Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters

Volker Nannen and A.E. Eiben
Department of Computer Science, Vrije Universiteit Amsterdam
{volker, gusz}@cs.vu.nl

Abstract

The main objective of this paper is to present and evaluate a method that helps to calibrate the parameters of an evolutionary algorithm in a systematic and semi-automated manner. The method for Relevance Estimation and Value Calibration of EA parameters (REVAC) is empirically evaluated in two different ways. First, we use abstract test cases reflecting the typical properties of EA parameter spaces. Here we observe that REVAC is able to approximate the exact (hand-coded) relevance of parameters and it works robustly with measurement noise that is highly variable and not normally distributed. Second, we use REVAC for calibrating GAs for a number of common objective functions. Here we obtain a common sense validation, REVAC finds mutation rate p_m much more sensitive than crossover rate p_c and it recommends intuitively sound values: p_m between 0.01 and 0.1, and $0.6 \leq p_c \leq 1.0$.

1 Introduction

Appropriate calibration of evolutionary algorithm (EA) parameters is essential to good EA performance. Nevertheless, current practice in EA calibration is based on ill-justified conventions and ad hoc methods. Without exaggeration, one could state that one of the canonical design problems in evolutionary computing (EC)is: How to choose operators and parameters for an evolutionary algorithm to ensure good performance? The related questions can concern relevance of operators and parameters as well as the values of (relevant) parameters. 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]. Relevance of parameters is also an issue, e.g., 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.[1]

In contemporary practice, EA parameters are set by common "wisdom", e.g., mutation size should be 1 divided by the chromosome length, or by statistical hypothesis testing, e.g., comparing EA performance for a number of different setups [Eiben *et al.*, 1999]. This is typically a laborious, ad hoc procedure involving much handwork and heuristic choices. So far, no procedure has yet been established that can do this systematically, exploring the full combinatorial range of possible parameter settings. Another problem is that studies on design aspects like confidence intervals for good parameter values and a sensitivity analysis for parameter robustness are scarce. In this paper we discuss a design method that calibrates the parameters of an EA in a robust way, regardless of the distribution of the results. For each parameter the method produces a distribution over the parameter's range that give high probability to values leading to good EA performance. In this scheme, a distribution with a narrow peak indicates a highly relevant parameter (whose values largely influence EA performance), while a broad plateau belongs to a moderately relevant parameter (whose values do not matter too much).

Related work includes meta-GAs as an early attempt to automate GA calibration [Greffenstette, 1986], and [Eiben *et al.*, 1999] that established parameter control in EAs as one of the big challenges in EC. Czarn et al. [2004] discuss current problems in EA design and use polynomial models of a response curve to estimate confidence interval for parameter values. François and Lavergne [2001] estimate response curves for EAs across multiple test cases to measure generalizability. The groundwork of statistical experimental design was laid by R.A. Fisher in the 1920s and 1930s. Response surfaces methods that exploit sequential sampling were introduced in Box and Wilson [1951]. A paradigm shift that emphasizes parameter robustness is due to G. Taguchi [1980]. A survey of optimization by building probabilistic models is given in [Pelikan *et al.*, 2002].

2 The REVAC Method

The REVAC method is based on information theory to measure parameter relevance. Instead of estimating the performance of an EA for different parameter values or ranges of values[2], the method estimates the expected performance when parameter values are chosen from a probability density distribution $\mathcal{C}$ with maximized Shannon entropy. This maxi-

[1]These examples are purely hypothetical.

[2]In the jargon of statistical experimental design a parameter is called *factor*. A range of parameter values is called *level*.

mized Shannon entropy is a measure of parameter relevance. In information theoretical terms we measure how much information is needed to reach a certain level of performance, and how this information is distributed over the different parameters. In these terms the objectives of the REVAC method can be formulated as follows:

- the entropy of the distribution $\mathcal{C}$ is as high as possible for a given performance
- the expected performance of the EA in question is as high as possible

Technically, the REVAC method is an Estimation of Distribution Algorithm (EDA) [Pelikan *et al.*, 2002] that is specifically designed to measure maximized entropy in the continuous domain. Given an EA with k (strategy) parameters the REVAC method iteratively refines a joint distribution $\mathcal{C}(\vec{x})$ over possible parameter vectors $\vec{x} = \{x^1, \ldots, x^k\}$. Beginning with a uniform distribution $\mathcal{C}^0$ over the initial parameter space $\mathcal{X}$, the REVAC method gives a higher and higher probability to regions of $\mathcal{X}$ that increase the expected performance of the associated EA. This is to increase the EA performance. On the other hand, the REVAC method maximizes entropy by *smoothing* the distribution $\mathcal{C}$. For a good understanding of how REVAC works it is helpful to distinguish two views on a set of parameter vectors as shown in Table 1. Taking a *horizontal* view on the table, a row is a parameter vector and we can see the table as m of such vectors $X = \{\vec{x_1}, \ldots, \vec{x_m}\}$. Taking the *vertical* view on the columns, column i shows m values for parameter i from its domain. These values naturally define[3] a marginal density function $\mathcal{D}(x)$ over the domain of parameter i, hence we can see the k columns of the table as k marginal density functions $X = \{\mathcal{D}^1, \ldots, \mathcal{D}^k\}$.

	$\mathcal{D}^1$	$\cdots$	$\mathcal{D}^i$	$\cdots$	$\mathcal{D}^k$
$\vec{x_1}$	$\{x_1^1$	$\cdots$	x_1^i	$\cdots$	$x_1^k\}$
$\vdots$		$\ddots$			
$\vec{x_j}$	$\{x_j^1$	$\cdots$	x_j^i	$\cdots$	$x_j^k\}$
$\vdots$				$\ddots$	
$\vec{x_m}$	$\{x_m^1$	$\cdots$	x_m^i	$\cdots$	$x_m^k\}$

Table 1: A table X of m vectors of k parameters

Roughly speaking, REVAC works by iteratively improving an initial table X_0 that was drawn from the uniform distribution over $\mathcal{X}$. Creating a new table X_{t+1} from a given X_t can be described from both the horizontal and the vertical perspective.

[3]Scaled to the unit interval $[0, 1]$ we define the density over the m+1 intervals between any two neighbors (including limits) x_a, x_b as $\mathcal{D}(x) = \frac{(m+1)}{|x_a - x_b|}$ with $\int_0^1 \mathcal{D}(x) = 1$ and differential entropy

$$H(\mathcal{D}) = -\int_0^1 \mathcal{D}(x) \log \mathcal{D}(x)$$

with $H(\mathcal{D}) = 0$ for the uniform distribution over $[0, 1]$. The sharper the peaks, the lower the entropy.

Looking at the table from the *horizontal* perspective we can identify two basic steps:

1. **Evaluating parameter vectors:** Given a parameter vector $\vec{x}$ we can evaluate it: the utility of $\vec{x}$ is the performance of the EA executed with these parameter values.
2. **Generating parameter vectors:** Given a set of parameter vectors with known utility we can generate new ones that have higher expected utility.

Step 1 is straightforward, let us only note that we call the performance that an EA achieves on a problem using parameters $\vec{x}$ the *response*. Response r is thus a function $r = f(\vec{x})$; the surface of this function is called a *response surface*. As for step 2, we use a method that is evolutionary itself, (but should not be confused with the EA we are calibrating). We work with a population of m parameter vectors. A new population is created by selecting $n < m$ parent vectors from the current population, recombining and mutating the selected parents to obtain a child vector, replacing one vector of the population.

We use a deterministic choice for parent selection as well as for survivor selection. The best n vectors of the population are selected to become the parents of the new child vector, which always replaces the oldest vector in the population. Only one vector is replaced in every generation. 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 is calculated, then a random value is chosen from this interval. To define the mutation interval for mutating a given x_j^i all other values $x_1^i, \ldots, x_n^i$ for this parameter in the selected parents are also taken into account. After sorting them in increasing order, the begin point of the interval can be specified as the h-th lower neighbor of x_j^i, while the end point of the interval is the h-th upper neighbor of x_j^i. The new value is drawn from this interval with a uniform distribution.

From the *vertical* perspective we consider each iteration as constructing k new marginal density functions from the old set $X_t = \{\mathcal{D}_t^1, \ldots, \mathcal{D}_t^k\}$. Roughly speaking, new distributions are built on estimates of the response surface that were sampled with previous density functions, each iteration giving a higher probability to regions of the response surface with higher response levels. Each density functions is constructed from n uniform distributions over overlapping intervals. In this context, the rationale behind the complicated mutation operator is that it heavily smoothes the density functions. Like all evolutionary algorithms, REVAC is susceptible for converging on a local maximum. By consistently smoothing the distribution functions we force it to converge on a maximum that lies on a broad hill, yielding robust solutions with broad confidence intervals. But smoothing does more: it allows REVAC to operate under very noise conditions, it allows REVAC to readjust and relax marginal distributions when parameters are interactive 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 h-th neighbors of x_j^i when defining the mutation interval[4]. Choosing a good value for h is an important aspect when using the REVAC method. A large h value can slow down convergence to the point of stagnation. A small h value can produce unreliable results. We prefer $h \approx n/10$.

Because the REVAC method is implemented as a sequence of distributions with slowly decreasing Shannon entropy, we can use the Shannon entropy of these distributions to estimate the minimum amount of information needed to reach a target performance level. We can also measure how this information is distributed over the parameters, resulting in a straightforward measure for parameter relevance. This measure can be used in several ways. First, it can be used to choose between different operators [Nannen and Eiben, 2006]. An operator that needs little information to be tuned is more fault tolerant in the implementation, easier to calibrate and robuster against changes to the problem definition. Second, it can be used to identify the critical parts of an EA. For this we calculate relevance as the normalized entropy, i.e., the fraction of total information invested in the particular parameter. When an EA needs to be adapted from one problem to another, relevant parameters need the most attention. With this knowledge, the practitioner can concentrate on the critical components straight away. Third, it can be used to define confidence intervals for parameter choices. 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 give the 25^{th} and the 75^{th} percentile of the distribution as a confidence interval for the parameter. 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.

Further documentation, a Matlab implementation and graphical demonstrations are available on the web sites of the authors: http://www.few.vu.nl/∼volker/revac and http:/www.few.vu.nl/∼gusz

3 Empirical Assessment of REVAC

For an elaboration on how to evaluate the REVAC method method we can distinguish 3 layers in using an EA:

Application layer The problem(s) to solve.

Algorithm layer The EA with its parameters operating on objects from the application layer (candidate solutions of the problem to solve).

Design layer The REVAC method operating on objects from the algorithm layer (parameters of the EA to calibrate).

A real *in vivo* assessment requires that we select a problem, or a set of problems, and execute real runs of an EA on this problem(s) for each evaluation of given parameter vectors. This can lead to excessive running times for REVAC and would deliver information on how REVAC works *on the given problem (s)*. As a second stage, we would need to make a generalization step to claim something about how REVAC works *in general*. Alternatively, we can make the generalization step first and create an abstract response surface that is representative for EA parameter spaces. Hereby we abstract from the application and algorithm layers and reduce run times enormously. Either way, there is a generalization step involved.

In this paper we perform both types of experiments, but put the emphasis on the second option. To define abstract response surfaces resembling the response surface of a typical EA we identify five essential properties:

1. Low dimensionality: The number of parameters to calibrate is in the order of magnitude of 10.
2. Non-linearity (epistasis, cross-coupling): Parameters interact in non-linear ways, implying that the response curves are non-separable, 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 response surface has one or few local optima, i.e., few regions of the parameter space have high EA performance.
5. Noise: Depending on the application, the performance of a GA can be highly variable and can follow a distribution that is far from the normal distribution.

In section 4 we present experiments on abstract response surfaces having these five properties. Thereafter, in section 5 results obtained with concrete objective functions are given. In all cases REVAC is used with 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 h = 5 upper and lower neighbors. In all experiments, REVAC is allowed to perform 1000 evaluations. For a test with a concrete objective function f this means 1000 runs of the EA on f (evaluating 1000 different parameter vectors).

4 Results on abstract EA response surfaces

We present three experiments: two on the accuracy of the relevance estimation (without noise) and one on the resistance of the method to noise. We use $k = 10$ parameters that can take values from the range $[0, 1]$.

Experiment 1: hierarchical dependency. In this experiment we pre-define and hard-code the dependency between parameters. The response $r = f(\vec{x})$, showing the utility of $\vec{x}$, is calculated as the sum over the responses per parameter. For the first parameter value x^1 the response r^1 is equal to 1 minus the distance to a target value, randomly chosen at the start of the experiment. The response of each consecutive parameter value is $r^i = r^{i-1}(1 - |x^i - x^{i-1}|) \in [0, 1]$. The response of parameter i is higher if its value is closer to that of parameter $i - 1$. But because it is multiplied by the response of that parameter, we have hierarchical dependencies. The better the first parameter is calibrated, the more the calibration of the second parameter can contribute, and so forth. Knowledge of such a hierarchy of dependency and the associated hierarchy in relevance is very useful for the practitioner, and we are interested if the REVAC method can reveal it.

Figures 1 and 2 show the results for one run of REVAC (1000 evaluations). The main quality indicator for REVAC is

[4]At the edges of the parameter ranges are no neighbors. We solve this problem by mirroring neighbors and chosen values at the limits, similar to what is done in Fourier transformations.

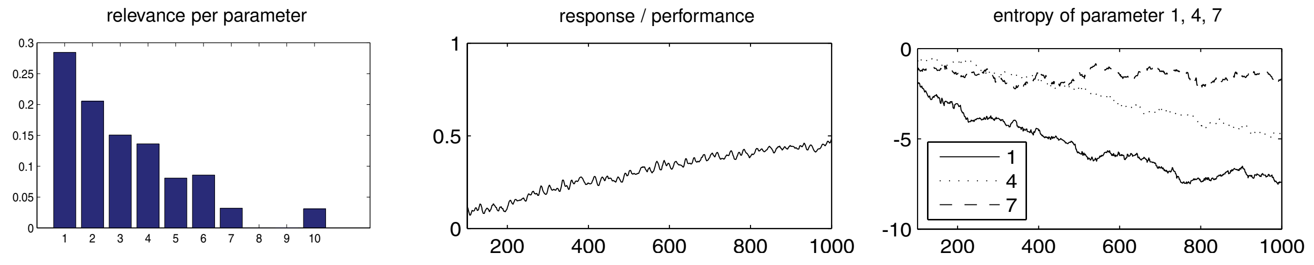

Figure 1: Utility and relevance of parameters. The left graph shows the relevance (normalized entropy) per parameter at the end, after 1000 REVAC evaluations. The middle graph shows how the measured response (parameter utility as the abstract EA performance) increases through the calibration process. The rightmost graph shows the development of entropy for parameters 1, 4, and 7. For both, the x-axis shows evaluations 101–1000 (the first 100 concern the initial population).

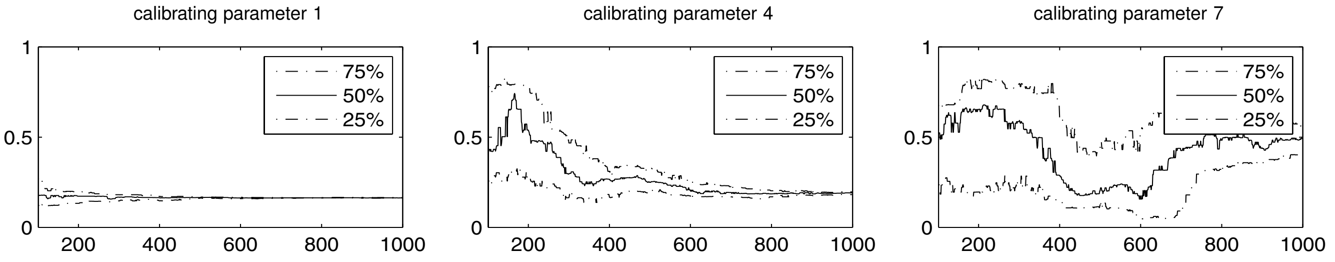

Figure 2: Calibration of parameters illustrated. The y-axis shows the value for the 25^{th}, 50^{th} and 75^{th} percentile of the distribution. The calibration effect is achieved by taking parameter values from the range between the 25^{th} and 75^{th} percentile. Note that the length of this confidence interval indicates parameter relevance: narrow/broad = highly/moderately relevant. Here parameter 1 shows highly relevant, parameter 7 does not.

the leftmost histogram in Figure 1 that shows how the relevance estimates for consecutive parameters approximate the predefined linear relationship. The rightmost graph of Figure 1 and the three graphs of Figure 2 show various aspects of the calibration process for parameters 1, 4, and 7.

Experiment 2: predefined relevance distribution. In this experiment we control the response contribution of each parameter directly to see whether the REVAC method can reveal arbitrary relationships in relevance. To this end we create a response curve by placing 1 peak randomly in the 10-dimensional unit space. Again, total response is a sum over individual responses, which is 1 minus the distance to this peak per parameter (i.e., Hamming distance). We weight each distance with a predefined target relevance. In this way, an irrelevant parameter has no contribution to the response, no matter how close its value is to the peak. On the other hand, the higher the target relevance of a parameter, the more its calibration contributes to the overall response.

Using three sets of weights we specify three predefined relevance distributions. These are given together with the experimental results in Figure 3, cf. black vs. white columns. The shown outcomes are from single runs of REVAC (1000 evaluations). The accuracy of results can be improved by averaging over the results of several runs. As can be seen in Figure 3, the REVAC method can reproduce a predefined relevance distribution quite accurately. The relationship in the right graph of Figure 3 is the most typical situation when calibrating real EAs, also known as the sparcity of effects principle: relevance is concentrated on only a few parameters. As can be seen, the REVAC method can identify the more relevant ones and also reproduce the correct order in relevance. However, it does not reproduce the fact that the most relevant parameter is twice as relevant as the follow up. This can not be repaired by averaging the results of several runs of REVAC.

Experiment 3: measurement noise. In this experiment we study how the reliability of a relevance estimation changes with noise of increasing variance. To measure the quality of the estimate we use the mean squared error between a relevance estimate to the target distribution, which we plot against the variance of the noise.

The abstract response surface we use here is the third one (non-linear increase) from experiment 2, see Figure 3, right. Addition of noise turns the abstract response curve into a stochastic equation $r = f(\vec{x}) + \eta$, where η is an i.i.d. random value. The noise is drawn from a Pareto distribution

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

with exponent $\gamma = 2.3$. c is a normalization value and also 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, a slowly convergent variance, and is generally incompatible with statistical methods that require a normal distribution.

The variance of i.i.d. noise can be reduced by averaging over multiple evaluations. The REVAC method however does not average over multiple evaluations. Rather, if allowed more evaluations, it uses them to get a better cover of the sampling space. Noise effects are only treated during the smoothing of the distributions.

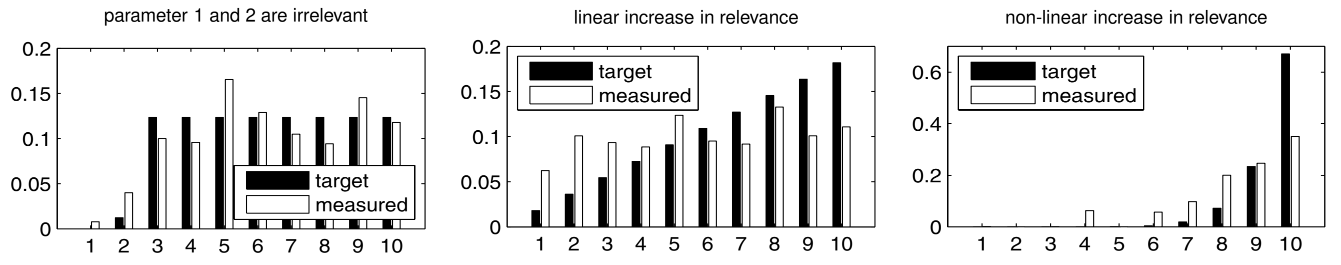

Figure 3: Comparing the relevance estimation of the REVAC method (white) to the hard-coded target relevance (black).

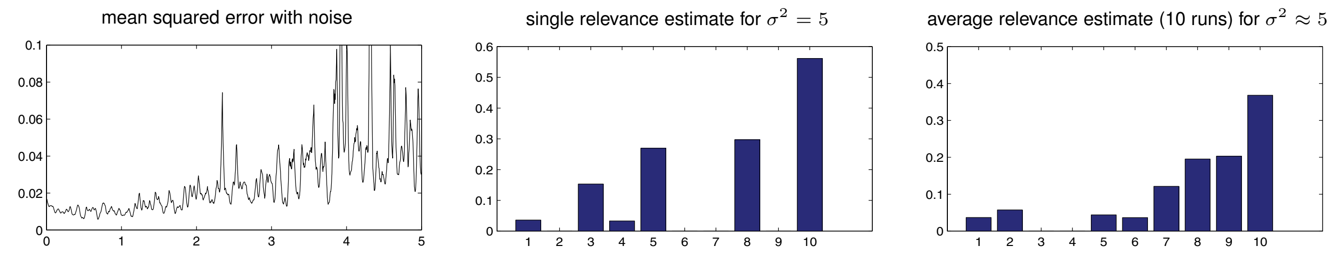

Figure 4: Impact of noise. The left graph shows the mean squared error of the relevance estimate as a function of the variance of the noise. The x-axis shows the increasing variance, from 0 to 5. (5 is very high for a function with a range between 0 and 1). The y-axis shows the results of 1000 different runs of the REVAC method over these noise levels. (The expected mean squared error of a completely random relevance estimate is 0.29.) The middle graph shows the estimate voor $\sigma^2 = 5$. The right graph is averaged over the last 10 evaluations. Note how averaging over multiple runs of REVAC recovers the main features of the target distribution (black columns in the rightmost graph of Figure 3, right above it).

Figure 4 shows the results of this experiment. As we increase the variance of the additive noise term, the mean squared error of the REVAC estimate increases roughly linearly with the variance. This error can be reduced by averaging over multiple runs of REVAC. That means that under noisy conditions REVAC can give a quick first approximation that can be refined by averaging over repeated calibrations. This leads to an effective use of all evaluations. The fact that the noise follows a non-normal distribution does not form a significant obstacle.

5 Results on concrete objective functions

Here we present the outcomes of *in vivo* experiments, as discussed in section 3. To this end we need to define an EA and some specific objective functions. For both purposes we rely on [Czarn *et al.*, 2004] who apply rigorous statistical exploratory analysis to a simple GA to study the effect of calibrating the mutation and crossover operators. The EA here is a generational GA with 22 bits per variable, Gray coding, probabilistic ranked-based selection, single point crossover with p_c, bit flip mutation with p_m, and a population of 50 chromosomes. The four test functions are rather standard: sphere, saddle, step, Schaffer's f_6. Our results are summarized in Table 2 and Figure 5. These can be considered from two perspectives, compared with the "usual" GA settings, and with the work of Czarn *et al.* As Table 2 shows the values found by REVAC are consistent with the conventions in EC: p_m between 0.01 and 0.1, and p_c between 0.6 and 1.0.

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

$$f_2(x) = \begin{array}{l} 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \\ -2.048 \leq x_i \leq 2.048 \end{array} \tag{2}$$

$$f_3(x) = \sum_{i=1}^{5} \lfloor x_i \rfloor, \quad -5.12 \leq x_i \leq 5.12 \tag{3}$$

$$f_6(x) = \begin{array}{l} 0.5 + \dfrac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1 + 0.0001(x_1^2 + x_2^2))^2}, \\ -100 \leq x_i \leq 100 \end{array} \tag{4}$$

Table 2: REVAC results after 1000 evaluations.

	optimum value	confidence interval	entropy	relevance
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

A direct comparison with [Czarn *et al.*, 2004] is difficult, because of the different types of outcomes. As for the method, they use screening experiments to narrow down the space of feasible parameter settings, partition this space into

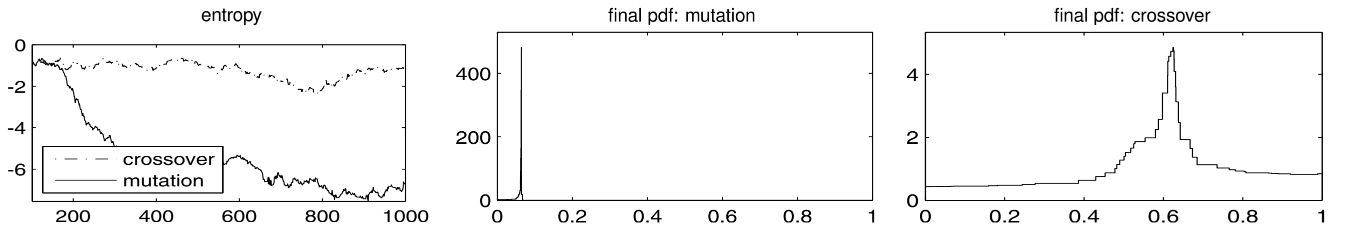

Figure 5: Calibrating Schaffer's f_6. The left graph shows the development of entropy for p_m and p_c. The entropy of p_m shows significant decrease, indicating its relevance; that of p_c does not. The other two graphs show the final probability density function (pdf) over the parameter values. Note the extremely sharp needle (high sensitivity) for p_m.

equally spaced discrete levels, repeatedly measure 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, the REVAC method uses no screening and does not partition the parameter space into discrete levels. It studies the complete and continuous parameter space. We 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.

6 Conclusions

In this paper we propose and evaluate a specialized Estimation of Distribution Algorithm (EDA) that estimates parameter relevance by normalized Shannon entropy. The method finds a distribution (marginal density function) for each EA parameter simultaneously such that parameter values yielding higher EA performance have higher probabilities. The method shows the *relevance of a parameter* graphically by the width of the peak(s): a distribution with a narrow peak indicates a highly relevant parameter (whose values have great influence on EA performance), while a broad plateau belongs to a moderately relevant parameter (whose values do not matter too much). *Parameter calibration* is achieved in a straightforward way, the values with a high probability are the ones to be used. In particular, for each parameter we choose choose the 50^{th} percentile as the robust optimum and the 25^{th} and 75^{th} percentiles as the confidence interval.

Our empirical validation is twofold. First, we use abstract test cases representing typical response surfaces of EA parameter spaces and observe that estimates reproduce the target values to a satisfiable degree. We also show that REVAC can work with significant levels of highly variable noise. The error of relevance estimates increases roughly linearly with the variance of the noise and can be reduced by averaging over several estimates. Second, we use REVAC to calibrate a GA on four common test functions and find that the recommended values are much in line with the usual settings, "optimized" by collective experience of the field.

Ongoing research is carried out along two lines. The first one concerns real-life testing, where we use the method to evaluate policies in complex economic simulations. The second one is a comparative study, where we calibrate genetic algorithms on a number of objective functions by REVAC and compare it to a meta-GA [Greffenstette, 1986].

References

[Box and Wilson, 1951] G. E. P. Box and K. G. Wilson. On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society, Series B (Methodological)*, 13(1):1–45, 1951.

[Czarn *et al.*, 2004] Andew Czarn, Cara MacNish, Kaipillil Vijayan, Berwin Turlach, and Ritu Gupta. Statistical Exploratory Analysis of Genetic Algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4):405–421, 2004.

[Eiben and Smith, 2003] 5BA. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, Heidelberg, New York, November 2003.

[Eiben *et al.*, 1999] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[François and Lavergne, 2001] Olivier François and Christian Lavergne. Design of Evolutionary Algorithms—A Statistical Perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148, 2001.

[Greffenstette, 1986] J. Greffenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

[Nannen and Eiben, 2006] Volker Nannen and A. E. Eiben. A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms. In Maarten Keijzer *et al.*, editors, *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 183–190, New York, 2006. ACM.

[Pelikan *et al.*, 2002] Martin Pelikan, David E. Goldberg, and Fernando Lobo. A Survey of Optimization by Building and Using Probabilistic Models. *Computational Optimization and Applications*, 21(1):5–20, 2002. Also IlliGAL Report No. 99018.

[Taguchi and Wu, 1980] Genichi Taguchi and Y. Wu. *Introdution to Off-Line Quality Control*. Central Japan Quality Control Association, Nagoya, Japan, 1980.