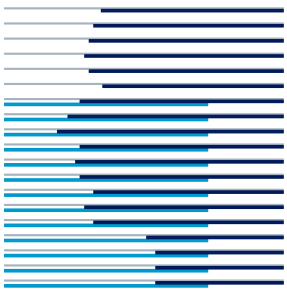
Explicit Local Models: Towards "Optimal" Optimization Algorithms

Jan Poland
IDSIA, Galleria 2, CH-6928 Manno
Switzerland
jan@idsia.ch



Technical Report No. IDSIA-09-04
June 16, 2004

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

IDSIA is a joint institute of both University of Lugano (USI) and University of Applied Sciences of Southern Switzerland (SUPSI), and was founded in 1988 by the Dalle Molle Foundation which promoted quality of life.

Explicit Local Models: Towards "Optimal" Optimization Algorithms

Jan Poland
IDSIA, Galleria 2, CH-6928 Manno
Switzerland*
jan@idsia.ch

Abstract

We address the problem of minimizing functions on a continuous vector space with the help of models. A theoretical background is established in terms of the AIXI theory, which concerns optimal rational agents in unknown environments. It implies recommendations for the design of optimization algorithms. In the light of this theory, existing model-based algorithms are reviewed. In the second part of the paper, an optimization algorithm using local quadratic models is stated and evaluated.

1 Introduction

In the recent past, a considerable amount of research has been devoted to model-based optimization algorithms, i.e. algorithms that use techniques from Machine Learning. Despite the efforts and considerable practical successes, there is no general theory which is both mathematically satisfying and practically relevant. (There are theories for particular methods, such as Quasi-Newton algorithms which may be termed model-based.) We won't close this gap completely: We will present Hutter's AIXI theory which is general and mathematically neat, but not of direct practical relevance. Yet this theory yields practical recommendations for the design of model-based optimization algorithms.

In principle, any optimization algorithm might be termed "model-based" as soon as it makes use of the information gained by evaluating the objective function. This rules out only trivial methods such as random sampling and complete enumeration. The internal representation of this information is then (maybe a very crude form of) a model. However, it is common to call only those algorithms model-based which incorporate fairly advanced Machine Learning methods. Any use of information aims at reducing the number of function evaluations needed. Looking closer, we can distinguish two different reasons for models in optimization algorithms:

1. **Models as surrogates for expensive functions.** If the evaluation of the objective function is expensive in some respect, e.g. time-consuming, then an approximation which is constructed from previous function evaluations is used. This procedure can be

*This work was supported by SNF grant 2100-67712.02.

integrated in any optimization algorithm. It has been largely applied in the evolutionary optimization of functions on $\mathbb{R}^d$.

2. **Models for facilitating the optimization.** This concerns the central issue of optimization, since it includes all techniques rendering the optimization more efficient than random sampling. In particular, cleverly chosen models may represent just the features of the objective function which are essential for optimization, neglecting all others. Examples are Quasi-Newton methods for functions on $\mathbb{R}^d$, where a quadratic model is maintained, or the Bayesian Optimization Algorithm (BOA) [PGCP99] on bit strings, which approximates the distribution of good solutions using a Bayesian network. Also noise filtering methods fall in this category.

It is desirable for an algorithm to cover *both* aspects, i.e. to incorporate models that facilitate the optimization and simultaneously work with a minimum amount of data. In particular, the models should be chosen in order to cope with *noisy* function evaluations. It is our aim to give guidelines for the design of such algorithms, for continuous real-valued functions on $\mathbb{R}^d$. As the dimension d becomes fairly large (already for $d = 10$), the exponentially growing search space renders a *global* optimization generally infeasible. It is therefore common to concentrate on *local* optimization, silently agreeing on the assumption that a good local optimum is sufficient.

This paper is structured as follows. In Section 2, the AIXI theory, which is a general theory of optimal rational behavior in unknown environments, is briefly introduced and applied to model-based optimization. This theory is appropriate for describing the optimization based on any kind of model, in particular explicit models and estimation-of-distribution. In Section 3, we review some existing model-assisted approaches. Section 4 proposes and evaluates an optimization algorithm using local quadratic models. The final discussion and presentation of open problems is contained in Section 5. We set our focus on *continuous domain*, i.e. minimizing continuous functions on $\mathbb{R}^d$.

2 AIXI Theory and Optimization

Answering the question of how a (hypothetical) optimal agent would behave in an unknown environment, Hutter [Hut00, Hut01b] proposed the AIXI theory. It is strongly motivated by Solomonoff's [Sol78] theory of optimal universal sequence prediction. Assume you are asked to sequentially predict the bits of a stochastic binary sequence which is generated by a computable distribution. Then the Solomonoff predictor has a surprisingly good performance: Its total expected number of prediction errors is bounded by the Kolmogorov complexity of the true distribution (see e.g. [LV97]). Its drawback is that it is not computable. This result is extendible [Hut01a] to arbitrary countable classes of distributions, where each distribution is endowed with a weight and the weights sum up to at most 1. Then the universal predictor is the Bayes mixture over the class given the past observations, and its total expected number of prediction errors is bounded by the negative logarithm of the weight of the true distribution.

AIXI theory applies as soon as the decision-maker, called agent henceforth, no longer only *predicts*, but also *acts* and its actions influence the environment. Let $\mathcal{Y}$ be the set of possible *observations*, $\mathcal{X}$ the set of possible *actions*, and $\mathcal{M}$ be a countable class of *environments*. An environment $\mu \in \mathcal{M}$ specifies distributions on $\mathcal{Y}$, namely for a given sequence of past actions and observations up to time $t-1$ plus a current action (input) at time t:

$$\mu(y \mid x_1 y_1 x_2 y_2 \dots x_{t-1} y_{t-1} x_t)$$

is the probability of observing y, where $t \geq 0$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and $\sum_{\mathcal{Y}} \mu(y|\ldots) = 1$. In the general case, the output $\mathcal{Y}$ usually consists of an informative part and a *reward*. The goal of the agent is maximizing the total future reward up to some time $T > 0$, the *horizon*. However, for optimization we may immediately make some important simplifications[1]: The environment is specified by a distribution depending only on the *current* input, i.e.

$$\mu(y \mid \ldots) = \mu_f(y \mid \ldots) = \mu_f(y \mid x_t)$$

Here, $f : \mathcal{X} \to \mathcal{Y}$ is the objective function, and the μ_f is a distribution since the function evaluations may be noisy (without noise, we just have $\mu_f(y|x) = \delta_{f(x)}(y)$). The output without the reward part is defined as $\mathcal{Y} = \mathbb{R}$. The rewards can be defined in different ways. If the function values are not noisy, we may simply set

$$\text{Reward}(x_T) = -f(x_T) \text{ and } \text{Reward}(x_t) = 0 \text{ if } t < T$$

(recall that we want to minimize f). That is, the last function evaluation determines the quality of the optimization. On the other hand, if the function evaluation is noisy, then an appropriate specification of the reward is not clear. Since we will not try to maximize the reward in the above form, we leave this question unanswered.

Assume for the moment that the objective function f is known and not noisy. Then trivially the optimal action is letting $x_T = \arg\min f$, but of course this does not solve the optimization problem. So we will require that *the class $\mathcal{M}$ contains only models which are* easier *to optimize than f*. Thus using the model actually saves computational resources. This may be given if f is expensive to evaluate, or if the graphs of the models are relatively simple compared to the graph of f, in particular if the models are only defined *locally*. In general, $\mathcal{M}$ will not contain f.

If the true environment μ is not known ("black-box optimization"), then we may estimate it using the Bayes mixture ξ:

$$\xi(y \mid x_1 y_1 \ldots x_{t-1} y_{t-1} x_t) = \frac{\sum_{\mu \in \mathcal{M}} \mu(y|x_t) w(\mu) \prod_{i=1}^{t-1} \mu(y_i|x_i)}{\sum_{\mu \in \mathcal{M}} w(\mu) \prod_{i=1}^{t-1} \mu(y_i|x_i)}. \tag{1}$$

Here, $w(\mu)$ is the prior weight of μ. The posterior weight of μ given past $(x_i y_i)_{i=1}^{t-1}$ is proportional to $w(\mu) \prod \mu(y_i|x_i)$. Observe that ξ does depend on the past, in contrast to each single μ. For example, if $\mathcal{M}$ contains two elements, the constant function 1 (which is the true function) and the constant function 2, then after the first function evaluation ξ always gives the correct output.

If ξ is the estimate for the true environment μ, then a rational agent should choose its action in order to *maximize its total expected future reward*: that is the definition of the AIXI agent. One can write this formally as the evaluation of an expectimax-tree up to level T [Hut00], but for our purpose it is sufficient to grasp the idea. In the following paragraphs, we will obtain some concrete recommendations from the AIXI theory for the design of optimization algorithms.

Multi-step lookahead. The AIXI agent chooses his action considering the complete relevant future. In contrast, the sampling strategy of most common optimization is *greedy*: The point with the expected minimal function value *in the next step* is chosen. However,

[1]The reader familiar with AIXI theory will note that this is the setup of the FMξ agent instead of the general AIXI agent, the latter not even "knowing" that he is doing function minimization.

evaluating the expectimax-tree generally involves a huge (i.e. exponential in T) computational effort, and it is not trivial to approximately evaluate the tree. This has a counterpart in machine learning, namely in active learning: There, the agent tries to sample in order to maximize the information gain by each function evaluation. Although this equally requires the evaluation of a multi-step lookahead tree, asymptotical analysis and empirical evidence suggest that here the greedy strategy is already quite good, [Coh94, Paa97]. Of course, a pure active learning approach maximizing only information gain and ignoring the objective value is unacceptable in general. However, combined with *local models* (see below) we can justify it: Sampling and evaluating serves for improving the local model, while the domain of the model is shifted towards the optimum. Of course, this heuristic procedure does not answer the problem of how to optimally do multi-step lookahead.

Explicit models. Given the AIXI background, it is clearly most simple and therefore preferable to use a set of functions, possibly with noise, as environment class. An example is the class of feed-forward networks with a variance parameter, reflecting the assumption that the noise is Gaussian and uniform over the input space. This results directly in *explicit* models. AIXI can also describe other model types. For example, we may consider the class of Bayesian networks describing distributions on binary strings. This leads to the BOA [PGCP99]. However it is likely that errors are amplified in the two-step process of using the objective function for selecting a distribution and then using the distribution for specifying a model. Moreover, using explicit models it is straightforward to handle noisy function evaluations appropriately. Nevertheless we will see also arguments in favor of estimation-of-distribution algorithms below.

Local optimization. In the passive case of sequence prediction, one usually chooses the class of distributions as large as possible (e.g. all computable distributions), since the true distribution only needs to be in that class in order to prove strong performance guarantees. In contrast no such assertions can be made for the general AIXI setup. More restrictive assumptions like ergodicity [Hut02] do allow for certain assertions, which however do not guarantee a practically relevant efficiency. In fact, consider the following example: $\mathbb{B}^n$ is the set of all bit strings of length n, and $\mathcal{M} = \{-\delta_b : b \in \mathbb{B}^n\}$ contains all "needle in the haystack" functions on $\mathbb{B}^n$ ($\delta_b(x) = 1$ if $x = b$ and $\delta_b(x) = 0$ otherwise). Then each function in $\mathcal{M}$ is described with n bits, but in order to locate the minimum, order of 2^n function evaluations are necessary. Of course, this is a very bad example for optimization, usually the functions behave nicer. However, the so-called "curse of dimension" is also present in general. In order to efficiently optimize functions in higher dimension, one therefore mostly assumes that a *local* optimum is sufficient. In the rest of the paper, we will accept this assumption. (Nevertheless we also discuss some global approaches in Section 3.) This means that we consider a class of "localized" models, which are defined only in (possibly several) parts of the search space. We will moreover restrict to local models in the sense that they are defined only in a *single* neighborhood of a point in the search space.

Unambiguous estimators. We have already mentioned that the evaluation of a Bayesian mixture (1) is problematic in general. The exact expression is usually not tractable. Monte Carlo methods may be used but are computationally expensive. This is the same problem as in machine learning, where most often instead of the integration over the model space just the most probable model (also known as MAP or MDL estimator) is used, or even only an approximation of the MAP estimator. However, the MAP estimator has worse performance than the Bayes mixture in general [PH03]. In optimization, there is a particular caveat: Models may tend to have "false optima" (e.g. [EBNK99]), more generally speaking, even

small model errors may heavily mislead the optimization process. This is particularly true for nonlinear models such as feed-forward networks. Moreover, they are ambiguous in the sense that there are usually several local optima in the model space, which can lead to further instability. If on the other hand a *linearly parameterized* such as polynomial is used, then the MAP estimator is unique and the resulting output depends linearly on the already evaluated function values. Therefore, the estimator is unbiased, provided that the noise is Gaussian. Thus a more robust behavior can be expected. False model optima are probably the main argument against explicit models and in favor of estimation-of-distribution, which is less prone to this problem.

3 Review of Algorithms

We classify some important classes of algorithms in terms of the AIXI framework, concentrating on continuous domain problems. Relevant research has been done for evolutionary algorithms, i.e. population based optimization with mutation and recombination. In continuous domain, these algorithms are termed *Evolution Strategies (ES)*.

Model-assisted Evolution Strategies (ES). Recently, much attention has been directed to the evolutionary optimization of expensive functions using surrogates (often also called metamodels). A surrogate is an explicit model for the true objective function, in the majority of cases a *Gaussian process* (also known as DACE model or Kriging) is used. Mostly, the model domain is not controlled, in this sense the Gaussian process acts as a global model (sometimes also the algorithm starts with a global experimental design). Recent work can be found in [EBK01, JOS02, EAG+02, USZ04]. In contrast, the Gaussian process is used as a local model in [BSK04]. This is in order to reduce the risk of converging to false optima of the model, as discussed in the previous section. Other approaches use different strategies to cope with this problem, e.g. "controlling" individuals or generations [JOS02] or employing the model for pre-selection only and adapting its influence [USZ04]. Apart from this issue, there is another point that appears questionable, both intuitively and in the light of the AIXI framework: The sampling strategy of an evolutionary algorithm is purely random and therefore often not optimal. Using a single Gaussian process model corresponds to a MAP estimator as opposed to the (preferable but more expensive) Bayesian mixture. The Gaussian process estimator is linear (and therefore unambiguous and unbiased) for fixed hyperparameters, but adapting the hyperparameters destroys this property.

Estimation-of-Distribution: The CMA-ES. Covariance Matrix Adaptation (CMA) [HO01] is generally accepted to be the most powerful available adaptation mechanism of the mutation distribution in Evolution Strategies. It can be interpreted as a local estimation-of-distribution method: The current best individual, the current step size, and the covariance matrix define together a distribution of good individuals. This distribution is used for sampling new individuals. The covariance matrix is adapted from the path of successful mutations. As already discussed, such a distribution contains less information than an explicit model, which may be also advantageous (in particular more robust). Given this model, random sampling from the distribution is optimal in order to find the optimum, no multi-step lookahead is needed. (However, this is not necessarily true for improving the model, depending on possible additional assumptions.) Note the parallel to the BOA [PGCP99] in the binary Genetic Algorithms (GA) setup, where however the model is obtained differently.

Global Optimization using Response Surfaces. A class of algorithms which is related to the AIXI framework very directly has been studied in [JSW98, Gut01, Jon01]. It works with

a global model (called response surface), which is a Gaussian process, or in case of [Gut01] a RBF network. New samples are obtained by directly maximizing a one-step lookahead criterion which depends on the model. The model is a MAP estimator instead of a Bayesian mixture. So these algorithms are actually heavily down-scaled AIXI-optimizers. Another approach in this class is the *mbminimize* algorithm [Pol02]. It maintains a global model (realized by a committee of arbitrary regression models) and employs a heuristic in order to balance exploration (experimental design) and exploitation (approaching the optimum). As already mentioned, the global optimization approach is problematic in increasing dimension, unless the objective function is very benign.

Derivative Free Optimization (DFO). A very interesting and powerful yet surprisingly little cited approach has been proposed in [CST97, CST98]. It is related to standard numerical optimization methods (such as BFGS), but does not make use of derivatives, nor tries to approximate them. Instead a local quadratic interpolation model is constructed from the appropriate number of previous function evaluations. The resulting information is then used in a trust-region optimization algorithm. This algorithm displays very interesting performance, however the interpolation is not supposed to be robust in the presence of noise. In fact, the algorithm we propose in the next section can be interpreted as a variant of the DFO, where the quadratic interpolation is substituted by a quadratic regression, and a different sampling strategy is employed.

4 A Local Quadratic Approach

After the preceding discussions, the following design of a model-based optimization algorithm for continuous functions on $\mathbb{R}^d$ seems natural. We would like to avoid the problem of false optima. So we choose a local model. We tried both local quadratic models and feed-forward networks, the former have been superior (see Fig. 3 below). The model is assumed to be valid in its local domain of ellipsoid shape (specified by a symmetric and positive definite transformation matrix $T \in \mathbb{R}^{d\times d}$ and a scale $\sigma > 0$) around a center $x_0 \in \mathbb{R}^d$. The domain is incrementally adapted, such that eventually the Hessian of the model is a multiple of the identity matrix. As already indicated, the sampling strategy is in order to improve the model, therefore points are sampled according to a space-filling criterion in the model domain. Afterwards the model domain is shifted towards decreasing function values. In the following, we give a precise specification of our LQM (Local Quadratic Model) algorithm.

4.1 The LQM Algorithm

We denote the set of points evaluated so far by X and the corresponding function values by Y. The current model is denoted by q, its domain is given by its center x_0, the transformation specified by T, and the scale σ. $\tilde{X}$ is the image of X under the inverse transformation. We will need the spectral decomposition (V, D) of a symmetric positive definite matrix A, where V is orthogonal and D is diagonal, this is abbreviated by *compute* $A = VD^2V'$. For notational convenience, operations are applied to sets with the obvious meaning, e.g. $V'(X - x_0) = \{\tilde{x} : \tilde{x} = V'(x - x_0) \,\forall\, x \in X\}$. (All sets are sorted, e.g. in chronological order, such that there is no correspondence problem.) We need a *weight function* $f_W : [0, \infty) \to [0, 1]$ which assigns weight 1 to points that are in the current model domain and smaller weights to the other points, according to their distance to x_0. $\|x\| = \sqrt{\langle x, x\rangle}$ denotes the Euclidean norm of a point $x \in \mathbb{R}^d$, and $\langle x, y\rangle$ the inner product of two points. Moreover, $r_s = \frac{1}{2d}$ is the search radius, r_{cur} the current step length, v_{cur} the current step direction, v_{last} the last step direction, and d_T the transformation update damping value.

Algorithm LQM

1. **Initialization**: Fix starting center $x_0 \in \mathbb{R}^d$, transformation matrix $T = I \in \mathbb{R}^{d\times d}$ (identity matrix) and its spectral decomposition $V_T = D_T = I$, and scale σ.
2. **Initial evaluations**: Let $X = \{x_0\} \cup X_S(x_0, \sigma)$, where $X_S(x_0, \sigma)$ contains the vertices of a regular simplex on the sphere with radius σ around x_0, and evaluate f on the set X.
3. **Transform and determine weights**: Compute $T = V_T D_T^2 V_T'$. Set
$$\tilde{X} = D_T^{-1}\sigma^{-1}V_T'(X - x_0) \text{ and } W = f_W(\|\tilde{X}\|).$$
4. **Compute model** q from $\tilde{X}$, Y, and W (see below).
5. **Shift model center**: Select $\tilde{x}_0 = \arg\min\{q(\tilde{x}) : \|\tilde{x}\| \le r_s\}$ and set $r_{cur} = \|\tilde{x}_0\|$, $x_0^{new} = x_0 + V_T D_T \sigma \tilde{x}_0$, $v_{cur} = x_0^{new} - x_0$, $x_0 = x_0^{new}$.
6. **Update scale** σ: If $r_{cur} \ge \frac{4}{5} r_s$ and $\langle v_{cur}, v_{last}\rangle \ge \frac{1}{2} r_s^2$, then increase σ to $2^{\frac{1}{4d}}\sigma$, else if $r_{cur} \le \frac{2}{5} r_s$ or $\langle v_{cur}, v_{last}\rangle \le -\frac{1}{2} r_s^2$, then decrease σ to $2^{-\frac{1}{4d}}\sigma$.
7. **Recalculate weights and model** as in steps 3 and 4.
8. **Update transformation**: Let H be the Hessian of the model q and $T_{new} = V_T D_T H^{-d_T} D_T V_T'$. Normalize T_{new} such that the product of its eigenvalues is 1, set $T = T_{new}$, and compute $T = V_T D_T^2 V_T'$.
9. **Select point for evaluation**: $\tilde{x}_{new} = \arg\max\{\min_{\tilde{x}\in\tilde{X}} \|\tilde{x}_1 - \tilde{x}\| : \|\tilde{x}_1\| \le 1\}$ (space-filling criterion) and $x_{new} = x_0 + V_T \sigma D_T \tilde{x}_{new}$.
10. **Evaluate**: Set $X = X \cup \{x_{new}\}$, $Y = Y \cup \{f(x_{new})\}$.
11. **Loop**: If termination criteria are not satisfied, go to step 3.

In order to compute the model (step 4), we solve the following least squares problem, which is standard except for the additional weights:

$$\min_\beta \stackrel{!}{=} \sum_{i=1}^{|\tilde{X}|} W_i^2 (Y_i - G(\tilde{X}_i)'\beta)^2 = (W(Y - G(\tilde{X})'\beta))'(W(Y - G(\tilde{X})'\beta)) \tag{2}$$

where $\beta \in \mathbb{R}^{\frac{d(d+1)}{2}+d+1}$ and $G(x)' = [1 \; x \; (x_i x_j)_{1\le i\le j\le d}] \in \mathbb{R}^{\frac{d(d+1)}{2}+d+1}$. Let $G_W(X_i) = W_i G(X_i)$, then the stationary point of the r.h.s. of (2) satisfies

$$\begin{aligned} 0 &= 2G_W(\tilde{X})G_W(\tilde{X})'\beta - 2G_W(\tilde{X})Y, \text{ implying} \\ \beta &= \left(G_W(\tilde{X})G_W(\tilde{X})'\right)^{-1} G_W(\tilde{X})'Y. \end{aligned}$$

If the matrix $G_W(\tilde{X})G_W(\tilde{X})'$ has not full rank, we use the quadratic model with diagonal terms only, or in case this also fails, the linear model. For the weight function f_W, we use

$$f_W(r) = \begin{cases} 1 & \text{if } r \le 1, \\ 1 - 2(r-1)^2 & \text{if } 1 < r \le \frac{3}{2}, \\ 2(2-r)^2 & \text{if } \frac{3}{2} < r \le 2, \\ 0 & \text{otherwise.} \end{cases}$$

This implies that the influence of points decreases smoothly with increasing distance to x_0 and thus the model depends continuously on x_0: This is the motivation for introducing the weights.

If the transformation update in step 8 were un-damped, then the new transformation matrix would be the inverse Hessian of the current model. This is easily seen, since $\tilde{x} = D_T^{-1} V_T' x$ implies $\tilde{x}' H \tilde{x} = x' V_T D_T^{-1} H D_T^{-1} V_T' x$. This specifies the Hessian in the untransformed space, and its inverse is $V_T D_T H^{-1} D_T V_T'$. However, without damping the transformation may change very rapidly, as a consequence only few points remain for modeling. Therefore we use $d_T = 0.2$, unless a larger value leads to a smaller condition of the transformation (we therefore also try $d_T = 0.4, 0.6, 0.8, 1$). That is, rapid changes towards the undistorted ball are admitted. Normalizing T ensures that the volume of the transformation ellipse remains constant.

We should motivate the choice of $r_s = \frac{1}{2d}$ for shifting the center in step 5 and in particular the constants arising in the scale update step 6 The choice of r_s matters in particular if the algorithm proceeds in one direction. Since one new point is evaluated in each step $r_s \simeq d^{-1}$ is natural in order to maintain d points in $\mathcal{X}$. The same argument applies to the update factors in step 6 (in addition, we assert $\sigma \in [\sigma_{min}, \sigma_{max}]$ there), that is in high dimension the scale is changed slowly. The scale is increased if the optimization proceeds essentially in one direction at maximum speed, and it is decreased if the center is shifted very slowly or oscillating (the term $\langle v_{cur}, v_{last} \rangle$ evaluates the angle between the last two steps). However, the exact constants in the conditions have been set to empirically motivated values. The problem of theoretically deriving a scale update strategy remains open.

4.2 Demonstration

We demonstrate the performance of the LQM algorithm using two benchmark functions in $\mathbb{R}^d$ for different d: The sphere function $f_1(x) = \|x\|^2$ and the generalized Rosenbrock function $f_{Rosen}(x) = \sum_{i=1}^{d-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$. The sphere function with optimum $[0 \ldots 0]$ is the prototype for an easy function and should be optimized quickly. The Rosenbrock function with optimum $[1 \ldots 1]$ requires a proper adaptation of the transformation in order to be minimized efficiently. We add experiments where the functions are corrupted by noise of amplitude 0.01, resulting in a "rough fitness landscape" and evaluating the ability of "stepping over" local optima. The starting point is set to $x_0 = [1 \ldots 1]$ (sphere) and $x_0 = [0 \ldots 0]$ (Rosenbrock), for the initial scale we choose $\sigma = 0.3$, the minimum scale is $\sigma_{min} = 0.01$. The LQM algorithm is compared to the Evolution Strategy with Covariance Matrix Adaptation, where we used the original MATLAB implementation from N. Hansen.

Figure 1 shows the average number of function evaluations needed to obtain function values stabilized below a threshold in the case of the sphere function. This threshold is set to $d \cdot 10^{-4}$ for the noiseless experiments, while for the noisy ones, we require that the *uncorrupted* function values stabilize below $d \cdot 10^{-3}$. Without noise, the LQM needs about $\frac{1}{6}$ of the function evaluations of CMA-ES on average, in the noisy case this ratio is even $\frac{1}{26}$. Most remarkably, with noise the LQM optimization is faster in dimension $d = 15$ than in dimension $d = 10$ on average. This is due to the fact that in $d = 15$ the quadratic model is often ill-defined and therefore the linear model (gradient estimate) is applied, which is sufficient for the sphere function.

This effect does not occur for the Rosenbrock function (Fig. 2), where the quadratic terms are necessary for the optimization. Here, the CMA-ES does not succeed to optimize the noisy function, while for LQM there seems to be almost no difference (but recall the different thresholds). Without noise, the LQM needs about $\frac{1}{4}$ of the function evaluations of

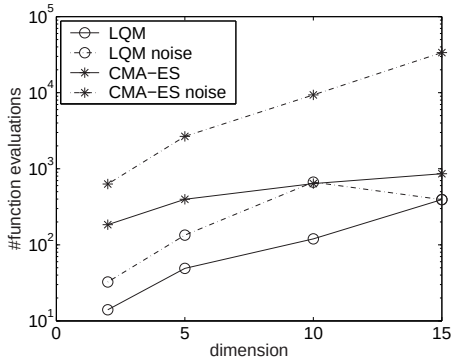


Figure 1: Sphere function: average number of function evaluations needed to stabilize the function values below the threshold ($d \cdot 10^{-4}$ for the noiseless and $d \cdot 10^{-3}$ for the noisy cases).

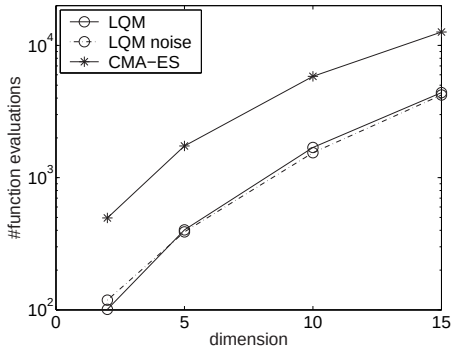


Figure 2: Rosenbrock function: average number of function evaluations needed to stabilize the function values below the threshold ($d \cdot 10^{-4}$ for the noiseless and $d \cdot 10^{-3}$ for the noisy cases). The CMA-ES does not achieve this goal in the noisy case.

CMA-ES on average. In Fig. 3, an example for $d = 5$ is shown. LQM converges rapidly. Noise almost does not affect the convergence speed in the beginning, but at a function value of about 10^{-3}, the optimization stagnates. This shows that the noise of amplitude 10^{-2} has been filtered out to a large extent. In contrast, the CMA-ES needs more function evaluations and stagnates with function values of about the noise amplitude. The last curve corresponds to a local model approach similar to LQM but with a feed-forward network instead of a quadratic model, the function evaluations have not been corrupted by noise. One can observe both slower convergence and robustness inferior to LQM.

5 Discussion and Open Problems

We have presented a general theoretical framework for optimization in terms of AIXI theory. Although this did not result in a directly realizable optimization algorithm, we have gained some insights and established recommendations for the design of such an algorithm. We proposed the LQM algorithm, which is a possible realization of these recommendations. We demonstrated that it is appropriate for reducing the number of function evaluations drastically compared to the CMA-ES, which is commonly accepted to be the most efficient (purely) evolutionary algorithm on continuous domain.

Many interesting questions remain open. First, a further elaboration on the consequences of AIXI theory for optimization algorithms is important. Can one derive strategies for the adaptation of the model domain, in particular the scale? So far, we have used ad hoc methods there. Also the sampling strategy could be improved, even if this might not result in very exciting changes of practical performance. Moreover, it would be interesting to know which model type is "better" (in some sense): The quadratic model we used, the commonly favored Gaussian process [BSK04], or yet a different one. For example, Fig. 4 shows that even quadratic models in dimension $d = 1$ without noise can have false optima. There could be an even more suitable set of basis functions for optimization. Finally, for functions like

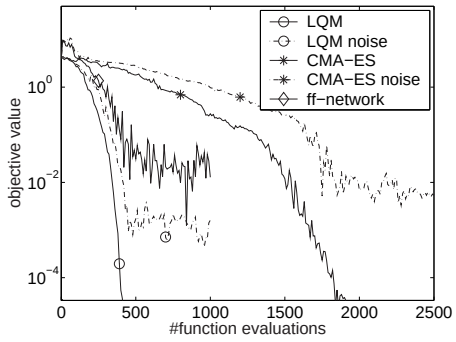


Figure 3: The Rosenbrock function: typical runs in $d = 5$. The last curve shows the performance of an LQM analog with a local feed-forward network instead of a quadratic model.

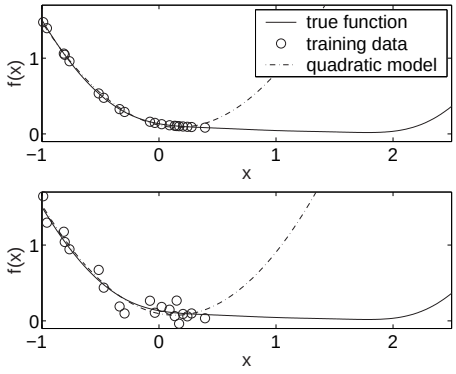


Figure 4: Also quadratic models can have false optima, even in dimension $d = 1$, both without noise (above) and with noise (below).

Rosenbrock, a number of function evaluations *quadratic* in d should not be necessary (as we need at the moment, and so does CMA-ES), since only a linear number of parameters in the covariance matrix does not vanish. One could achive this improvement by adapting the quadratic models using e.g. stepwise regression.

So far for continuous domains. For discrete domains, the same approach could be pursued, choosing an explicit model for functions on binary strings (the setup of Genetic Algorithms). Which model type is suitable for this purpose?

References

[BSK04] D. Büche, N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *EEE Transactions on Systems, Man, and Cybernetics C34*, 2004. to appear.

[Coh94] David A. Cohn. Neural network exploration using optimal experiment design. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 679–686. Morgan Kaufmann Publishers, 1994.

[CST97] A. R. Conn, K. Scheinberg, and Ph. L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79(3):397–415, 1997.

[CST98] A. R. Conn, K. Scheinberg, and Ph. L. Toint. A derivative free optimization algorithm in practice. In *Proceedings of the AIAA St Louis Conference*, 1998.

[EAG+02] Michael Emmerich, Alexios, Giotis, Mutlu Özdemir, Kyriakos Giannakoglou, and Thomas Bäck. Metamodel assisted evolution strategies. In J.J. Merelo et al., editor, *PPSN 6*, pages 361–270, 2002.

[EBK01] M. El-Beltagy and A. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In H. Arabnia, editor, *Proc. Int. Conf. on Artificial Intelligence*, pages 708–714, 2001.

[EBNK99] M. El-Beltagy, P. Nair, and A. Keane. Metamodeling techniques for evolutionary optimization of expensive problems: Promises and limitations. In W. Banzhaf et al, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 196–203, 1999.

[Gut01] H.-M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227, 2001.

[HO01] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[Hut00] M. Hutter. A theory of universal artificial intelligence based on algorithmic complexity. Technical Report cs.AI/0004001, München, 62 pages, April 2000.

[Hut01a] M. Hutter. Convergence and error bounds of universal prediction for general alphabet. *Proceedings of the 12th Eurpean Conference on Machine Learning (ECML-2001)*, pages 239–250, December 2001.

[Hut01b] M. Hutter. Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decisions. *Proceedings of the 12^{th} Eurpean Conference on Machine Learning (ECML-2001)*, pages 226–238, December 2001.

[Hut02] M. Hutter. Self-optimizing and Pareto-optimal policies in general environments based on Bayes-mixtures. In J. Kivinen and R. H. Sloan, editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT 2002)*, Lecture Notes in Artificial Intelligence, pages 364–379, Sydney, Australia, July 2002. Springer.

[Jon01] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[JOS02] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Trans. Evolutionary Computation*, 6(5):481–494, 2002.

[JSW98] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[LV97] M. Li and P. M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition, 1997.

[Paa97] G. Paass. Query sampling for prediction and model selection. *IPSJ Magazine*, 38(7):562–568, 1997.

[PGCP99] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian Optimization Algorithm. In W. Banzhaf et al, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 525–532. Morgan Kaufmann, 1999.

[PH03] J. Poland and M. Hutter. On the convergence speed of MDL predictions for Bernoulli sequences. preprint, 2003.

[Pol02] J. Poland. *Modellgestützte und Evolutionäre Optimierungsverfahren für die Motorentwicklung*. PhD thesis, University of Tübingen, 2002.

[Sol78] R. J. Solomonoff. Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Inform. Theory*, IT-24:422–432, 1978.

[USZ04] Holger Ulmer, Felix Streichert, and Andreas Zell. Evolution strategies with controlled model assistance. In *CEC*, 2004. to appear.