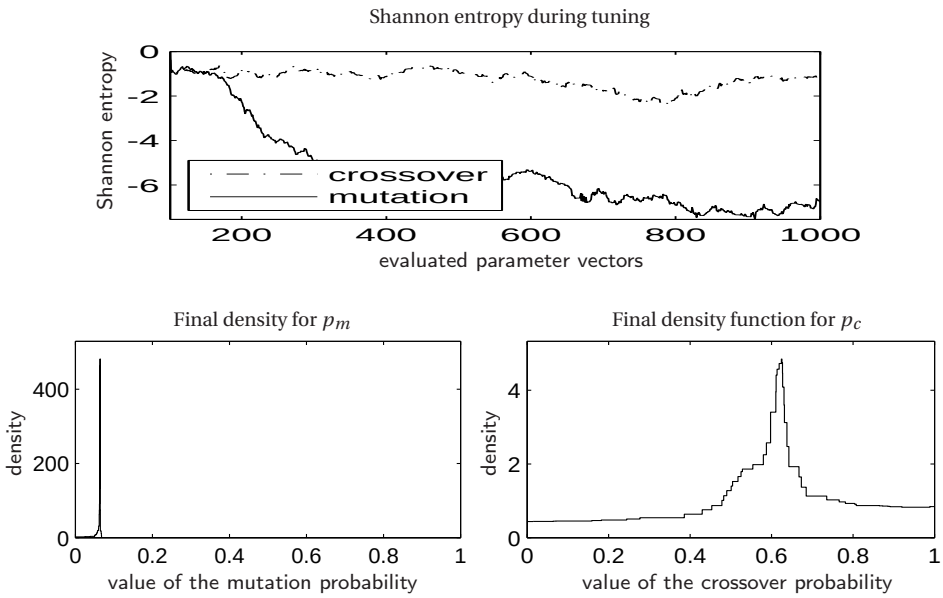


Figure 2.7: Tuning Schaffer's f_6

the performance for each level and use ANOVA to calculate the significance of mutation and crossover rates. Then they procede to approximate the response curve for both parameters by fitting low order polynomials to the performance measures, suggesting to calculate confidence intervals from these approximations. As a main result, Czarn et al. find that the marginal response curves for crossover and mutation are linear and quadratic and that mutation is more significant than crossover. By contrast, REVAC uses no screening and does not partition the parameter space into discrete levels. It studies the complete and continuous parameter space. It can narrow the solution space to any arbitrarily small subspace and can directly read off confidence intervals for any given level of performance. Our global outcomes, however, are in line with those in Czarn et al. (2004): p_m is much more peaked and relevant than p_c.

2.4 Assessing the algorithmic efficiency of REVAC

To reduce the variance in the measured performance of an EA, statistical methods commonly rely on measurement replication. Until some confidence is achieved as to which vectors lead to a higher EA performance, these methods invest valuable computational resources in evaluating the same vectors of parameter values over and over. By contrast, REVAC reduces the variance in the measured performance implicitly through extensive sampling and smoothing. An increase of the pool size m and the number of selected vectors n means that estimated densities are based on a larger and more reliable number of evaluated parameter vectors. An increase of the smoothing parameter w means that the densities are averaged over a larger number of adjacent parameter intervals.

By using sufficiently large numbers for m, n, and w, REVAC aims to both correct for the variance in the measured performance as well as to get a better cover of the parameter space. Here we address the question whether this is indeed achieved, or, conversely, whether measurement replication will improve the quality of REVAC estimates or reduce the computational cost of obtaining them. This question is particularly pressing since REVAC is intended to tune EAs under conditions where established methods like ANOVA are inefficient, and where a maximum of information has to be extracted from every available measurement. We formulate two research questions: First, how does the replication of measurements affect the quality of REVAC estimates? And second, how does the replication of measurements affect the computational efficiency of the REVAC search process?

In order to study the merits of measurement replication for REVAC, a new parameter r for the number of measurement replications is added to step 1 of the REVAC algorithm, cf. Figure 2.1. Upon drawing a new parameter vector $\vec{x}$ from the joint distribution $\mathscr{D}(x)$, it is evaluated r times, and the average result is recorded. We use our standard REVAC implementation with $m = 100$, $n = 50$, $w = 5$. Two sets of experiments are reported: tuning a simple genetic algorithm (GA) on standard numerical optimization problems and the tuning evolutionary mechanism of a complex simulation as part of our research on evolutionary agent-based economics.

2.4.1 Assessing algorithmic efficiency on a simple genetic algorithm

Like in section 2.3.2 we rely on Czarn et al. (2004) for the GA and the objective functions sphere (f_1), saddle (f_2), step (f_3), and Schaffer's f_6 of equations 2.10—2.13. In addition to the two parameters tuned there, mutation $p_m \in [0,1]$ and crossover $p_c \in [0,1]$, we also tune the population size of $n \in [10,200]$ chromosomes, a total of 3 parameters. Figure 2.8 demonstrates how the Shannon entropy and the percentiles of the marginal distributions change during a typical tuning session without measurement replication. Here the step function (f_3) is used. The upper left graph shows the Shannon entropy of all three GA parameters. The other three graphs show the median and the 25^{th} and 75^{th} percentiles per parameter.

Experimental setup of Section 2.4.1

design tool	*REVAC*
evolutionary algorithm	*simple genetic algorithm*
application problem	*standard numerical optimization problems*

The performance measure of the GA that we wish to optimize is the computational cost of maximizing the objective function to which it is applied. This computational cost is the number of fitness evaluations, which in this case is calculated as the population size of the GA times the number of generations that are needed to maximize the objective function. The performance of the GA is maximized when the cost is minimized. When a GA needs 100 generations of 100 individuals or 200 generations of 50 individuals, we will say that it has a cost of 10,000 fitness evaluations. An objective function is considered maximized as soon as one individual of the population encodes a value that is within certain bounds of the best feasible solution. These bounds are chosen such

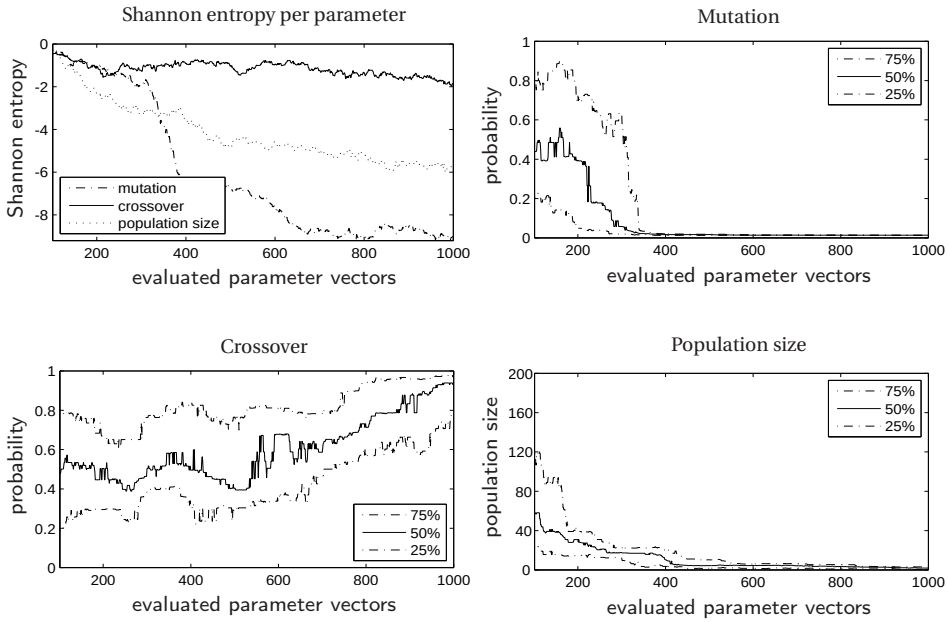


Figure 2.8: Shannon entropy and percentiles of the marginal distributions over 3 GA parameters during a typical tuning session.

that a well tuned algorithm can solve each objective function with a cost of between 5,000 and 10,000 fitness evaluations. If the algorithm does not solve the objective function with a cost of less than 25,000 fitness evaluations (e.g., in 1,000 generations if the population size is 25), execution is aborted and a cost of 25,000 fitness evaluations is recorded. We assess 5 different levels of measurement replication, $r \in \{1,2,3,5,10\}$. For each level of r and each objective function we run REVAC ten times, each run constituting and independent tuning session of the GA, and we report the average of the ten estimates.

In order to assess the quality of REVAC relevance estimates for each level r we need a reliable target value to compare to. Section 2.3.1 has shown that the average of repeated REVAC estimates without replication converges on the predefined distribution of parameter relevance. Section 2.3.1 has shown that REVAC can tune the simple GA to the four numerical optimization problems and give reasonable relevance estimates. We assume that the average relevance estimate of multiple REVAC runs converges on the correct values and we use these convergent values as the target values. For this reason our target value for each parameter on each objective function is the average Shannon entropy at the end of all REVAC runs with all levels of replications, 50 runs for each objective function. The exact values can be seen in Table 2.3. To assess the quality of the estimates obtained with a given number of measurement replications we use the error or mean squared distance to this target value, cf. equation 2.9. We consider the following four quantities:

- the number of different parameter vectors that REVAC needs to evaluate in order to reach an error < 0.1 with regard to the target values,
- the total number of measurements that are needed to reach an error < 0.1 (i.e., the number of parameter vectors times the number of measurement replications),
- the error after evaluating 1,000 vectors, regardless of the total number of measurements, and
- the error after a total number of 1,000 measurements.

Table 2.4 shows the recorded values for each level of measurement replication. Results are averaged over all objective function. The table clearly shows that a higher number of replications comes with a heavy computational penalty, without leading to a significant improvement in the quality of the relevance estimates. To be precise, there is no observable trend in the error after 1,000 evaluated parameter vectors for $r > 1$, while at this point it is still significantly higher for $r = 1$. As Figure 2.9 indicates, this is can be due to the fact that with fewer overall measurements, after 1,000 evaluated parameter vectors REVAC with $r = 1$ is still converging on the final value. The graph plots the error against the number of evaluated parameter vectors for $r \in \{1, 2, 10\}$, and indeed, only after evaluating about 800 parameter vectors does REVAC with $r = 2$ and $r = 10$—which makes 1,600 and 10,000 measurements—reach an error that is visibly lower than the final error of REVAC $r = 1$. We conclude that while the evidence regarding $r = 1$ and $r = 2$ is inconclusive, there is no evidence that a number of replication of measurements greater than 2 leads to a significant improvement of the estimate.

Table 2.3: Average Shannon entropy of the 3 free GA parameters

	Sphere (f_1)	Saddle (f_2)	Step (f_3)	Schaffer's f_6
Mutation	-11.1	-11.3	-10.9	-9.6
Crossover	-1.7	-3.5	-2.2	-0.9
Population size	-5.9	-4.5	-6.2	-1.0

Table 2.4: Quality of the relevance estimate for different numbers of measurement replication. Results are averaged over all objective function.

Number of measurement replications	Number of vectors until error < 0.1	Number of measurements until error < 0.1	Error at 1,000 vectors	Error at 1,000 measurements
1	404	404	0.08	0.09
2	413	826	0.04	0.07
3	741	2,223	0.05	0.23
5	844	4,220	0.04	0.35
10	236	2,360	0.06	0.37

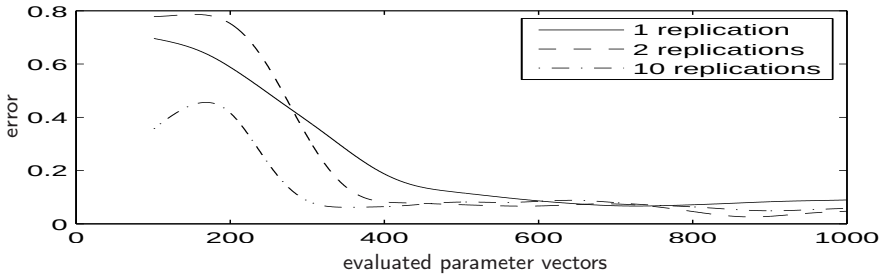


Figure 2.9: Error of relevance estimate for three levels of measurement replication, averaged over all 4 objective function. The lines are smoothed for readability. Note that the x-axis shows the number of evaluated parameter vectors, not the computational cost, which is measured in the total number of measurements.

Table 2.5: Best performance for each objective function, measured as number of fitness evaluations to solution. A lower value is better.

	Sphere	Saddle	Step	Schaffer's f_6
Optimum performance	3,786	2,770	2,107	3,260

To compare the quality of parameter values that REVAC has tuned we need an indication of how well a simple GA can perform on each objective function if properly tuned, i.e., the minimum amount of fitness evaluations that is needed to maximize the objective function. For this purpose we choose from among the 50 REVAC runs per objective function the parameter vector that achieved the best GA performance, and record the average number of fitness evaluations that the GA with this parameter vector needs to maximize the respective objective function. These best performances are shown in Table 2.5. We again consider four quantities:

- the number of parameter vectors that REVAC needs to evaluate in order to bring the computational cost of the GA down to no more than twice the computational cost of the best performance (i.e., 10,000 fitness evaluations if the best GA performance is 5,000 fitness evaluations),
- the number of measurements REVAC needs to perform in order to achieve the same as above,
- the average GA performance after REVAC has evaluating 1,000 parameter vectors, regardless of the number of measurements involved, and
- the average GA performance after REVAC has performed 1,000 measurements.

Table 2.6 and Figure 2.10 show the results. As Figure 2.10 reveals, the performance of the tuned GA on these problems is rather independent from the number of measurement replications employed by REVAC and depends primarily on the number of

Table 2.6: Quality of the tuned parameter values for different levels of measurement replication. Performance is measured in number of fitness evaluations. Results are averaged over all objective functions.

Number of measurement replications	Number of vectors until cost < 2∗best	Number of measurements until cost < 2∗best	Cost at 1,000 vectors	Cost at 1,000 measurements
1	411	411	9,954	9,789
2	397	795	6,326	7,250
3	241	722	4,783	4,877
5	380	1,901	10,576	10,424
10	277	2,772	9,006	9,072

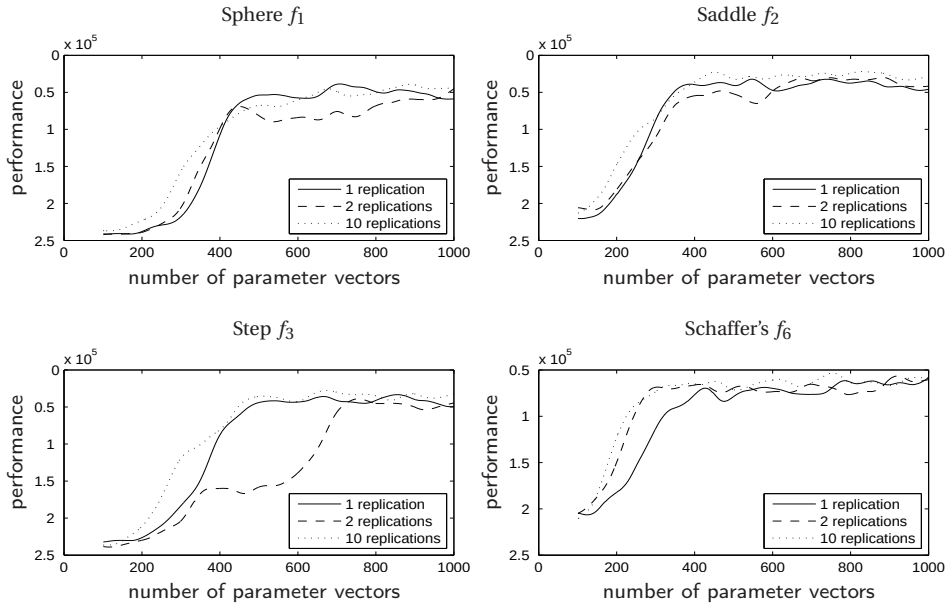


Figure 2.10: EA performance during tuning for three different levels of replications. Performance is calculated as the average number of fitness evaluations of the GA when parameter vectors are drawn from the respective REVAC distributions. The x-axis is counting from top to bottom, so that a higher performance is indeed on top. Graphs are smoothed for readability.

parameter vectors that REVAC has evaluated so far. Note in particular how performance is maximized around parameter vector 400 for all numbers of measurement replication. While measurement replication does not improve the absolute capability of REVAC to tune the parameter values, the performance penalty is huge. The amount of computation needed to reach an arbitrary level of performance increases almost linearly with the level of replication

2.4.2 Assessing algorithmic efficiency on an economic modeling problem

Here we test REVAC as part of our research in evolutionary agent-based economics. The use of evolutionary algorithm to model an economic system comes with a number of unique requirements that warrant a separate verification of whether REVAC can be applied to such modeling problems. These include, but are not limit to, non-standard evolutionary mechanisms, non-linear autocatalytic dynamics, and the need to create stable and realistic system behavior on the population level rather than to find a single optimal solution. Also, economic modeling can force an evolutionary algorithm to include domain specific features of which it is altogether unknown whether the system behavior depends on their correct parameterization.

Experimental setup of Section 2.4.2

design tool	*REVAC*
evolutionary algorithm	*selective imitation in a social network*
application problem	*dynamic growth and production functions*

To describe the experimental setup in a nutshell: 200 agents evolve their investment strategies over a period of 500 time intervals. In each interval each agent invests its current income in a number of economic sectors. The agent's income of the next interval is then calculated according to some production function. The production function changes dynamically, so that the same investment strategy will lead to different growth rates at different points in time. Agents adapt their investment strategies through random mutation and selective imitation in a complex social network. Mutation here is a random change to the way the investment is distributed over the economic sectors. For imitation an agent compares its own economic growth rate to that of its peers in the social network. If a peer has a higher growth rate than that of the comparing agent, the comparing agent can copy the strategy of that peer, wholly or in part, akin to crossover. The evolutionary mechanism will be discussed in detail in Section 4, while Section 5 will elaborate on the growth and production functions that are used here, as well as on the dynamic changes that the agents have to adapt to.

The performance measure of the EA that REVAC has to maximize is the mean log income of all economic agents at the end of a simulation, corresponding to what an economic agent with constant relative risk aversion prefers. Figure 2.11 shows a typical histogram of the performance measure, based on 1,000 runs with identical tuned

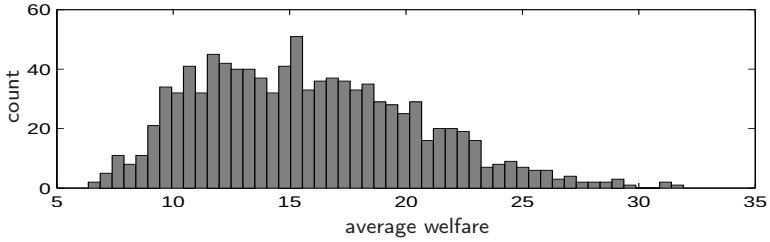


Figure 2.11: Histogram of EA performance, based on 1,000 runs.

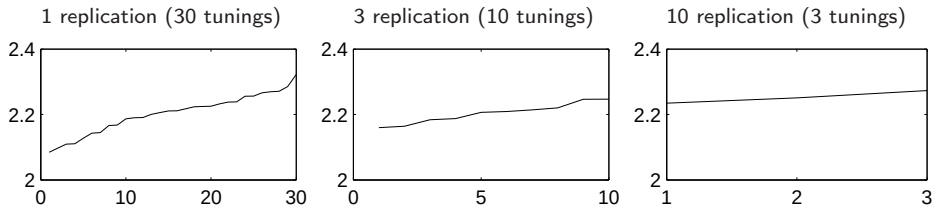


Figure 2.12: Distribution of the performance after tuning. The y-axis shows the average performance after tuning. The x-axes shows how many REVAC runs resulted in an average performance of that level or lower. Note that the median is similar in each graph.

parameter values. The distribution is skewed and has a flat tail, limiting the value of measurement replication. The distribution is not lognormal, but the estimated mean of the logarithmic performance seems to converge faster than the estimated mean of the performance itself and is a more reliable statistic. For this reason we average over the logarithm of the performance measure when we report the performance reached by different sets of tuned parameter values, even though the tuning is done in the original domain.

The algorithm layer has 6 parameters that need to be tuned, corresponding to the simplified evolutionary mechanism at the end of Section 4: *mutation probability*, *mutation variance*, *imitation probability*, *imitation ratio* (how much of the original strategy is perserved), *imitated fraction* (the fraction of well performing peers that are considered for imitation), and the *connectivity* of the social network. For the application layer we consider four different dynamic economic environments: changes occur sudden and with high frequency, sudden and with low frequency, gradual and with high frequency, and gradual and with low frequency.

We use REVAC with one, three and ten replications of measurements to tune the algorithm layer to each of the four economic environments. All other REVAC parameters are as described before. To improve the reliability of the tuning, we also look into the option of tuning the parameter values several times, choosing those tunings that achieved the highest performance, and averaging over the results. Due to limited computational resources we used different numbers of tunings for each replication scheme: 30 for 1 replication, 10 for 3 replications and 3 for 10 replications.

Figure 2.12 shows the average (log) performance that each tuning achieved during the last 10% of its measurements. Results are sorted per replication scheme to show how the tuned parameter values vary. With only 3 tunings in the case of 10 measurement replications no clear conclusion is possible, but a general trend is visible: the distribution of tuned parameter values is similar for all numbers of replication, with similar mean and variance. The same can be observed for each relevance estimate and each tuned parameter value: all tuning results follow a similar distribution, regardless of the number of measurement replications.

Since not all tuning sessions of REVAC achieve the same level of performance, we decide to take only the better 50% and average over the result. To compare REVAC with 1 measurement replication and with 3 measurement replications we start by randomly

Table 2.7: REVAC estimates. Average values in bold, followed by the measured variance.

	1 replication		3 replications		10 replications	
	Relevance estimate (absolute entropy)					
Mutation probability	**0.6**	0.4	**0.5**	0.2	**0.3**	0.2
Mutation variance	**0.3**	0.1	**0.2**	0.0	**1.2**	0.3
Imitation probability	**0.9**	0.3	**1.2**	0.3	**1.1**	0.4
Imitation ratio	**1.2**	0.5	**1.8**	1.0	**1.8**	1.0
Imitation fraction	**0.8**	0.4	**0.7**	0.1	**0.9**	0.5
Connectivity	**0.1**	0.0	**0.2**	0.1	**0.0**	0.0
All parameters	**3.9**	1.0	**4.6**	1.7	**5.3**	0.3
	Suggested parameter values					
Mutation probability	**0.20**	0.03	**0.21**	0.02	**0.45**	0.09
Mutation variance	**0.28**	0.02	**0.30**	0.03	**0.15**	0.01
Imitation probability	**0.83**	0.01	**0.86**	0.00	**0.85**	0.01
Imitation ratio	**0.90**	0.00	**0.93**	0.00	**0.93**	0.00
Imitation fraction	**0.85**	0.01	**0.77**	0.01	**0.83**	0.01
Connectivity	**0.54**	0.04	**0.58**	0.05	**0.51**	0.01

selecting 10 out of the 30 runs of REVAC with 1 measurement replication. Of these we take the better 5 and compare their average results to those of the better 5 from the implementation with 3 measurement replications. From the implementation with 10 measurement replications we only use the better 2 tunings. Table 2.7 shows the average relevance estimate (in absolute entropy) for every parameter, and the suggested value for each parameter (the median of the distribution) for one economic environment (sudden, low frequency). Average tuned values for each parameter are shown in bold, followed by the measured variance. Note how the measured variances for the different measurement replications are all of the same order.

To see if each REVAC implementation correctly differentiates between different problems in the application layer we apply each of the four tuned EAs a thousand time to each economic environment and average over the logarithm of the measured performance. This is done separately for each replication scheme. Table 2.8 shows the results. Each row stands for one economic environment and has four entries, showing the results when applying its own set of tuned parameter values and the other three sets of tuned parameter values to that environment. The bold values show the highest value for each row. With correct differentiation we expect to see the highest value for each economic environment when parameters are used that were tuned to that environment. As can be seen, this is almost always the case. The variance of the measured means is below 0.001 and therefore insignificant.

In general one can conclude that there is no significant difference in results obtained with 1, 3, or 10 measurement replication, even though in the case of 1 replication the total number of measurements is significantly smaller. With the exception of one environment, the tuned parameter values perform best on the application to which they

Table 2.8: Performance of the tuned simulation. Each row shows four sets of tuned parameter values applied to the same dynamic environment.

	Gradual, low frequency	Gradual, high frequency	Sudden, low frequency	Sudden, high frequency
	1 measurement replication, 10 runs of REVAC			
Gradual, low freq.	**2.628**	2.619	2.603	2.582
Gradual, high freq.	2.269	**2.539**	2.524	2.511
Sudden, low freq.	2.686	2.713	2.724	**2.727**
Sudden, high freq.	2.089	2.233	2.226	**2.256**
	3 measurement replications, 10 runs of REVAC			
Gradual, low freq.	**2.610**	2.591	2.597	2.584
Gradual, high freq.	2.375	**2.531**	2.520	2.512
Sudden, low freq.	2.710	2.716	**2.733**	2.704
Sudden, high freq.	2.102	2.247	2.230	**2.258**
	10 measurement replications, 3 runs of REVAC			
Gradual, low freq.	**2.625**	2.589	2.595	2.577
Gradual, high freq.	2.202	**2.540**	2.521	2.502
Sudden, low freq.	2.691	2.712	2.710	**2.713**
Sudden, high freq.	2.024	2.243	2.202	**2.261**

were optimized, indicating that REVAC is indeed able to tune parameter values to the problem at hand. One of the design goals of REVAC is to tune parameter values in a robust way so that they work well on problems that are similar to the problem they were tuned on. And indeed, all tuned parameter values achieve good results on all economic environments. To compare, without tuning, the system has a mean logarithmic performance of between 1.7 and 2, depending on the economic environment.

2.5 Comparing REVAC to other tuning methods

In this section we compare the EA parameter values tuned by REVAC to those found by hand-tuning and meta-GA. By hand tuning we mean that the practitioner chooses one or more vectors of parameter values for the EA, evaluates them, and uses the obtained information either to decide on the final parameterization, or to continue and evaluate more vectors of parameter values. This is arguably the most common tuning method even today. Despite the fact that hand tuning can be guided by common wisdom and the extensive experience of a practitioner, it is not effective. Grefenstette (1986) clearly showed that an EA that is tuned by a basic genetic algorithm (GA) outperforms all known hand tuned versions of the EA. Meta-GA tunes an evolutionary algorithm by optimizing a population of parameter vectors through selection, mutation, and recombination. Parent selection is fitness proportional, where the fitness of a vector of parameter values is the mean best fitness returned by the EA of the application layer when executed with

these values. Survival selection is generational, and a population size of 100 parameter vectors is used. To create offspring, one-point crossover is applied with a crossover rate of .5, and thereafter bit-flip mutation with a mutation rate of .001. We use a Gray code to represent the parameter values. The best parameter values are provided by the parameter vector of the last generation of the meta-GA that had the highest fitness.

Experimental setup of Section 2.5

design tool	*REVAC / meta-GA / hand tuning*
evolutionary algorithm	*Simple GA*
application problem	*Multi-modal problem generator*

To compare REVAC results with those of meta-GA and hand tuning, we follow Eiben et al. (2006) for the application and the algorithm layer. The authors tune a simple genetic algorithm by hand to maximize instances of the multi-modal problem generator (Spears, 2000). They also provide us with a benchmark performance of the hand-tuned algorithm. While the multi-modal problem generator is generally not adequate for assessing the performance of evolutionary algorithms (Lobo and Lima, 2006), Eiben et al. use it specifically to study the effect of different degrees of multi-modality on the performance of the simple GA.

The simple genetic algorithm of the algorithm layer uses a steady-state population model, uniform crossover, bit-flip mutation, tournament parent selection, and delete-worst survival selection. It terminates after 10,000 evaluations. The four free parameters of the algorithm are the crossover rate, the mutation rate, the population size, and the tournament size. The first two parameter values can take values between 0 and 1, encoded with 16 bits. The last two parameter values can take a value between 2 and 1025, encoded in 10 bits.

The multi-modal problem generator works as follows: generate n binary strings of length l to be the local optima. Different local optima are assigned different heights. A point x in the l-dimensional search space is evaluated by first finding the local optimum i with the lowest Hamming distance, i.e., the local optimum that matches x in the largest number of bits. The fitness of x is the fraction of matching bits scaled by the height of i. In case of ambiguity, the highest possible fitness is chosen. Here we use strings of $l = 100$ bits. Eiben et al. define ten different problem classes, each with a different number n of local optima. Those numbers are $\{1, 2, 5, 10, 25, 50, 100, 250, 500, 1,000\}$. The height of the global optimum is 1.

To tune the simple genetic algorithm, both REVAC and meta-GA are allowed 3,000 measurements per tuning session. We run REVAC with 3 measurement replications, so that it has performed 3,000 measurements by the time it has evaluated 1,000 parameter vectors. Table 2.9 shows the best parameter values found by each tuning method. Note that meta-GA and REVAC suggest much larger the values for population size and tournament size than the rather conventional values found by hand-tuning. For both meta-GA and REVAC we find that in two thirds of all solutions the tournament size equals the population size, effectively cancelling tournament selection from the algorithm, something a human designer is not likely to do.

To compare the performance of the best parameter vector of each tuning method, the simple GA is executed 25 times with each best parameter vector on the problem it

Table 2.9: Best parameter values found

	Crossover rate	Mutation rate	Population size	Tournament size
Hand tuned	0.5	0.01	100	2
Meta-GA	0.32	0.017	468	347
REVAC	0.41	0.0043	501	421

was tuned to, for each tuning method. The average performance in terms of mean best fitness is shown in Table 2.10. The highest performance per problem class is printed in bold. The observed differences have no statistical significance. No method performs better than any other on any problem class, which is nicely reflected by the random scatter of bold values over the table.

In our final experiment we measure how robust the tuned parameter vectors are against changes to the problem definition, which in this case amounts to changing the number of local optima. To do so we take a GA with parameter values that are tuned to a problem class with $n = x$ local optima, apply it to a problem class with $n = y$ local optima, and record the mean best fitness. Results are shown in Table 2.11. The average REVAC performance (0.993) seems to be better than that of meta-GA (0.991), but the difference does not have statistical significance.

2.6 Conclusions

In this chapter we introduced and evaluated a customized Estimation of Distribution Algorithm that uses differential Shannon entropy to estimate the relevance of EA parameters. The method searches for high-entropy distributions over the EA parameters that give high probability to parameter values with a high EA performance. Unlike most statistical optimization methods it does not depend on measurement replications to reduce variance. Instead, it reduces variance implicitly by averaging over *adjacent* vectors of parameter values. This allows it to get a good cover of the search space and to extract a high amount of information out of the available measurements. In terms of concrete parameter values, the median of these distributions provides a robust optimum, and the 25^{th} and 75^{th} percentile the confidence interval. The Shannon entropy of these distributions can be used to estimate how much tuning each parameter needs in order to to reach a given level of EA performance, independent of the actual tuning method.

The method proves to be able to reproduce the predefined relevance levels of abstract tuning problems to a satisfactory degree, even under high levels of measurement noise from a distribution with far outliers. REVAC results on a simple GA and standard numerical optimization problems are in line with what is reported in the literature. Tests on agent-based simulations of different dynamic economy-environments show that REVAC optimizes a 6-parameter evolutionary algorithm such that each vector of tuned parameter values performs best on the economy-environment it was tuned to, despite a high level of non-Gaussian system noise in the dynamic system. With regard to other tuning methods, we found that the performance of an algorithm tuned by REVAC is roughly comparable to the performance of the same algorithm when tuned by meta-GA.

Table 2.10: Average mean best fitness of the GA tuned with each method

	Number of peaks of the problem class									
	1	2	5	10	25	50	100	250	500	1,000
Hand-tuned	**1**	**1**	**1**	**.996**	.989	.988	.985	.985	.987	**.989**
Meta-GA	**1**	**1**	.988	.993	**.994**	.994	.983	**.992**	**.989**	.987
REVAC	**1**	**1**	**1**	**1**	.991	**.995**	**.989**	.966	.970	.985

Table 2.11: Mean best fitness when cross-validating the tuned parameter values

	Number of local optima the tuned parameters are applied to										
	1	2	5	10	25	50	100	250	500	1,000	mean
	meta-GA										
1	1	1	1	1	.994	1	.999	.986	.986	.988	.995
2	1	1	1	1	1	.980	.977	.981	.991	.998	.993
5	.977	.980	1	.966	.987	.987	.959	.958	.963	.958	.974
10	1	1	1	1	.970	.980	.975	.992	.988	.995	.990
25	1	1	1	1	.994	.983	.989	.986	.994	.987	.993
50	1	1	1	1	1	1	.994	.985	.997	.982	.996
100	1	1	1	1	1	.997	.985	.974	.992	.997	.994
250	1	1	1	1	.987	.997	.995	.970	.986	.970	.990
500	1	1	1	1	.987	.987	.992	.990	.993	.990	.994
1000	1	1	1	.982	1	.980	.985	.995	.992	.982	.992
mean	.998	.998	1	.995	.992	.989	.985	.982	.988	.985	.991
	REVAC										
1	1	1	1	1	.990	.990	.998	.990	.994	.991	.995
2	1	1	1	1	.985	.983	.985	.988	.997	.989	.993
5	1	1	1	1	.974	.983	.990	.988	.971	.971	.988
10	1	1	1	1	1	.995	.974	.989	.993	.995	.995
25	1	1	1	1	.990	.998	.994	.994	.988	.996	.996
50	1	1	1	1	1	.975	.990	.991	.993	.995	.994
100	1	1	1	1	.995	.993	.980	.986	.980	.992	.993
250	1	1	1	1	.990	.985	.975	.987	.990	.988	.992
500	1	1	1	.987	.995	.975	.980	.992	.999	.968	.990
1000	1	1	1	1	1	.988	.994	.996	.975	.970	.992
mean	1	1	1	.999	.992	.986	.986	.990	.988	.986	.993

Notes. Rows show the problem class to which the parameters are tuned (labeled by the number of local optima n), columns show the problem class to which the parameters are applied (labeled by the number of local optima n).

REFERENCES

Bartz-Beielstein, T., Lasarczyk, C. W. G., Preuss, M., 2005. Sequential parameter optimization. In: IEEE Congr. Evol. Comput., CEC'05. IEEE, Edinburgh, UK, pp. 773–780.

Box, G. E. P., Wilson, K. B., 1951. On the experimental attainment of optimum conditions. J. R. Stat. Soc. B 13 (1), 1–45.

Czarn, A., MacNish, C., Vijayan, K., Turlach, B., Gupta, R., 2004. Statistical Exploratory Analysis of Genetic Algorithms. IEEE Trans. Evol. Comput. 8 (4), 405–421.

de Landgraaf, W. A., Eiben, A. E., Nannen, V., 2007. Parameter calibration using meta-algorithms. In: IEEE Congr. Evol. Comput., CEC'07. IEEE, Singapore, pp. 71–78.

Eiben, A. E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3 (2), 124–141.

Eiben, A. E., Schut, M. C., de Wilde, A. R., 2006. Is self-adaptation of selection pressure and population size possible?–A case study. In: Parallel Probl. Solving Nat., PPSN IX. Springer, Reykjavik, Iceland, pp. 900–909.

Eiben, A. E., Smith, J. E., 2003. Introduction to Evolutionary Computing. Springer, Berlin / Heidelberg.

François, O., Lavergne, C., 2001. Design of evolutionary algorithms-a statistical perspective. IEEE Trans. Evol. Comput. 5 (2), 129–148.

Freisleben, B., Hartfelder, M., 1993. Optimization of genetic algorithms by genetic algorithms. In: Albrecht, R. F., Reeves, C. R., Steele, N. C. (Eds.), Artificial Neural Networks and Genetic Algorithms. Springer, Berlin / Heidelberg, pp. 392–399.

Grefenstette, J. J., 1986. Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man and Cybernet 16 (1), 122–128.

Jong, K. A. D., 1975. An analysis of the behaviour of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan.

Jong, K. A. D., 2006. Evolutionary Computation: A Unified Approach. MIT Press, Cambridge, MA.

Lobo, F. G., Lima, C. F., 2006. On the Utility of the Multimodal Problem Generator for Assessing the Performance of Evolutionary Algorithms. In: Proc. Genet. Evol. Comput. Conf., GECCO'06. ACM, Seattle, Washington, pp. 1233–1240.

Mühlenbein, H., 1997. The equation for response to selection and its use for prediction. Evol. Comput. 5 (3), 303–346.

Mühlenbein, H., Höns, R., 2005. The estimation of distributions and the minimum relative entropy principle. Evol. Comput. 13 (1), 1–27.

Nannen, V., Eiben, A. E., 2007a. Efficient relevance estimation and value calibration of evolutionary algorithm parameters. In: IEEE Congr. Evol. Comput., CEC'07. IEEE, Singapore, pp. 103–110.

Nannen, V., Eiben, A. E., 2007b. Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: Proc. Int. Jt. Conf. Artif. Int., IJCA'07. AAAI Press, Hyderabad, India, pp. 975–980.

Schaffer, J., Caruana, R., Eshelman, L., Das, R., 1989. A study of control parameters affecting online performance of genetic algorithms for function optimization. In: Proceedings of the third international conference on genetic algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 51–60.

Shannon, C. E., 1948. A mathematical theory of communication. Bell Syst. Tech. J. 27, 379–423– 623–656.

Spears, W. M., 2000. Evolutionary Algorithms: The Role of Mutation and Recombination. Springer, Berlin / Heidelberg.

Taguchi, G., Wu, Y., 1980. Introdution to Off-Line Quality Control. Central Japan Quality Control Association, Nagoya, Japan.