A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms

Volker Nannen*
Department of Computer Science
Vrije Universiteit Amsterdam
volker@cs.vu.nl

A.E. Eiben
Department of Computer Science
Vrije Universiteit Amsterdam
gusz@cs.vu.nl

ABSTRACT

We present and evaluate a method for estimating the relevance and calibrating the values of parameters of an evolutionary algorithm. The method provides an information theoretic measure on how sensitive a parameter is to the choice of its value. This can be used to estimate the relevance of parameters, to choose between different possible sets of parameters, and to allocate resources to the calibration of relevant parameters. The method calibrates the evolutionary algorithm to reach a high performance, while retaining a maximum of robustness and generalizability. We demonstrate the method on an agent-based application from evolutionary economics and show how the method helps to design an evolutionary algorithm that allows the agents to achieve a high welfare with a minimum of algorithmic complexity.

Categories and Subject Descriptors
I.6.5 [Simulation and Modelling]: Model Development I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms
Algorithms, Experimentation, Performance, Economics

Keywords
Parameter control, evolutionary algorithms, agent-based simulations, model selection, information theory

1. INTRODUCTION

One of the canonical challenges in evolutionary computing is to select and calibrate parameters of an evolutionary algorithm (EA) [6, 7], i.e., parameters that regulate variation (mutation and recombination), selection, population size, and so on. Often these parameters need to be optimized such that the EA delivers good and robust solutions for a whole family of similar problems. This is true for the "traditional" optimization and design applications. For instance, when solving a scheduling task with a genetic algorithm, it can be hard to establish good values for the mutation rate, crossover rate, tournament size and population size that give good solutions for all possible problem instances. The problem intensifies in more complex applications like agent-based simulations used for artificial life, artificial societies and evolutionary economics. In such applications evolution is not only the "problem solver" that is expected to lead to "optimality" in some application specific sense. It also has to fit in with the general system description and provide a better understanding of the general dynamics of the evolutionary system under investigation. This often forces the evolutionary algorithm to include problem specific features that need to be correctly parametrized. For instance, mating selection might depend on past interactions between individuals, or mutation might be sensitive to environmental factors. When selecting and calibrating the parameters for such complex evolutionary systems, common EA wisdom (heuristics and conventions learned over the last decades) regarding EA setups is hardly applicable, since this wisdom is mainly based on applications of the traditional type.

In this paper we present the parameter calibration and relevance estimation (CRE) method, a numerical method to select and calibrate the parameters of any evolutionary algorithm that strives for general validity. The rationale behind the CRE method is that the simpler the calibration, the broader the set of different problems it can be applied to with success. This principle has long been known as Occam's Razor and has been put on sound mathematical footing by the theory of algorithmic complexity, formulated simultaneously by R. Solomonoff [17], A. Kolmogorov [12] and G. Chaitin [4, 5]. Unfortunately, the algorithmic complexity of an arbitrary set of values is not computable. To avoid this problem, the CRE method works on distributions over parameter values and not on the actual values themselves. The algorithmic complexity of values drawn from a distribution can be estimated directly from the Shannon entropy of that distribution [16]. The CRE method uses the Shannon entropy to estimate the relevance of a parameter.

The objectives of the CRE method are:

1. To estimate how sensitive the performance of the evolutionary algorithm is to the used values of each parameter.
2. To calibrate the parameters, that is, to identify good values for them.

We demonstrate the CRE method with an agent-based application from evolutionary economics. The application is

*Long term visitor at the Multi-Agent Systems Division at ISI, Turin, Italy

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

needed for proof of concept but we could have implemented any other evolutionary process. We provide a brief summery of the agent-based application, the non-linear system dynamics that the agents have to adapt to, and the specific evolutionary algorithm which consists of random innovation (mutation) and selective imitation (recombination) in a social peer network. We describe our initial evolutionary algorithm of 13 parameters, reflecting our best intuitions on the evolutionary dynamics in the given context. Next we describe the details of the CRE method and demonstrate how the CRE method effectively disproves our initial intuitions. Instead, it leads us to a simplified evolutionary algorithm with only 6 parameters that reaches the same level of performance. Because it does so with less complexity and information, we conclude that its predictive power and general validity have improved.

Related Work. Eiben e.a. established parameter control in EAs as an important research question [6]. A practical method to find good parameter settings for an EA was introduced by François and Lavergne in [10], using numerical simulations to estimate the functional relationship between parameter values and the performance of the EA, both for a single test case and for multiple test cases (generalization).

Some fundamental insights by A. Kolmogorov on the relation between individual data and (probabilistic) sets that contain them where published only recently [20]. Early attempts to relate the generalization power of a statistical model to some practical estimate of algorithmic complexity were based on the number and precision of the parameters involved: first the Akaike Information Criterion (AIC) [1] and then Rissanen's theory of Minimum Description Length (MDL) [15]. Later, J. Rissanen, A. Barron and B. Yu developed a version of MDL based on parametric complexity [3]. All these methods are based on a functional analysis of the statistical model in question. This is not possible here, simply because no analytic tool can tell us how many previously unsolvable problems can be solved by adding feature x to an EA. The CRE method is intended to fill this gap by numerical estimation.

2. THE AGENT-BASED APPLICATION

The agent-based application treated here is concerned with a fixed number of economic agents that operate in a simulated economic environment [14]. The welfare or fitness of the agents is determined by their individual investment strategies. Each cycle t of a simulation is divided into two steps: *1) updating the economy*—in accordance with its respective investment strategy $S_{a,t}$ each agent a distributes its income $Y_{a,t}$ over consumption and a number of capital sectors. A fixed set of growth and production functions is then used to calculate the new income $Y_{a,t+1}$ for each agent. We use the consumption of an agent to measure the welfare or fitness $W_{a,t}$ of that agent. *2) updating the strategies*—all agents change their respective investment strategy $S_{a,t}$ simultaneously into the new strategy $S_{a,t+1}$. This process of collectively changing the strategies constitutes an evolutionary process that is described hereafter. During the initialization phase of the simulation (50 cycles) all strategies are fixed. During the main experimental phase (500 cycles) all agents are free to change their strategy.

System Dynamics. The level of technology and therefore the productivity of each capital sector grows with the cumulative investment of all agents in that sector. The speed with which the productivity improves with investment is different for each sector, rendering some sectors economically more attractive than others. We distinguish between viable technologies where investment in the corresponding capital sector will eventually pay off, and nonviable technologies where investment will never pay off. The economic model also has one capital sector that causes economic damage through climatic change. Investment in that sector should be minimized.

The result of these complex non-linear dynamics is that the investment strategy that maximizes the income of an agent changes during the run of a simulation. The only information the agents have on the current relation between income and strategies is the amount of income that each agent achieves with its current strategy. Agents can compare the welfare (fitness) of their peers and choose the agent with the highest welfare to imitate (i.e., recombine with) a part or all of that agent's strategy. An agent can also create a new strategy by innovation (i.e., mutation), which it does by applying some random change to its current strategy.

Social Peer Network. The agents of our application exchange information through a social peer network. The importance of such peer networks for economic simulations is discussed in [18]. According to Tomasini, an evolutionary algorithm with spatial structure is of advantage when dealing with dynamics problems [19]. Lieberman ea. have shown that spatial structures like scale free networks are a potent selection amplifier for mildly advantegous mutants [13]. We use bidirectional peer networks that are generated by a stochastic growth process prior to each run of the simulation. In accordance with the current theory on social networks we ensure that these networks have random connectivity [8, 9], a high cluster coefficient [21], and a scale free degree distribution [2] (actually a gamma distribution with an exponent between 2 and 3). The average connectivity k is a free parameter of the evolutionary algorithm and varies between 2 and 30.

Fitness and Performance Measures. We measure the welfare or fitness $W_{a,t}$ of each agent by how much the agents invests in consumption. The aim of the CRE method is to find a calibration $\mathcal{C}$ of the evolutionary algorithm that allows the agents to achieve a high welfare or fitness in a broad set of simulations. To this end we need a performance measure $\mathcal{F}(\mathcal{C})$ that relates a calibration to the aggregate welfare (i.e., welfare at the population level) that the agents achieve with this calibration. A variety of measures for aggregate welfare are used in economics. They differ in how to correct for risk, social equality, stability of growth, and the discount rate to compare welfare over time. We have found that the results obtained by the CRE method are quite similar for all aggregate welfare measures tried so far and we limit us here to only one measure for $\mathcal{F}(\mathcal{C})$: the discounted mean log welfare

$$\mathcal{F}(\mathcal{C}) = \sum_t \delta^t \sum_{a \in A} \frac{\log W_{a,t}}{|A|}$$

where $|A|$ is the size of the population and where the discount rate is $\delta = 0.97$.

3. THE EVOLUTIONARY ALGORITHM

Representation, Individuals and Population. It is important to note that in this EA agents and strategies are not the same: an agent carries or maintains a strategy, but it can change its strategy and we still consider it as the same agent. This dichotomy is necessary so that we can maintain a social network among the *agents*, while evolving, i.e., changing, the strategies. Because every agent has exactly one strategy at a time, the number of active strategies is the same as the number of agents. The population size (in the search space of strategies) is thus constant and equals the number of agents.

We represent an economic investment strategy $S_{a,t}$ as an n-dimensional investment vector that specifies the fractions $I_1, \ldots, I_n$ of income that agent a invests at cycle t in consumption and the $n-1$ available capital sectors. Formally, a candidate solution or individual strategy is defined as:

$$S_a = \langle I_1, \ldots, I_n \rangle, \quad \sum_{i=1}^{n} I_i = 1$$

with each $I_i \in [0, 1]$.

Fitness and Selection. An agent is evaluated according to the strategy it carries. It is essential to our model that the fitness of a strategy is determined by the welfare of its hosting agent *relative to that of its peers in the social network*. This implies that the structure of the social network has a direct effect on the definition of fitness, hence on the sType name of new folderelection mechanism. The first step in determining the selection probabilities is to rank all agents and their peers according to their respective welfare or fitness. The normalized rank $r_a(b) \in (0\,1]$ is the position of agent b among the peers of a (including a), divided by the size of this group.

$$r_a(b) = \frac{|\{c \,|\, c \in N_a \cup a,\ W_c \leq W_b\}|}{|N_a| + 1}$$

where N_a are the peers of agent a. In case of equal welfare or fitness, i.e., if $W_a = W_b$, agents have the same rank. Note that the best agent of a group always has rank 1 while the worst one has rank $\frac{1}{|N_a|+1}$ which is close to zero.

We introduce two probabilistic selection mechanisms: one to decide whether a given strategy will be changed by random innovation and one to decide whether it will be changed by selective imitation from a peer in the social network. In terms of traditional evolutionary computing (EC) [7], innovation and imitation correspond to mutation and recombination. However, there is an important difference between imitation as used here and usual recombination in EC. In EC the two recombinants have a symmetrical role: they both receive (genetic) information from each other and incorporate it into the offspring. In our imitation mechanism the roles are asymmetrical. One agent imitates the other by receiving its strategy and recombining it with its own. The imitating agent changes, while the agent that is imitated does not.

Reflecting our best knowledge and intuition on social systems, we assume that these selection mechanisms should work differently for agents that have a high fitness (i.e., a high welfare relative to their respective peers) and for agents that have a low fitness (i.e., a low welfare relative to their respective peers). We therefore create two sets of parameters for each selection mechanism: two for agents with a high fitness (specified by a subscript $_r$ for rich), and two for agents with a low fitness (specified by a subscript $_p$ for poor). We also introduce two threshold parameters: one parameter m_f to specify which fraction of the agents define themselves as fit with regard to innovation (mutation), and one parameter m_g to specify which fraction of the agents define themselves as fit with regard to imitation (recombination).

Innovation. Innovation in our simulation is implemented by Gaussian mutation. That is, an agent innovates by changing its strategy vector by the addition of random noise drawn from a Gaussian distribution with zero mean. This implies that small mutations are more likely than large ones. The parameters f_p and f_r (for poor and rich agents respectively) specify the probability that an agent will apply a random change to its strategy at each cycle of the simulation,

$$P[a \text{ innovates}] = \begin{cases} f_p & \text{if } r_a(a) \leq m_f \\ f_r & \text{if } r_a(a) > m_f \end{cases}.$$

The parameters σ_p and σ_r specify the standard deviation of the random change that is applied to a strategy when an agent innovates. The exact formula for changing the strategy vector S_t into S'_{t+1} is

$$S'_{a,t+1} = \begin{cases} S_{a,t} + \mathcal{N}(0, \sigma_p) & \text{if } r_a(a) \leq m_f \\ S_{a,t} + \mathcal{N}(0, \sigma_r) & \text{if } r_a(a) > m_f \end{cases}$$

where $\mathcal{N}(0, \sigma)$ denotes a normally distributed random vector with zero mean and standard deviation σ. In order to avoid negative investments we add the additional constraint that $S_{a,t} + \mathcal{N}(0, \sigma)$ is non-negative.

Imitation. Imitation is performed by combining two strategies through linear combination. The resulting vector replaces the strategy of the imitating agent, the strategy of the imitated agent remains the same. The parameters g_p and g_r specify the probability that an agent will imitate at each cycle of the simulation

$$P[a \text{ imitates}] = \begin{cases} g_p & \text{if } r_a(a) \leq m_g \\ g_r & \text{if } r_a(a) > m_g \end{cases}.$$

If agent a imitates, it needs to choose another agent b to imitate. The parameters h_r and h_p specify the fractions of rich peers from which the agent chooses a random peer to imitate. That is, a poor agent ($r_a(a) \leq m_g$) chooses an agent to imitate according to

$$P[a \text{ imitates } b] = \begin{cases} 0 & \text{if } r_a(b) \leq h_p \\ \frac{1}{\lceil (1-h_p)|N_a| \rceil} & \text{if } r_a(b) > h_p \end{cases}.$$

and a rich agent ($r_a(a) > m_g$) chooses an agent to imitate according to

$$P[a \text{ imitates } b] = \begin{cases} 0 & \text{if } r_a(b) \leq h_r \\ \frac{1}{\lceil (1-h_r)|N_a| \rceil} & \text{if } r_a(b) > h_r \end{cases}$$

If a imitates b, then the strategy $S_{a,t}$ is linearly combined with $S_{b,t}$ into $S'_{a,t+1}$, according to

$$S'_{a,t+1} = \begin{cases} (1 - w_p)\, S_{a,t} + w_p\, S_{b,t}, & \text{if } r_a(a) \leq m_g \\ (1 - w_r)\, S_{a,t} + w_r\, S_{b,t}, & \text{if } r_a(a) > m_g \end{cases}$$

where w is the weight that is given to the imitated strategy.

Table 1: The 13 initial parameters of the evolutionary algorithm. All parameters have the initial range 0–1 except for connectivity which has 2–30. Column 3 and 4 are explained in Section 5.

	parameter θ	mean $\mathcal{K}(\mathcal{C}^\theta)$	std. deviation
k	average connectivity	0.1	0.2
m_f	threshold rank for innovation	0.3	0.4
f_p	P[poor agent innovates]	**1.0**	0.7
f_r	P[rich agent innovates]	**3.9**	1.4
σ_p	innovation variance of poor agent	**1.8**	1.4
σ_r	innovation variance of rich agent	**2.2**	1.4
m_g	threshold rank for imitation	0.1	0.1
g_p	P[poor agent imitates]	0.5	0.3
g_r	P[rich agent imitates]	0.3	0.3
w_p	imitation weight of poor agent	0.3	0.3
w_r	imitation weight of rich agent	0.2	0.2
h_p	imitated neighbors of poor agent	0.2	0.2
h_r	imitated neighbors of rich agent	0.6	0.7

Since the fractions of income have to sum up to one, as the last step in performing imitation, the strategy is normalized:

$$S_{a,t+1} = \frac{S'_{a,t+1}}{|S'_{a,t+1}|}$$

Full Evolutionary Model. The resulting 13 parameters are shown in the first two columns of Table 1. On top of the "traditional" task of finding good values for these parameters we want to know if they are 1) indeed relevant for the evolutionary algorithm and 2) sufficient to calibrate the system. We call a parameter relevant if with some values for the parameter the evolutionary algorithm consistently reaches a higher performance, in this case a higher aggregate welfare for the agents. Calibration of a relevant parameter therefore significantly improves the performance of the evolutionary algorithm. An irrelevant parameter on the other hand does not influence the performance of the evolutionary algorithm, no matter what value it takes, and it should be removed for the sake of analytic clarity and computational stability. If a parameter proves to influence the performance, but the parameter values for which the performance is highest are highly sensitive to changes in the specification of the simulation, the parametrization is not sufficient. In this case more parameters need to be added to the respective component of the evolutionary algorithm in order to capture all the necessary information.

4. THE CRE METHOD

As discussed in the introduction, the main objective of the parameter calibration and relevance estimation (CRE) method is to find a set of relevant parameters, to find good values for these parameters, and to establish how accurate these values have to be. One could say that we are after a good evolutionary system that allows the agents to achieve a high performance not only in one particular simulated environment, but in as many simulated environments as possible. Since we assume that this depends largely on the algorithmic complexity of the model, we define "good" as having a good tradeoff between performance and algorithmic complexity. Thus, the quality of any given list of parameters and a specific vector of values x for these parameters is determined by the performance $\mathcal{F}(x)$ obtained by running the simulation using x, and the information or algorithmic complexity $\mathcal{K}(x)$ needed to specify x. We start the formalization of this matter by defining the term *calibration* as a distribution over parameter vectors.

Calibration. Let $M = \langle \theta_1, \ldots, \theta_k \rangle$ be a list of k parameters with an initial finite[1] domain of possible values for each parameter θ_i and let $\mathcal{X}_M$ stand for the Cartesian product of these domains. Then a *calibration* $\mathcal{C}$ is a distribution $\mathcal{C}(x)$ over all possible parameter settings $x \in \mathcal{X}_M$. We define $\mathcal{C}_0$ to be the uniform distribution over $\mathcal{X}_M$ and $\mathcal{C}^\theta$ to be the marginal distribution of $\mathcal{C}(x)$ on parameter θ.

Calibration Performance. We can now define *calibration performance* $\mathcal{F}(\mathcal{C})$ as the expected performance when drawing the parameter settings from the calibration distribution $\mathcal{C}$

$$\mathcal{F}(\mathcal{C}) = \mathop{\mathrm{E}}_{x \in \mathcal{X}_M} [\mathcal{C}(x)\mathcal{F}(x)] .$$

Calibration Complexity. Algorithmic complexity as understood by A.N. Kolmogorov [12] is the minimum amount of information needed to compute a vector of values x. While algorithmic complexity itself is not computable, it is sufficient for our purpose to estimate it from the Shannon entropy[2] $H(\mathcal{C})$ of the distribution $\mathcal{C}(x)$. The probability that the true algorithmic complexity $\mathcal{K}(x)$ of any x chosen by the distribution $\mathcal{C}(x)$ is significantly lower than the Shannon entropy of $\mathcal{C}(x)$ is exponentially low [11] and can be neglected. We define *calibration complexity* as

$$\mathcal{K}(\mathcal{C}) = H_0 - H(\mathcal{C}).$$

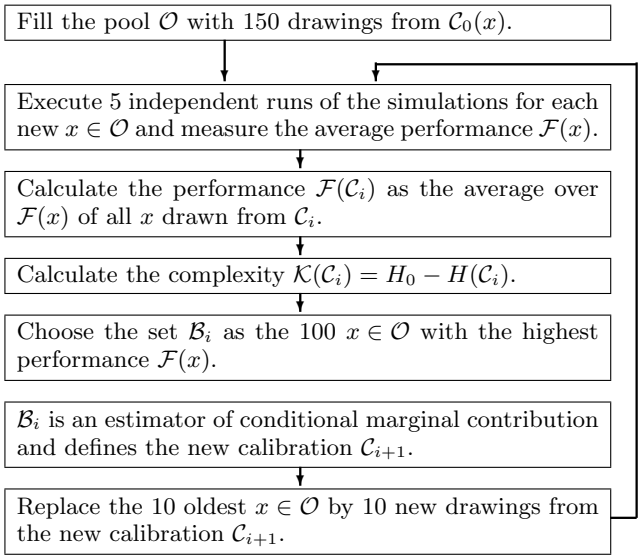
where H_0 is the average entropy of calibrations that the CRE method produces when fed with white noise. This will be explained shortly. Calibration complexity estimates the amount of additional information needed to calibrate a set of parameters, given that we have already specified a finite parameter space $\mathcal{X}_M$. We use binary bits as the unit of Shannon entropy and calibration complexity.

Minimax Calibration. Not all possible calibrations are equally interesting to us. Of all calibrations that achieve a given performance we are only interested in those of the lowest complexity. And of all calibrations of a given complexity we are only interested in those that achieve the highest performance. While we assume that a calibration that satisfies both constraints generally does not exist, it is essential that the CRE method comes reasonably close to producing minimax calibrations because these are the only calibrations of any practical interest. We loosely define an approximate minimax calibration as a calibration $\mathcal{C}_i$ that has the following properties: no calibration $\mathcal{C}_j$ with performance $\mathcal{F}(\mathcal{C}_j) = \mathcal{F}(\mathcal{C}_i)$ is *significantly* less complex (i.e., no $\mathcal{K}(\mathcal{C}_j) \ll \mathcal{K}(\mathcal{C}_i)$) and no calibration $\mathcal{C}_k$ with complexity $\mathcal{K}(\mathcal{C}_k) = \mathcal{K}_(\mathcal{C}_i)$ can perform *significantly* better (i.e., no $\mathcal{F}(\mathcal{C}_k) \gg \mathcal{F}(\mathcal{C}_i)$).

[1] If the initial domain is $\mathbb{R}$ we can use a transformation like the sigmoid to make it finite.

[2] Shannon entropy is defined as $H(\mathcal{C}) = -\sum_x \mathcal{C}(x) \log \mathcal{C}(x)$.

Figure 1: Diagram of the CRE method



Conditional Marginal Contribution. At the heart of the CRE method lies a reliable estimation of conditional marginal contribution of parameter values to performance. By this we mean the contribution of the values of a specific parameter to performance (hence *marginal*) if the distribution of values over the other parameters are fixed by a specific calibration (hence *conditional*).

The Search Algorithm. We use an iterative algorithm that produces a series of approximate minimax calibrations of increasing complexity and performance. As shown in Figure 1, we work with a pool $\mathcal{O}$ of 150 different parameter settings x. These are initially drawn from $\mathcal{C}_0$, the uniform distribution over the finite parameter space $\mathcal{X}$. At each iteration i we replace the oldest 10 x from $\mathcal{O}$ by 10 new x drawn from the latest calibration $\mathcal{C}_i$, calculated below. We run the simulation 5 times for each new setting and measure the average performance $\mathcal{F}(x)$ from the aggregate welfare measure discussed in Section 2 (we independently verified that 5 runs produce a reliable estimator of the average performance). We measure the performance $\mathcal{F}(\mathcal{C}_i)$ as the average over $\mathcal{F}(x)$ of all x that were drawn from $\mathcal{C}_i$. Finally we rank all $x \in \mathcal{O}$ according to the individual performance $\mathcal{F}(x)$ and select the best 100 settings $\mathcal{B}_i$.

The density of $\mathcal{B}_i$ is a good estimator of the conditional marginal contribution of the parameters to performance: the higher the density over a certain range of a parameter, the higher the marginal contribution of values from that range to performance, conditioned on the fact that the values for the other parameters were chosen from the current and previous calibrations. We consider the Shannon entropy of this density to be an excellent estimator of parameter relevance. Figure 2 shows how the density of $\mathcal{C}_i$ over two parameters changes during a search. The Shannon entropy of the density of $\mathcal{B}_i$ decreases much faster for the innovation variance σ_r than for the imitation weight w_r.

We use a two step process to generate the next distribution $\mathcal{C}_{i+1}$ from the the density of $\mathcal{B}_i$. When drawing a new x from $\mathcal{C}_{i+1}$, we first draw a random member $y \in \mathcal{B}_i$ for each parameter $\theta \in M$. When drawing this y it is absolutely

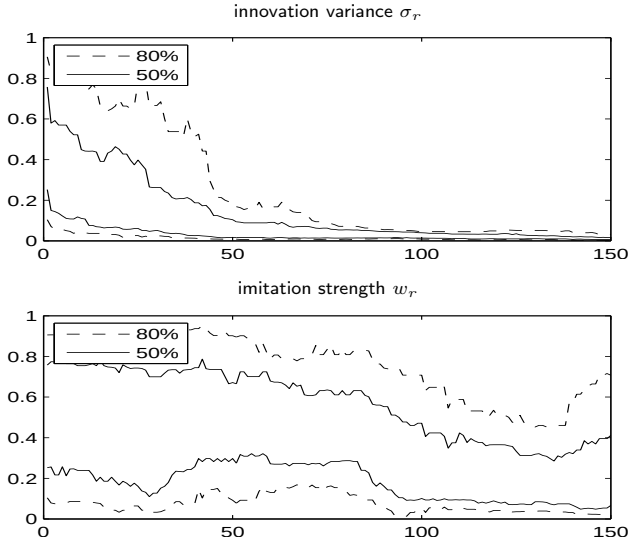


Figure 2: Example of how the density of $\mathcal{B}_i$ changes during the search. The x-axis shows the iterations i of the search, the y-axis shows the parameter ranges. Each point i along the x-axis is a crosscut through the corresponding calibration $\mathcal{C}_i$, reflecting the density of the 100 best settings $\mathcal{B}_i$ along the parameter. 50% of $\mathcal{B}_i$ are concentrated between the two solid lines, 80% between the dashed lines. The graphs are taken from an experiment with 200 agents, 2 nonviable technologies and low vulnerability.

essential that we give a slightly higher probability to those members $\in \mathcal{B}_i$ that have the lowest density. This minimizes the Shannon entropy of $\mathcal{C}$ and ensures that the method approximates the minimax calibration. Next we determine the parameter range that is formed by the two members $\in \mathcal{B}_i$ that are the closest upper and lower neighbors of y along the parameter. We now determine the value of the parameter for the new x by drawing a random value from this range.

To give an example, let us draw a new value for f, the chance to innovate. We first draw a random $y \in \mathcal{B}_i$, say with value $y^f = 0.959$. Assume the closest neighbors $\in \mathcal{B}_i$ with regard to the innovation rate have values 0.955 and 0.961. The final parameter value x^f for the innovation rate will now be drawn from the uniform distribution [0.955–0.961].

Calculating Complexity. When measuring the calibration complexity, we need to correct for the effects of noise. Due to the random nature of the CRE method, the Shannon entropy of the density of $\mathcal{B}_i$ is always lower than the Shannon entropy of the uniform distribution, even when the contribution to performance is uniform over all parameters. We can measure this trivial decrease in Shannon entropy by replacing the performance measurements from actual simulations by white noise and we found that for every parameter θ the resulting Shannon entropy H_0^θ is typically 0.9 bit lower than the Shannon entropy of the uniform distribution over θ. Since we are only interested to know how much the Shannon entropy of $\mathcal{C}_i$ differs from this trivial entropy level, we calculate the calibration complexity of each parameter θ as $\mathcal{K}(\mathcal{C}_i^\theta) = H_0^\theta - H(\mathcal{C}_i^\theta)$. And since $\mathcal{C}_i$ is constructed form the marginal distributions $\mathcal{C}_i^\theta$, which therefore contain all the necessary information, we have $\mathcal{K}(\mathcal{C}_i) = \sum_\theta \mathcal{K}(\mathcal{C}_i^\theta)$.

Interpretation of the Results. As the search progresses, the CRE method produces approximate minimax calibrations of increasing performance and complexity. These can be plotted in a convenient way so that we can use visual inspection to select calibrations with a good performance/complexity tradeoff. As can be seen in Figure 3, $\mathcal{F}(\mathcal{C})$ usually stops to significantly increase much earlier than $\mathcal{K}(\mathcal{C})$. Once the incease in performance has come to an obvious halt the CRE method can be safely terminated. The performance/complexity tradeoff has passed its maximum and will only decline.

We can use the minimum complexity needed for any given performance to compare the calibration of different sets of parameters. If two sets of parameters can achieve equivalent performance, we can select the set that achieves this performance with less complexity as the one that will likely be of a more general value. We are also interested in $\mathcal{K}(\mathcal{C}^\theta)$, the calibration complexity per parameter θ. The higher this complexity, the more information is needed to calibrate the parameter. When applying the EA to a new problem, such a parameter is more likely to be miss-calibrated than a low complexity parameter and consequently deserves more attention from the practitioner. A low complexity on the other hand (say $\mathcal{K}(\mathcal{C}^\theta) < 1$ bit) is an indication that a parameter is irrelevant and could be removed from the evolutionary model.

The 25^{th} and 75^{th} percentile of the calibration distribution on each parameter θ are what we consider the practical calibration of the CRE method: the values from this range have the highest conditional marginal contribution. As long as the parameters of our EA stay within this range we are confident of a high performance.

5. EXPERIMENTS

The aim of the CRE method is to find a calibration of the evolutionary algorithm that allows the evolutionary algorithm to achieve a high level of performance in a broad set of different problem specifications. In the present application we want the economic agents to achieve a high welfare or fitness in a broad set of simulated economic environments. In order to demonstrate the CRE method, we add three scaling parameters to the agent-based application described in Section 2. These parameters allow us to define a total of 18 simulated economic environments. They are:

- the number of agents (200 or 2000).
- the number of capital sectors with nonviable alternative technologies (2, 20, or 200). In all cases the simulation has exactly one capital sector with a viable technology and the number of nonviable technologies controls the difficulty of finding this viable technology.
- the vulnerability ("low", "moderate", or "high") of the agent economies to climatic change. Exactly one capital sector leads to climatic change and the agents have to avoid investing in this sector.

Starting with the initial parameter set of 13 parameters described in Section 3 we search for calibrations that have a good performance/complexity tradeoff. We do so for each of the 18 environments. We then average the calibration complexity of each parameter over the good calibrations of all environments. We used this average complexity to decide if a parameter is relevant or not.

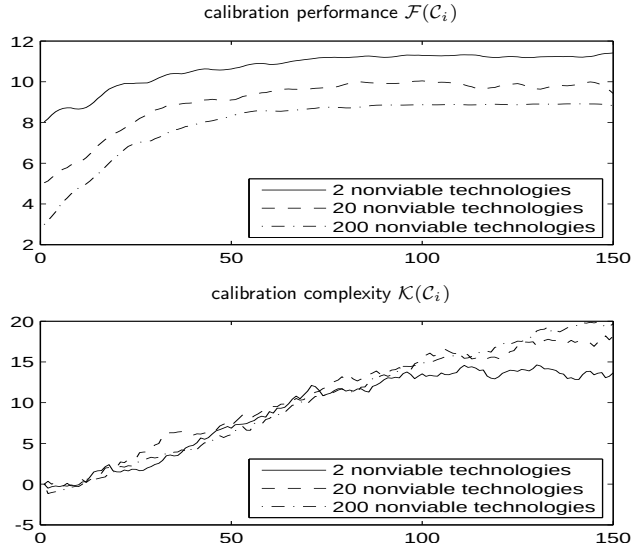


Figure 3: Illustration of the change in calibration performance and calibration complexity for the initial evolutionary model of 13 parameters. The x-axis shows the first 150 iterations i of a search, the y-axes show calibration performance $\mathcal{F}(\mathcal{C}_i)$ and calibration complexity $\mathcal{K}(\mathcal{C}_i)$. The graphs are from 3 experiments with 200 agents, low vulnerability and 2, 20, and 200 nonviable technologies.

As the CRE method searches for a good calibration, both the performance and the complexity of the investigated calibrations increase almost monotonously. Figure 3 illustrates this with graphs of $\mathcal{F}(\mathcal{C}$ and $\mathcal{K}(\mathcal{C})$ from three different environments. The increase in calibration performances is greatest at the beginning of a search, slows down soon, and comes to a halt somewhere between iteration 60 and iteration 90. Calibration complexity on the other hand increases linearly until about iteration 100. In most simulated environments it continues to increase even after that. Visual inspection of the results of all experiments leads us to conclude that the trade off between performance and complexity is best between iteration 75 and 85 of the CRE method.

Table 1 shows the average complexity per parameter in bits, together with the standard deviation. The results are averaged over iteration 75–85 of the CRE method of all 18 simulated environments. Only 4 parameters show a significant complexity of 1 bit or more, which means that the performance of the evolutionary algorithm depends heavily on the calibration of these parameters. On the other hand, calibration of the other parameters seems largely irrelevant to the performance of the evolutionary algorithm and their number should be reduced as much as possible. The 4 relevant parameters define the probabilities to innovate (f_p, f_r) for poor and rich agents and the innovation variance (σ_p, σ_r) for poor and rich agents. The CRE method calibrates these pairs of parameters to similar values (not shown in the table) and we concluded that they can be combined into one parameter each. These results thoroughly falsify our original hypothesis that agent behavior should depend on relative welfare and that it needs to be calibrated by different sets of parameters.

Table 2: The 6 parameters of the simplified evolutionary model, averaged over iteration 75–85. Initial parameter ranges are 0–1, except for connectivity, which has 2–30.

	parameter θ	mean $\mathcal{K}(\mathcal{C}^\theta)$	std. deviation	$25^{th}+75^{th}$ percentiles
k	average connectivity	0.1	0.2	5.8–18.2
f	$P[a$ innovates]	**3.2**	1.2	**0.01–0.07**
σ	innovation variance	**3.3**	1.3	**0.02–0.07**
g	$P[a$ imitates]	0.5	0.4	0.54–0.88
w	imitation weight	0.3	0.3	0.41–0.88
h	threshold rank imitated	**1.0**	0.6	**0.69–0.93**

A Simplified Evolutionary Model. To verify these conclusions we simplify the evolutionary model by using exactly one parameter for each component shown in Table 2: connectivity k, probability to innovate f, innovation variance σ, probability to imitate g, imitation weight w and the threshold rank h that distinguishes between agents that can be imitated and that cannot be imitated. Applying the simplified evolutionary model to all 18 simulated economic environments reveals that with equal calibration complexity, the simplified model always achieved a higher performance, typically 10%, see Figure 4. (We also tried evolutionary models with intermediate numbers of parameters but the improvements were less significant.)

After concluding that the simplified set of evolutionary parameters has a far better performance/complexity payoff, we still need to choose a calibration for it. Figure 5 shows how the CRE method calibrates the parameters of the simplified model. When comparing the graphs of welfare and complexity visually we find that the best payoff between welfare and complexity is again achieved in the neighborhood of iteration 80 of the search. Table 2 shows the average calibration complexity $\mathcal{K}(\mathcal{C}^\theta)$ and its standard deviation and the 25^{th} and 75^{th} percentile of $\mathcal{C}^\theta(x)$ for each parameter of the simplified model, averaged over calibration 75-85 of the CRE method and all tested environments.

As with the initial set of parameters, innovation proves to be the most sensitive part of the simplified evolutionary model. Both the probability to innovate and the innovation variance have to be well calibrated in order to allow the agents to adapt to their environment. The calibration of h (the fraction of richer neighbors to be imitated) also shows significant impact on aggregate welfare. The imitation parameters and the average number of neighbors prove almost irrelevant.

6. CONCLUSIONS

The CRE method provides us with useful measures of calibration complexity, both numerically in the form of Shannon entropy and visually in the form of percentiles. Visual inspection and manual addition or removal of parameters proves sufficient to select and calibrate the relevant parameters of an evolutionary system. When comparing the initial and the final simplified sets of parameters we find that with equal calibration complexity the final set of parameters consistently achieves a significantly higher welfare for the agents.

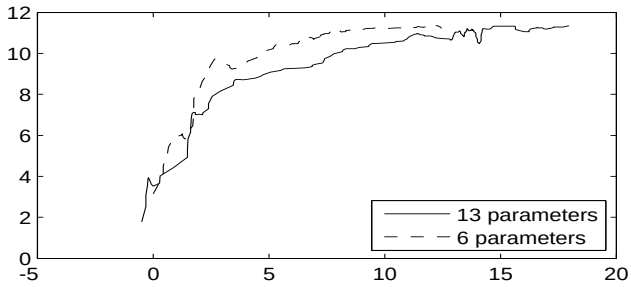


Figure 4: Welfare as a function of calibration complexity. The x-axis shows calibration complexity as it increases during the search. The y-axis shows aggregate welfare. The right cut off of each graph marks the complexity the calibration has reached after 250 iterations of the CRE method, not an absolute maximum. The graphs are taken from an experiment with 200 agents, 2 nonviable alternative technologies and low vulnerability to climatic change.

Generality of Results. The calibrations produce stable and consistent results. Differences in calibration between environments are small. For example, an increase in the number of nonviable technologies leads to lower values for the innovation parameters, apparently because random innovation becomes riskier. However, such dependencies show consistently only in a later stage of the CRE method, typically beyond iteration 100. The results of iteration 75–85, which have the best performance/complexity tradeoff, are similar for all tested environments. At least for the simulated environments discussed here, the CRE method maximizes the performance of the evolutionary algorithm without compromising general validity.

As mentioned in Section 2, we also tested the method with a number of different aggregate welfare measure, and again we found that results were very stable and consistent. If a calibration led to high welfare according to one measure, it did so according to most measures.

Minimax Calibration. We verified in independent experiments (not reported here) that the calibrations produced by the CRE method were good approximations of the maximum/minimum calibration: we could not get significantly better results by relaxing, within feasible bounds, the key parameters of the method like the pool size, the number of runs per setting or the extra probability that is given to low probability ranges. But only an extensive analysis of different search spaces can asses how close the method approximates the best possible minimax calibrations.

Application Specific Results. We conclude that most parameters of our original evolutionary model are not relevant. In particular, there is no evidence that agents should condition their evolutionary behavior on their own relative welfare. The innovation parameters are the most relevant parameters in all versions of the evolutionary model and their calibration has the biggest influence on agent welfare. Calibrating the fraction of agents to imitate is also important. The details of the social network, in particular the average connectivity, seem to be irrelevant, but warrant further research.

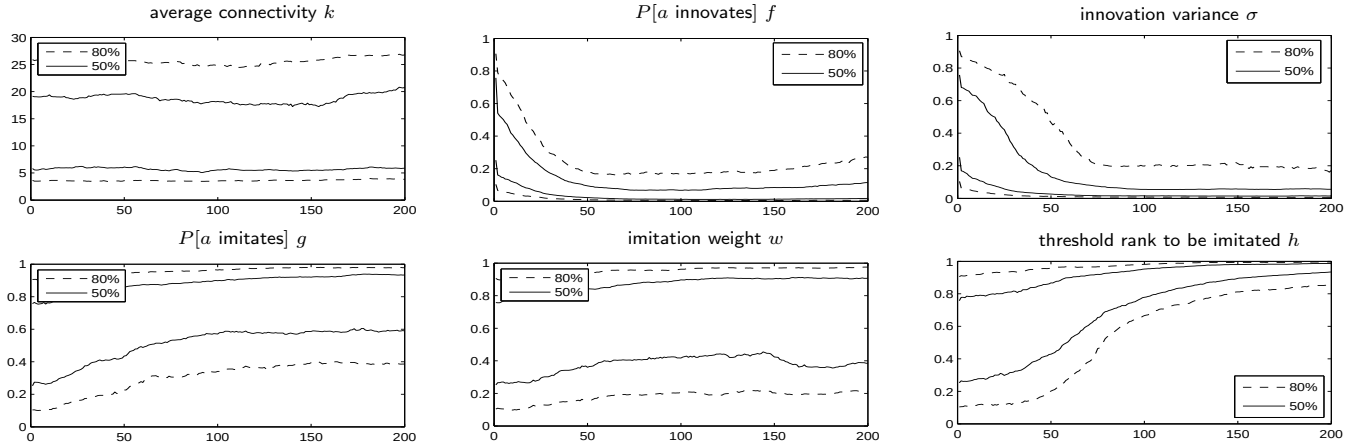


Figure 5: Calibration of the 6 parameters of the simplified model, averaged over the 18 simulated environments. The x-axis shows the iterations of the CRE method, the y-axis the parameter range. 50% of $\mathcal{B}_i$ are located between the two solid lines, 80% of $\mathcal{B}_i$ between the dashed lines. At iteration 80 (where performance has stopped to increase) the three parameters f, σ and h show a significant reduction in Shannon entropy.

Simulation Details

The experiments are written and analyzed in Matlab. Evaluating a single simulation takes between 1 and 10 seconds on a 64 bit computer with a 2GHz processor. Searching through 200 iterations of the CRE method amounts to about 10.000 simulations and takes between 3 and 30 hours. Proper optimization of the method parameters can reduce the time complexity of the method by a factor of 10.

Acknowledgment

We thank J.C.J.M. van den Bergh, M. Horváth, M. Schut and G. Daniel for support and helpful discussions.

7. REFERENCES

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *2nd Int. Symp. Inf. Theory, ISIT*, pages 267–281, Tsahkadsor, Armenian SSR, 1973.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[3] A. R. Barron, J. J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. Inform. Theory*, 44(6):2743–2760, 1998.

[4] G. J. Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13(4):547–569, 1966.

[5] G. J. Chaitin. On the length of programs for computing finite binary sequences: statistical considerations. *J. ACM*, 16(1):145–159, 1969.

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

[7] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin / Heidelberg, 2003.

[8] P. Erdős and A. Renyi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[9] P. Erdős and A. Renyi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 5:17–61, 1960.

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

[11] P. D. Grünwald and P. M. B. Vitányi. Kolmogorov complexity and information theory. *J. Log. Lang. Inf.*, 12(4):497–529, 2003.

[12] A. N. Kolmogorov. Three approaches to the quantitative definition of information*. *Int. J. Comput. Math.*, 2(1-4):157–168, 1968.

[13] E. Lieberman, C. Hauert, and M. A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316, 2005.

[14] V. Nannen. *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*. PhD thesis, Artificial Intelligence, Vrije Universiteit, Amsterdam, Amsterdam, 2009.

[15] J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

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

[17] R. J. Solomonoff. A formal theory of inductive inference. Part I and II. *Inform. Control*, 7(1):1–22–224–254, 1964.

[18] L. Tesfatsion. Agent-based computational economics: A constructive approach to economic theory. In K. L. Judd, editor, *Handbook of Computational Economics*, pages 831–880. Elsevier, 2006.

[19] M. Tomasini. *Spatially Structured Evolutionary Algorithms*. Springer, Berlin / Heidelberg, 2005.

[20] N. K. Vereshchagin and P. M. B. Vitányi. Kolmogorov's Structure Functions and Model Selection. *IEEE Trans. Inform. Theory*, 50(12):3265–3290, 2004.

[21] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.