Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

Adapted maximum-likelihood Gaussian models for numerical optimization with continuous EDAs

P.A.N. Bosman, J. Grahl, D. Thierens

REPORT SEN-E0704 DECEMBER 2007

Centrum voor Wiskunde en Informatica (CWI) is the national research institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organisation for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

Adapted maximum-likelihood Gaussian models for numerical optimization with continuous EDAs

ABSTRACT
This article focuses on numerical optimization with continuous Estimation-of-Distribution Algorithms (EDAs). Specifically, the focus is on the use of one of the most common and best understood probability distributions: the normal distribution. We first give an overview of the existing research on this topic. We then point out a source of inefficiency in EDAs that make use of the normal distribution with maximum-likelihood (ML) estimates. Scaling the covariance matrix beyond its ML estimate does not remove this inefficiency. To remove the inefficiency, the orientation of the normal distribution must be changed. So far, only Evolution Strategies (ES) and particularly Covariance Matrix Adaptation ES (CMA-ES) are capable of achieving such re-orientation. In this article we provide a simple, but effective technique for achieving re-orientation while still only performing the well-known ML estimates. We call the new technique Anticipated Mean Shift (AMS). The resulting EDA, called Adapted Maximum-Likelihood Gaussian Model -- Iterated Density-Estimation Evolutionary Algorithm (AMaLGaM-IDEA) adapts not only the ML estimate for the covariance matrix, but also the ML estimate for the mean. AMaLGaM-IDEA has an improved performance compared to previous EDAs that use ML estimates as well as compared to previous EDAs that scale the variance adaptively. Also, we indicate the circumstances under which AMaLGaM-IDEA is found to be robust to rotations of the search space. A comparison with CMA-ES identifies the conditions under which AMaLGaM-IDEA is able to outperform CMA-ES and vice versa. We conclude that AMaLGaM-IDEA is currently among the most efficient real-valued continuous EDAs while at the same time it is relatively simple to understand (especially in the naive, univariate case). Pseudo-code is provided in this article; source-code can be downloaded from the web.

2000 Mathematics Subject Classification: 49M99
1998 ACM Computing Classification System: G.1; I.2
Keywords and Phrases: estimation--of--distribution algorithms; evolutionary algorithms; normal distribution; maximum likelihood; optimization; adaptive variance scaling; anticipation

Adapted Maximum–Likelihood Gaussian Models for Numerical Optimization with Continuous EDAs

Peter A.N. Bosman*, Jörn Grahl† and Dirk Thierens‡

December 2007

Abstract

This article focuses on numerical optimization with continuous Estimation–of–Distribution Algorithms (EDAs). Specifically, the focus is on the use of one of the most common and best understood probability distributions: the normal distribution. We first give an overview of the existing research on this topic. We then point out a source of inefficiency in EDAs that make use of the normal distribution with maximum–likelihood (ML) estimates. Scaling the covariance matrix beyond its ML estimate does not remove this inefficiency. To remove the inefficiency, the orientation of the normal distribution must be changed. So far, only Evolution Strategies (ES) and particularly Covariance Matrix Adaptation ES (CMA–ES) are capable of achieving such re–orientation. In this article we provide a simple, but effective technique for achieving re–orientation while still only performing the well–known ML estimates. We call the new technique Anticipated Mean Shift (AMS). The resulting EDA, called Adapted Maximum–Likelihood Gaussian Model — Iterated Density–Estimation Evolutionary Algorithm (AMaLGaM–ID$\mathbb{E}$A) adapts not only the ML estimate for the covariance matrix, but also the ML estimate for the mean. AMaLGaM–ID$\mathbb{E}$A has an improved performance compared to previous EDAs that use ML estimates as well as compared to previous EDAs that scale the variance adaptively. Also, we indicate the circumstances under which AMaLGaM–ID$\mathbb{E}$A is found to be robust to rotations of the search space. A comparison with CMA–ES identifies the conditions under which AMaLGaM–ID$\mathbb{E}$A is able to outperform CMA–ES and vice versa. We conclude that AMaLGaM–ID$\mathbb{E}$A is currently among the most efficient real–valued continuous EDAs while at the same time it is relatively simple to understand (especially in the naive, univariate case). Pseudo–code is provided in this article; source–code can be downloaded from the web.

Keywords: estimation–of–distribution algorithms, evolutionary algorithms, normal distribution, maximum likelihood, optimization, adaptive variance scaling, anticipation.

1 Introduction

The premise of the Estimation–of–Distribution Algorithm (EDA) is a strong one; its methodology is principled and elegant. It is for this reason that research into EDAs has gained interest since its introduction [4, 21, 18, 19, 23].

EDAs can be seen as a specific type of Evolutionary Algorithm (EA). It is as such that EDAs have first been introduced and it is also in the EA research community that EDAs have received the most attention. The only characteristic that sets EDAs apart from other EAs is the way in which new solutions are generated. Instead of repeatedly combining information of only a few solutions, EDAs combine the information of all selected solutions at once. This is done by using an interim representation that compresses and summarizes this information: a probability distribution over the solution space. New solutions are generated by sampling the distribution. The premise that flows from this methodology is one of efficient optimization under suitable conditions. Initially, there is typically a uniform distribution over the entire search space. This also means that the probability distribution is uniform over all solutions that have a quality at least as good as that of the worst solution. If this latter condition remains true after selection, the probability distribution will zoom in exponentially fast on the optimal solution [11]. A similar argument can be made for the use of the Boltzmann distribution in EDAs [20].

In practice we generally do not have access to this powerful uniform probability distribution or to the Boltzmann distribution. Moreover, these probability distributions can be arbitrarily complex as the problem itself can be arbitrarily complicated. Hence, to obtain an actual EDA to work with, we must approximate the probability distribution using practical techniques. In the continuous case, the most common technique is the use of the normal distribution or combinations thereof. It is not surprising that some of the first EDAs in continuous spaces were based on the normal distribution. The most important question is of course how efficient EDAs are in the continuous domain if we use approximated probability distributions.

In this article, we focus particularly on the normal distribution. Much is known about the normal distribution, allowing both analytical and experimental analysis to be done. In this article we summarize the current state of the research on using the normal distribution in EDAs. This research mostly pertains to certain factorizations of the normal distribution. We complement the overview by providing additional results and insights for the unfactorized normal distribution, which corresponds to using a full covariance matrix. By doing so, we paint a more complete picture of continuous EDAs based on the normal distribution than ever before. We discuss the earliest approaches in which maximum–likelihood (ML) estimates are computed from the selected solutions and used directly to generate new solutions. We then point out how it was recently shown that without precaution, premature convergence is likely to occur with these approaches, even on slope–like regions of the search space. The main reason for this is that the variance decreases too fast. The current state of the art exists of techniques that attempt

*P.A.N. Bosman is with the Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands (e-mail: Peter.Bosman@cwi.nl).

†J. Grahl is with the Dept. of Logistics, University of Mannheim, Mannheim, Germany (e-mail: joern.grahl@bwl.uni-mannheim.de).

‡D. Thierens is with the Institute of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands (e-mail: Dirk.Thierens@cs.uu.nl).

to remedy premature convergence (e.g. adaptive variance scaling [22, 14]). In this article, we show that a source of inefficiency however exists that cannot be removed by these remedies. This inefficiency has not come to light before because of the way most benchmark problems are used for testing: symmetrically initialized around the optimum. The source of the inefficiency lies in the fact that the use of ML estimates results in a fit that describes the set of selected solutions well. When on a slope however, it is not always the configuration of the set of selected solutions that is interesting, but it is the direction of descent. With at least two problem variables, the configuration of the selected solutions may however be different from the orientation of the direction of descent. When this happens, the shape of the estimated distribution can become strongly misaligned with the direction of descent if covariances are estimated also. Improved solutions are then not found efficiently. Scaling the variance doesn't improve efficiency. To improve efficiency, the estimated distribution needs to be reshaped. Currently, there is no EDA that uses ML estimates and achieves such reshaping of these estimates. In this article we provide a simple, yet elegant extension to the current state–of–the–art EDAs that does achieve the required reshaping. We call this extension AMS (Anticipated Mean Shift). The resulting EDA has an improved performance, even if no covariances are estimated. Also, the algorithm still only makes use of ML estimates which are well understood and provide a sensible way of estimating parameters from data.

The goal in this article is twofold. On the one hand we shall paint a more complete picture of EDAs based on the normal distribution than ever before. To do so, we'll revisit existing algorithmic design principles and theoretic results and align them to arrive at an insightful line of research. We will cite the existing literature whenever relevant. On the other hand we shall take an important next step along this line of research in designing efficient EDAs based on the normal distribution and ultimately arrive at a new EDA called AMaLGaM–ID$\mathbb{E}$A (Adapted Maximum–Likelihood Gaussian Model — Iterated Density–Estimation Evolutionary Algorithm). We illustrate the drawbacks and advances of the methods described in this article and compare the results of AMaLGaM–ID$\mathbb{E}$A with CMA–ES, currently the most efficient evolution strategy for continuous optimization.

The remainder of this article is organized as follows. In Section 2 we present basic definitions and first results of EDAs based on the normal distribution with ML estimates. Subsequently, in Section 3 we discuss the technique of adaptive variance scaling that is used to prevent premature convergence. We present new results that signal a source of inefficiency and we identify this source. Then, in Section 4, we present the technique of anticipated mean shift that removes the identified inefficiency. In Section 5 we provide guidelines for setting the population size and compare the resulting EDA with the CMA–ES on a set of 10 benchmark problems using different initialization ranges as well as rotations of the benchmark problems. We discuss our findings and identify avenues of future research in Section 6. We sum up and conclude this article in Section 7.

2 Maximum-Likelihood Gaussian Model

The use of the normal distribution is a logical choice in the design of continuous EDAs since this distribution is well understood and relatively simple.

2.1 Definition, Factorization, Estimation and Sampling

We introduce a random variable X_i for each real–valued problem variable x_i, $i \in \{0, 1, \ldots, l-1\}$ where l is the problem dimensionality. The normal distribution $P^{\mathcal{N}}_{(\boldsymbol{\mu}_{\boldsymbol{v}}, \boldsymbol{\Sigma}^{\boldsymbol{v}})}(X_{\boldsymbol{v}})$ for a vector of random variables $X_{\boldsymbol{v}} = (X_{v_0}, X_{v_1}, \ldots, X_{v_{|\boldsymbol{v}|-1}})$ is parameterized by a vector $\boldsymbol{\mu}_{\boldsymbol{v}}$ of means and a symmetric covariance matrix $\boldsymbol{\Sigma}^{\boldsymbol{v}}$. It is defined as follows:

$$P^{\mathcal{N}}_{(\boldsymbol{\mu}_{\boldsymbol{v}}, \boldsymbol{\Sigma}^{\boldsymbol{v}})}(X_{\boldsymbol{v}} = \boldsymbol{x}) = \frac{(2\pi)^{-\frac{|\boldsymbol{v}|}{2}}}{(\det \boldsymbol{\Sigma}^{\boldsymbol{v}})^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{v}})^T (\boldsymbol{\Sigma}^{\boldsymbol{v}})^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{v}})} \quad (1)$$

Maximum–likelihood (ML) parameter estimation is a common way of estimating probability distributions. It is a principled approach that effectively minimizes the empirical error and thereby maximizes the match between the probability distribution and all given data [28]. A ML estimation for the parameters of the normal distribution is obtained from a vector $\boldsymbol{\mathcal{S}}$ of solutions if $\boldsymbol{\mu}_{\boldsymbol{v}}$ and $\boldsymbol{\Sigma}^{\boldsymbol{v}}$ are estimated by the sample average and sample covariance matrix respectively [2, 27]:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{v}} = \frac{1}{|\boldsymbol{\mathcal{S}}|} \sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} (\boldsymbol{\mathcal{S}}_j)_{\boldsymbol{v}} \quad (2)$$

$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{v}} = \frac{1}{|\boldsymbol{\mathcal{S}}|} \sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} ((\boldsymbol{\mathcal{S}}_j)_{\boldsymbol{v}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{v}})((\boldsymbol{\mathcal{S}}_j)_{\boldsymbol{v}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{v}})^T$$

The number of parameters to be estimated equals $\frac{1}{2}|\boldsymbol{v}|^2 + \frac{3}{2}|\boldsymbol{v}|$. Different from the discrete case, the number of parameters to be estimated therefore does not grow exponentially with $|\boldsymbol{v}|$ but quadratically.

Initial EDAs that used the normal distribution employed the univariate factorization [24, 25]. In the univariate factorization, the distribution is estimated separately for each variable. Similarly, the generation of a new solution is done by sampling each of the one–dimensional normal distributions separately. Such EDAs are sometimes also called naive EDAs [10]. Let $\boldsymbol{\mathcal{X}} = (X_0, X_1, \ldots, X_{l-1})$ be a vector containing all random variables. The univariate factorization then is defined as follows:

$$P(\boldsymbol{\mathcal{X}}) = \prod_{i=0}^{l-1} P(X_i) \quad (3)$$

Estimating the univariate factorization in the case of the normal distribution entails the estimation of l means and variances. Sampling a new solution is also quite straightforward as only sampling from a single–dimensional normal distribution is required; once for each of the l variables. It was realized in subsequent research that using the univariate factorization may give rise to a clear mismatch between the model and the fitness landscape if there are dependencies between problem variables. If the fitness landscape for instance is an ellipsoid that is not aligned with the main axes (i.e. it is rotated), the density contours of the univariately factorized normal distribution cannot match the contour lines of the fitness landscape well. Therefore, efficient sampling of solutions of a certain minimal quality is prohibited. To incorporate dependencies, the full covariance matrix could be used instead, i.e. use Equation 2 with $X_{\boldsymbol{v}} = \boldsymbol{\mathcal{X}}$. Alternatively, it is possible to use a greedy algorithm to determine the

most important dependencies and use only those. To this end, a Bayesian factorization can be estimated.

To briefly recall Bayesian factorizations, recall that the vector of random variables indicated by $X_{\boldsymbol{\pi}_i}$ on which X_i is conditioned in the Bayesian factorization is called the vector of parents of X_i. A Bayesian factorization can now be written as follows:

$$P(\boldsymbol{\mathcal{X}}) = \prod_{i=0}^{l-1} P(X_i | X_{\boldsymbol{\pi}_i}) \tag{4}$$

A greedy learning algorithm is (typically) used to compute the Bayesian factorization. For more details, we refer the interested reader to the relevant EDA literature [18, 19, 23]. The factorization expresses a subset of all dependencies between the variables that are subject to search and hence allows rotation of the multivariate normal density, resulting in search directions that differ from the axis–parallel directions.

To estimate the conditional distributions $P^{\mathcal{N}}(X_i|X_{\boldsymbol{\pi}_i})$ when constructing Bayesian factorizations, let $\boldsymbol{W^j}$ be the inverse of the symmetric covariance matrix, that is $\boldsymbol{W^j} = (\boldsymbol{\Sigma^j})^{-1}$. Matrix $\boldsymbol{W^j}$ is commonly called the precision matrix. It can be shown that a ML estimate of $P^{\mathcal{N}}(X_i|X_{\boldsymbol{\pi}_i})$ can be expressed in terms of Equation 2 [8]:

$$\hat{P}^{\mathcal{N}}(X_i = x_i \mid X_{\boldsymbol{\pi}_i} = x_{\boldsymbol{\pi}_i}) = \frac{1}{(\breve{\sigma}_i\sqrt{2\pi})} e^{\frac{-(x_i-\breve{\mu}_i)^2}{2\breve{\sigma}_i^2}} \tag{5}$$

$$\text{where } \begin{cases} \breve{\sigma}_i & = & \frac{1}{\sqrt{\hat{\boldsymbol{W}}_{00}^{(i,\boldsymbol{\pi}_i)}}} \\ \breve{\mu}_i & = & \frac{\hat{\boldsymbol{\mu}}_i \hat{\boldsymbol{W}}_{00}^{(i,\boldsymbol{\pi}_i)} - \sum_{j=0}^{|\boldsymbol{\pi}_i|-1} (x_{(\boldsymbol{\pi}_i)_j} - \hat{\boldsymbol{\mu}}_{(\boldsymbol{\pi}_i)_j}) \hat{\boldsymbol{W}}_{(j+1)0}^{(i,\boldsymbol{\pi}_i)}}{\hat{\boldsymbol{W}}_{00}^{(i,\boldsymbol{\pi}_i)}} \end{cases}$$

Because Equation 5 has the form of a single dimensional normal distribution, sampling from the Bayesian factorization is again straightforward once all relevant computations have been performed.

The density contours of a normal factorized probability distribution are ellipsoids. Depending on the dependencies modeled by the factorization, these ellipsoids can be aligned along any axis. If there is no dependency between a set of random variables (i.e. the univariate factorization), the projected density contours in those dimensions are aligned along the main axes. Use of the complete covariance matrix can be established by using a Bayesian factorization in which each X_i is conditioned on all X_j with $j > i$. In any case, a normal distribution is only capable of efficiently modeling linear dependencies.

2.2 EDA Performance

The use of Bayesian factorizations led to improved results in terms of a smaller number of required evaluations to obtain a solution of a certain quality on various benchmarks [8, 17, 7, 9]. For the benchmarks used, the algorithms even compared favorably compared to evolution strategies (ES [3]). Without loss of generalization, we only consider minimization problems in this article. As an illustration, consider the following two unimodal minimization problems:

Name	Definition	Value to reach (VTR)
Sphere	$\sum_{i=0}^{l-1} x_i^2$	10^{-10}
Ellipsoid	$\sum_{i=0}^{l-1} 10^{6\frac{i}{l-1}} x_i^2$	10^{-10}

We also consider rotations of these functions. Specifically, we consider rotations of θ degrees. The reason for this is that the problems above do not have any dependencies between their problem variables. Rotating the search space by, for instance, 45 degrees for every combination of variables introduces strong dependencies between the problem variables. These dependencies become harder to cope with as the elongation of the landscape becomes more heterogeneous along the different axes. For the Sphere function, the elongations are homogeneous and rotations thus have no influence. For the Ellipsoid function however, each quadratic form is scaled differently. Hence, rotating this function introduces dependencies. To perform rotations, we define rotation matrices $R^{lij}(\theta)$ for each dimensionality l and each $i, j \in \{0, 1, \ldots, l-1\}, j > i$. Rotation matrix $R^{lij}(\theta)$ is the rotation matrix that rotates the plane spanned by dimensions i and j, i.e. $R^{lij}(\theta)$ is the identity matrix of dimension $l \times l$ but with $R^{lij}_{ii}(\theta) = \cos(\theta)$, $R^{lij}_{ij}(\theta) = -\sin(\theta)$, $R^{lij}_{ji}(\theta) = \sin(\theta)$ and $R^{lij}_{jj}(\theta) = \cos(\theta)$. We now define a full rotation matrix $R^l(\theta)$ as the product of all pairwise rotation matrices, i.e.

$$R^l(\theta) = \prod_{i=0}^{l-1} \prod_{j=i+1}^{l-1} R^{lij}(\theta) \tag{6}$$

Function $f(\boldsymbol{x})$ then can be rotated by evaluating it as $f(\boldsymbol{y})$ with $\boldsymbol{y} = R^l(\theta)\boldsymbol{x}$.

One specific EDA that uses Equation 5 to draw new samples from the normal distribution is called the IDEA (Iterated Density–Estimation Evolutionary Algorithms) [8]. The IDEA was initially introduced as a framework and an alternative acronym to EDA. It has in the literature however often been associated specifically with the use of the normal distribution in combination with Equation 5. We use this algorithm and the IDEA acronym throughout the remainder of this article in our experiments.

2.2.1 Symmetric initialization

Traditionally, EAs have been benchmarked using initialization ranges that are centered around the optimum. This was also the case for the first results reported for continuous EDAs. We ran experiments for $l \in \{2, 4, 8, 10, 20, 40, 80\}$. Truncation selection was used with $\tau = 0.3$ and elitist replacement (i.e. all selected solutions are preserved and all non–selected solutions are replaced with newly generated solutions). This scheme will be used throughout the remainder of this article. The optimal population size is determined. By optimal population size we mean the population size for which the minimum number of evaluations is obtained to reach the value–to–reach (VTR) in at least 95 of 100 independent runs. The maximum population size we allowed is 10^5. Results on the rotated Ellipsoid function were not reported in the literature before as also wasn't the scalability of an EDA that uses the univariate factorization or the full covariance matrix on any problem. The results for an initialization range of $[-7.5, 7.5]$ for each variable are presented in Figure 1.

The results are quite like what one would expect. The univariate factorization requires the smallest population sizes because the smallest number of parameters is required for the estimates to become reliable. The univariate factorization is however unable to solve the rotated Ellipsoid benchmark. The univariate factorization is well–suited only for problems without dependencies. The Bayesian factorization with greedy learning is in addi-

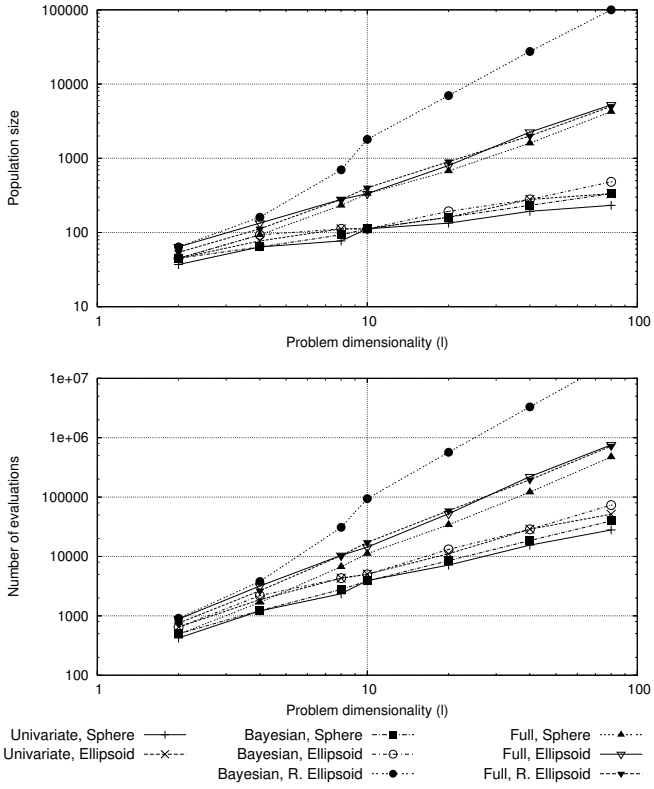

Figure 1: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) required to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-7.5, 7.5]$ for each variable (symmetric initialization). Maximum–likelihood estimates are used.

tion well–suited for problems with a low to medium number of dependencies between the problem variables. The Bayesian factorization does require a slightly larger population size for problems without dependencies because some overhead is required to detect that there are no dependencies. Because dependencies can be incorporated, it is able to solve the rotated Ellipsoid function. For the greedy search algorithm and the search metric used it is typically hard to detect all dependencies though. Very large population sizes are required to this end. It is for this reason that the learning approach is able to solve the rotated Ellipsoid benchmark, but it is not as efficient as when the full covariance matrix is used and all dependencies are automatically taken into account in the EDA. The unfactorized normal distribution, corresponding to the use of the full covariance matrix, is well–suited for any order of linear dependency between the problem variables. The scalability is independent of rotation of the search space. This observation has not been reported in the literature on continuous EDAs before. The observed behavior is however what one would expect given that with a full covariance matrix there are no restrictions on the orientation of the probabilistic search direction. The EDA that uses the full covariance matrix is the most powerful variant in the sense that it is able to solve the largest class of problems. The downside is that it requires the largest population size because it requires the most parameters to be estimated.

One problem with these results is that of an undesirably big bias. The optima of the problems are in the center of the initialization ranges. The normal distribution has the largest density at its mean, which lies in its center. An EDA based on the normal distribution with ML estimates thus wants to focus its search by contracting the region being explored towards its estimated mean. Hence, the problems at hand and the search bias of the EDA are very favorably matched and may have led to overenthousiastic conclusions.

2.2.2 Asymmetric initialization

In recent years, it has become more common to combine the benchmark functions with asymmetric initialization around the optimum such as $[-10, 5]$. Retesting the original EDAs that use ML estimates with this initialization range indeed produces worse results, see Figure 2. For full covariance matrix variants, the population size even no longer scales polynomially, but exponentially on all problems. We shall find the underlying reason for this specific deterioration to exponential scalability later on in Section 3.3.3.

Figure 3 shows the effect that the initialization range has on the optimization performance on a sliding scale. The initialization range moves from the symmetric case to the asymmetric case and beyond toward the far–away case. A range shift of 0 corresponds to symmetric initialization. A range shift of 2.5 corresponds to the asymmetric initialization we used in this article. The far–away initialization we used corresponds to a range shift of 107.5 (not included in the range in Figure 3).

From the results in Figure 3 it can be seen very clearly that the performance of the EDA with ML estimates of the normal distribution is compromised very quickly as the initialization range moves away from the symmetric case. Using the full covariance matrix even results in faster deterioration than using the univariate factorization. The underlying reason is the same as why the full covariance matrix results in an exponential scale–up for the case of asymmetric initialization. We return to this issue in Section 3.3.3.

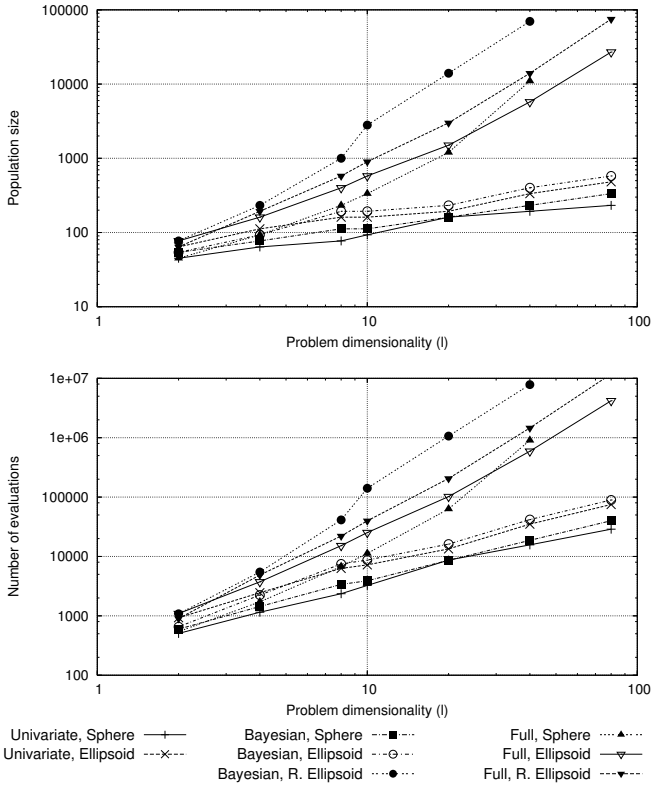

Figure 2: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) required to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-10, 5]$ for each variable (asymmetric initialization). Maximum–likelihood estimates are used.

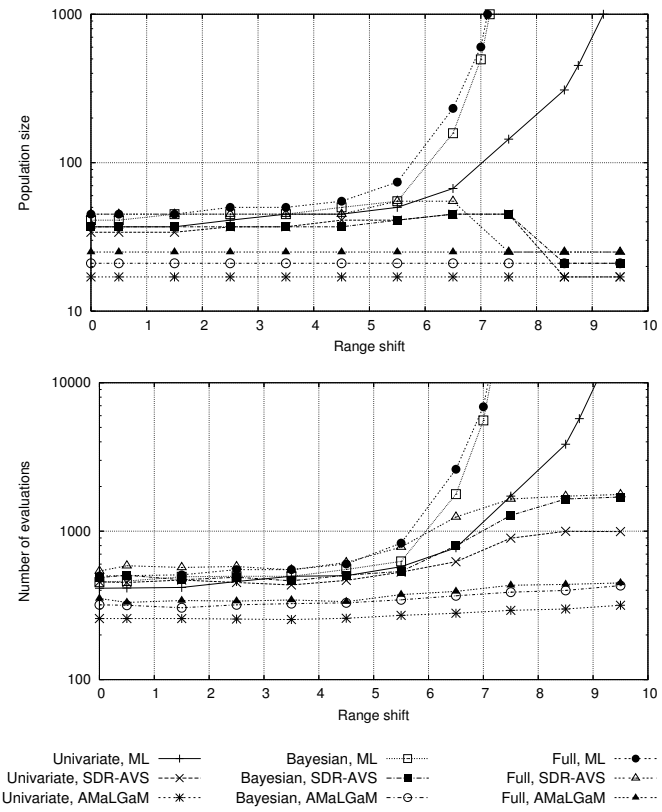

Figure 3: The population size (*top*) that leads to the minimum number of evaluations (*bottom*) to reach the VTR on the Sphere function for $l = 2$, averaged over 100 independent runs. The initialization range is $[-7.5 - s, 7.5 - s]$ where s is the range shift.

2.2.3 Far–away initialization

Limitations of the use of ML estimates of the normal distribution in an EDA were experimentally noticed in the literature before [9], most notably on Rosenbrock's function:

Name	Definition	VTR
Rosenbrock	$\sum_{i=0}^{l-2} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	10^{-10}

Rosenbrock's function is a smooth function with low–order dependencies. Only subsequent variables are directly linked. This means that using the Bayesian factorization it is to be expected that the EDA is able to reliably find the optimum. This is however not the case. For $l > 2$ the required population size quickly becomes extraordinarily large. This has led researchers to investigate what goes wrong in more detail [12, 15, 29]. Rosenbrock's function has a parabolically shaped sharp valley. The landscape outside and inside the valley is relatively flat. The valley is too sharp to be able to obtain many points inside it upon initialization. As a result, the population quickly moves toward a certain part of the valley, depending on the population configuration upon initialization. Once the population has focused somewhere in the valley, the valley must be traversed to find the optimum. The optimum is now however no longer contained somewhere in the region that is covered by the population. Using ML estimates of the normal distribution then the search contracts somewhere near where the valley was entered and the optimum is not found.

This observation means that a much simpler case can be studied to understand what is going wrong. The trajectory to be traversed along the bottom of the valley can be seen as a one–dimensional path to be followed through a high–dimensional space with the optimum far away from the current location of the population. Hence, we are in a situation in which the optimum is not contained in the initialization range. This corresponds to the situation on the far right in Figure 3. The population size indeed increases radically fast as the initialization range shifts away from the optimum. The results even suggest that an asymptote exists to which the maximum range shift converges with an increase in population size.

For a closer analysis, we can just take the one–dimensional slope function $f(x_0) = x_0$. As this function does not have a bounded minimum, the minimum is certainly not contained in any initialization range. For the initialization range $[-5, 5]$, Figure 4 shows the variance and mean in subsequent generations with a population size of 50. Clearly, premature convergence occurs as the variance vanishes exponentially fast. The EDA using ML estimates is not able to traverse the slope to $-\infty$, even though the population size is large enough for a one–dimensional problem.

It is perhaps even more striking to see that the EDA is not able to move the solutions far outside the initialization range. It was proven that indeed the mean can only shift a bounded length from its initial position [5]. Let $\hat{\mu}_0(t)$ be the mean of the one–dimensional normal distribution in generation t and let $\hat{\sigma}_0^2(t)$ be the variance. Summarizing the result, the mean obeys[1]:

$$\lim_{t\to\infty} |\hat{\mu}_0(t) - \hat{\mu}_0(0)| = \hat{\sigma}_0(0)\frac{d(\tau)}{1 - \sqrt{c(\tau)}} \qquad (7)$$

For the derivation of the definitions of $c(\tau)$ and $d(\tau)$ we refer the interested reader to the relevant literature [5]. For $\tau = 0.3$

[1]Functions $c(\tau)$ and $d(\tau)$ are accidentally interchanged in Equations 9 and 11 in reference [5]. Here, in Equation 7, they are used correctly.

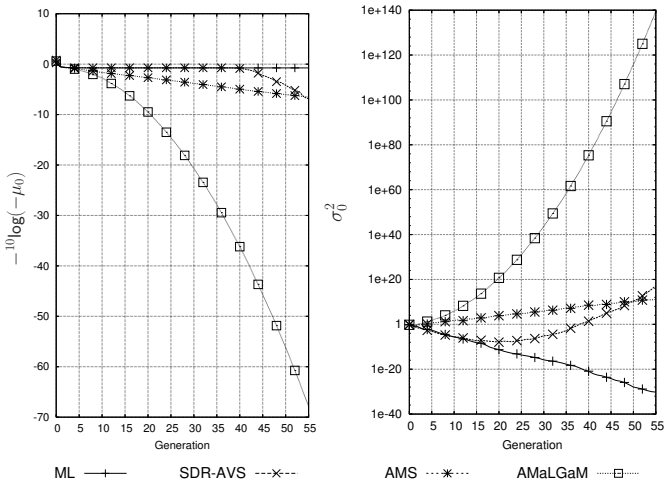

Figure 4: Progression of estimated mean (*left*) and variance (*right)* in subsequent generations for typical runs of EDAs with different models on the one–dimensional slope function $f(x_0) = x_0$ with initialization range $[-5, 5]$.

that we use throughout this article for instance, $c(\tau) \approx 0.27$ and $d(\tau) \approx 1.2$, meaning a maximum shift of only roughly 2.5 times the initial standard deviation, given an infinite population size. For the example illustrated in Figure 4, we find $\hat{\mu}_0(0) \approx -3.0$ and $\hat{\sigma}_0(0) \approx 0.95$. The maximum shift therefore becomes ≈ 2.4, meaning that the mean cannot shift lower than ≈ -5.4. With the variance vanishing further with each generation, this means that convergence will take place to no lower than that same value. In our example the final value found for the EDA with ML estimates is ≈ -5.3.

Concluding, running the EDA with ML estimates of the normal distribution is not able to solve problems if initialization is far away from the optimum or if a similar situation presents itself during optimization (e.g. on Rosenbrock's function). The EDA is not able to traverse the search space. It is only effective at contraction.

3 Standard-Deviation-Ratio-Triggered Adaptive Variance Scaling

To remedy the problem of the prematurely vanishing variance, the variance can be scaled adaptively beyond its ML estimate [22, 14, 5]. The ML estimate gives a reliable configuration of the normal distribution for describing the selected solutions in the current generation. The adaptivity must then lie in determining whether the spread of the distribution should be enlarged to search outside the focus prescribed by the ML estimate. One successful scheme for doing variance scaling in an adaptive fashion (i.e. during optimization) was recently introduced under the name adaptive variance scaling (AVS) [14]. This scheme significantly improves performance and allows the EDA to solve problems that it couldn't solve without scaling the variance.

3.1 Adaptive Variance Scaling (AVS)

The smaller the variance, the smaller the area of exploration for the EDA. The variance in the normal distribution is stored in the covariance matrix $\mathbf{\Sigma}$. With AVS, a variance multiplier c^{AVS}

is maintained. Upon sampling new solutions, the distribution is scaled by c^{AVS}, i.e. the covariance matrix used for sampling is $c^{\text{AVS}}\mathbf{\Sigma}$ instead of just $\mathbf{\Sigma}$. If the best fitness value improves in one generation, then the current size of the variance allows for progress. A further enlargement of the variance may allow further improvement in the next generation. To fight the variance–diminishing effect of selection, the size of c^{AVS} is scaled by $\eta^{\text{Increase}} > 1$. If on the other hand the best fitness does not improve, the range of exploration may be too large to be effective and the variance multiplier should be decreased by a factor $\eta^{\text{Decrease}} \in [0, 1]$. For symmetry, $\eta^{\text{Decrease}} = 1/\eta^{\text{Increase}}$. As the objective of the AVS scheme is to enlarge the variance to prevent premature convergence, c^{AVS} is not allowed to become smaller than 1. This scheme deviates slightly from the original implementation of AVS [14, 5]. In the original implementation, the magnitude of c^{AVS} was bounded from above by a predefined value $c^{\text{AVS–MAX}} > 1$ and from below by $c^{\text{AVS–MIN}} < 1$. The upper bound is however not needed as the variance will automatically grow into the maximum variance for which improvements can still be obtained. The lower bound was introduced to allow the variance to shrink to less than its original size. This allows the algorithm to choose a niche in the case of a multimodal landscape. As we are here only interested in unimodal landscapes, we simplify the scheme and enforce $c^{\text{AVS}} \geq 1$.

3.2 Standard–Deviation Ratio (SDR) Trigger

In the AVS scheme, improved fitness values automatically increase c^{AVS}. Improved fitness values however do not always mean that the variance needs to be enlarged. This is especially the case if the normal kernel is near the optimum. In this case, the induced bias of the normal distribution already leads the EDA to the optimum as was observed in Section 2. Increasing the variance will then only slow down convergence, as the EDA is forced to explore a larger area of the search space unnecessarily. It is therefore sensible to attempt to separate two cases: traversing a slope, and searching around an optimum. A first approach to designing such a trigger used the ranked correlation between the density of the selected solutions and their fitness values [14]. If correlation is strong, then the search is focused around the optimum and no variance scaling is required.

Because the correlation coefficient is computed for all variables jointly, this approach doesn't always work, especially in higher dimensions. Suppose that all but a few dimensions do not require the scaling of variances. The contribution from the few non–correlated dimensions to the correlation measure becomes insignificant as the dimensionality increases. As a result, variance scaling is no longer triggered. Without variance scaling however, the ML normal EDA fails in the dimensions where scaling is required and hence, optimization fails altogether.

This motivates looking at the search directions of the EDA separately. A recent approach does so by focusing on the Bayesian factorization [6]. If improvements mostly take place far away from the mean, then obviously, the mean needs to shift. As we know that mean–shift is problematic for ML normal EDAs, this is a situation in which AVS is called for. If however most of the improvements are obtained near the mean, then the EDA with ML parameters already has a good focus and no further variance enlargement is required. It is known (see, e.g. [1]) that for any value of the standard deviation σ, a fixed percentage of the density of the normal distribution is contained within $[\mu - c\sigma, \mu + c\sigma]$ where μ is the mean of the normal distribution and $c \geq 0$. Now, let $\overline{\boldsymbol{x}}^{\text{Improvement}}$ denote the average of all new samples drawn that were an improvement over the set of selected solutions in that same generation. A threshold $\theta^{\text{SDR}} \in [0, \infty]$ is used that triggers the further enlargement of the variance multiplier in the next generation only if $\overline{\boldsymbol{x}}^{\text{Improvement}}$ has a distance d to the estimated mean $\hat{\mu}$ such that $d/\hat{\sigma} > \theta^{\text{SDR}}$. Otherwise, the variance multiplier should remain unchanged. Note that this trigger is independent of the sample range and has a fixed, predefined notion of being "close" to the mean.

This method can easily be factorized according to the search distribution of the EDA by following the Bayesian factorization that was estimated from the selected solutions as defined in Equation 5. For each factor separately the standard–deviation ratio of $\overline{\boldsymbol{x}}^{\text{Improvement}}$ can be computed:

$$\rho_i = \frac{|\overline{\boldsymbol{x}}_i^{\text{Improvement}} - \breve{\mu}_i|}{\breve{\sigma}_i} \tag{8}$$

where $\breve{\mu}_i$ and $\breve{\sigma}_i$ are computed from $\overline{\boldsymbol{x}}_i^{\text{Improvement}}$.

To complete the trigger, we must make a decision based upon all ρ_i. To this end, we decide to trigger the further enlargement of the variance multiplier if the ratio in any direction is larger than the threshold. In other words, if there is any search direction that requires scaling (i.e. slope traversing), AVS is triggered. This is identical to computing a single ρ as the maximum of the ρ_i and comparing this value to θ^{SDR}:

$$\rho = \max_{i=0}^{l-1} \{\rho_i\} \tag{9}$$

An intuitive way to see why SDR leads to an improved efficiency is the following. In any generation, given an estimation of the mean, there is an optimal covariance matrix $\mathbf{\Sigma}^*$ to use for sampling. Optimality here can be taken to be the highest probability of drawing a solution that is better compared to what has been encountered so far. Obviously, if the search is on a slope and the optimum is not enclosed within the region of the currently available solutions, $\mathbf{\Sigma}^*$ corresponds to a wider distribution than the estimated $\mathbf{\Sigma}$ and hence, for the optimal variance multiplier we have $c^{\text{AVS},*} > 1$. The AVS scheme increases the variance multiplier whenever an improvement is found. At first, especially in the slope–case, this means that the actual variance multiplier becomes closer to $c^{\text{AVS},*}$. However, AVS will then continue to increase the variance multiplier to a value of $c^{\text{AVS},+}$ which corresponds to a very small probability of finding an improvement. Thus, the AVS scheme will result in varying c^{AVS} roughly in the interval of $[c^{\text{AVS},*}, c^{\text{AVS},+}]$. Note that for large values of the variance multiplier, improvements are found relatively close to the mean in terms of standard–deviation ratio. This is exactly the case when the SDR trigger indicates that no further upscaling should be applied to the variance multiplier, preventing the variance multiplier from become excessively large. Thus, the SDR–AVS scheme will result in varying c^{AVS} in an interval that is more centered around $c^{\text{AVS},*}$, resulting in more efficient optimization.

3.3 EDA Performance

The combined addition of AVS and SDR to the use of ML estimates led to improved results in terms of a better scaling in the number of required evaluations to obtain a solution of a certain quality on various benchmarks [14, 5, 6]. In this article we follow the suggested settings found in that same literature for the parameters of the AVS and SDR mechanisms. Specifically, for AVS we use $\eta^{\text{Decrease}} = 0.9$, i.e. a small factor to allow

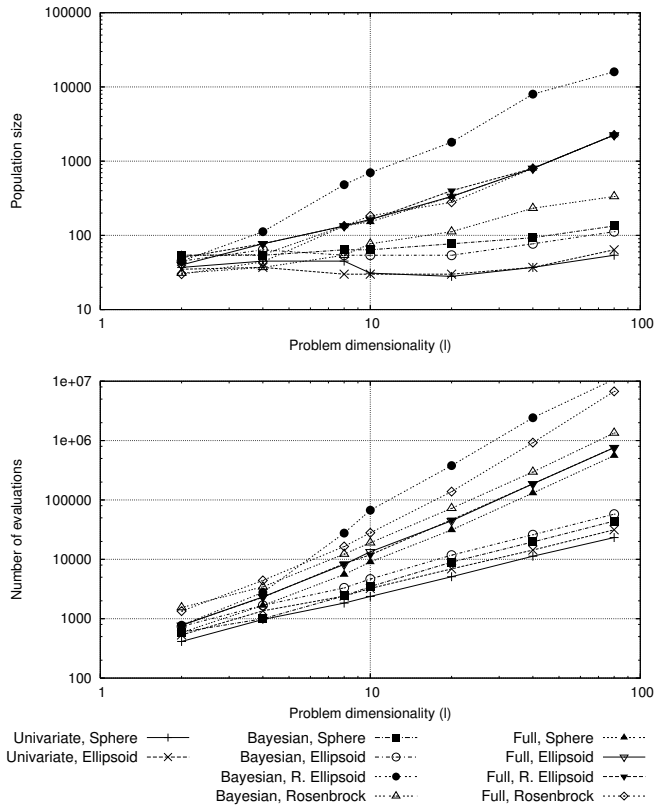

Figure 5: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-7.5, 7.5]$ for each variable (symmetric initialization). Maximum–likelihood estimates are used in combination with AVS and SDR.

for smooth adaptation of the variance multiplier. For SDR we use $\theta^{\text{SDR}} = 1.0$, i.e. if improvements occur outside the $\approx 68\%$ region of all density symmetrically surrounding the mean of the normal distribution, the AVS mechanism is allowed to increase the variance multiplier.

3.3.1 Symmetric initialization

Reconsider the Sphere, Ellipsoid and rotated Ellipsoid function introduced in Section 2. The results when AVS and SDR are used in addition to ML estimates is shown in Figure 5 for the case of symmetric initialization. Results on the rotated Ellipsoid were not reported before in the literature as also wasn't the combination of AVS and SDR with the univariate factorization and the full covariance matrix on any problem.

Overall it stands out that the optimal population size is much smaller than if only ML estimates are used although the number of required function evaluations is rather similar. The reason for this is that for population sizes for which the use of ML estimates lead to premature convergence, SDR–AVS scales the covariance matrix such that premature convergence doesn't occur. Hence, a part of the population size that is required only to ensure the EDA doesn't converge prematurely is now replaced by the SDR–AVS technique.

Further, the univariate factorization again requires the smallest population size. Also, due the mismatch between model an function landscape, use of the univariate factorization does not allow for Rosenbrock's function or the rotated Ellipsoid function to be optimized. It is for the same reason that the Bayesian factorization still cannot optimize the rotated Ellipsoid efficiently.

The relatively simple SDR–AVS extension is able to prevent the variance from monotonously shrinking while on a slope, i.e. in a far–away from optimum situation. Indeed, Rosenbrock's function can now be solved using either the Bayesian factorization or the full covariance matrix (see Figures 5 and 6). An illustration on the one–dimensional slope function from Section 2 is given in Figure 4. At first the variance shrinks exponentially fast as the center of the improvements is not outside a single standard deviation. Because the search is initialized uniformly over a range, it takes a few generations before the estimated distribution converges enough to the edge of this range to obtain a significant probability of generating better solutions than the ones already available. After 7 generations, this moment has been reached. The variance normally shrinks at a rate of $c(\tau)$ each generation. To overcome the shrinking rate, it is thus required that $c^{\text{AVS}} > 1/c(\tau)$. Initially, $c^{\text{AVS}} = 1$ and it is multiplied by η^{Increase} each improving generation. Hence, it takes at least $\lceil \log(1/c(\tau))/\log(\eta^{\text{Increase}}) \rceil$ generations before the variance is stopped from shrinking further. In our example, we find this to be at least 13 generations. Indeed, after a total of $7 + 13 = 20$ generations, the variance no longer decreases in Figure 4 and it steadily starts to increase, resulting in the mean to start an exponential shift towards $-\infty$. The addition of SDR and AVS thereby extend the class of problems that can be solved without deprecation of earlier results.

3.3.2 Asymmetric initialization

Now reconsider the Sphere, Ellipsoid and rotated Ellipsoid functions introduced in Section 2. The results when AVS and SDR are used in addition to ML estimates is shown in Figure 6 for the case of asymmetric initialization.

At first glance there is virtually no difference with the results of symmetric initialization. The technique of SDR–AVS thus appears to overcome all problems of premature convergence. Observing the results from shifting the initialization range from symmetric to beyond asymmetric in Figure 3 confirms that indeed with SDR–AVS the EDA is able to find the optimum even if the optimum is not contained in the initialization range. There is however a rather marked phase transition. When the initialization shift is such that the optimum moves to the edge of the initialization range, the SDR–AVS mechanism starts to become necessary. Before that it is hardly required. Indeed, the optimal population size then shrinks even further. The number of required evaluations however grow quite a lot more than when the optimum is inside the range or outside the range. This phase transition is due to the overtaking of the SDR–AVS mechanism. Moving the optimum outside of the initialization range sometimes triggers AVS and sometimes it doesn't. Without the scaling, the number of required function evaluations increases drastically. With the scaling it doesn't increase much. As the range shift becomes larger, AVS is triggered more often and the increase in the required function evaluations is reduced. It is this tradeoff on the boundary that is reflected in Figure 3.

3.3.3 Far–away initialization

In Figure 7 the scalability results are shown for the far–away initialization range. What stands out is the marked increase in function evaluations for the case in which the full covariance ma-

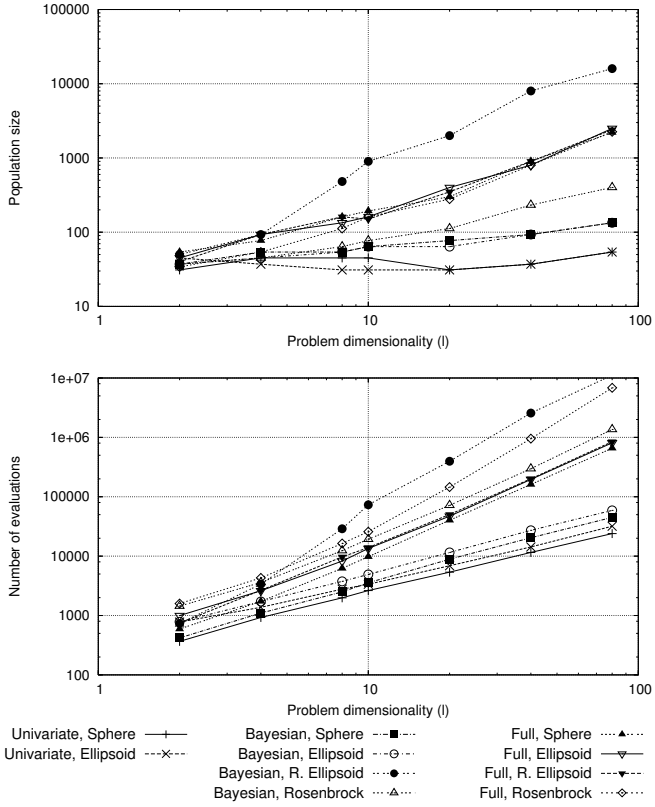

Figure 6: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-10, 5]$ for each variable (asymmetric initialization). Maximum–likelihood estimates are used in combination with AVS and SDR.

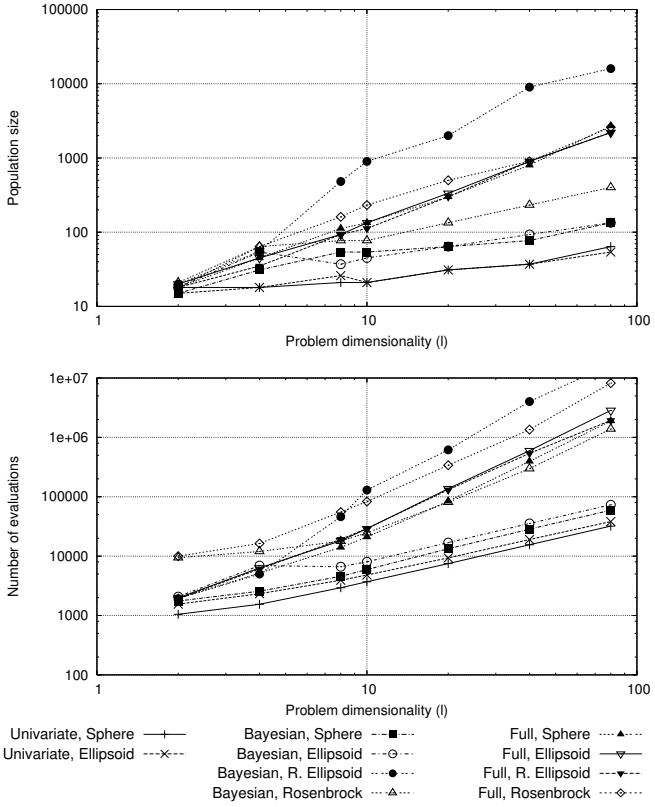

Figure 7: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-115, -100]$ for each variable (far–away initialization). Maximum–likelihood estimates are used in combination with AVS and SDR.

trix is used. For the unfactorized case and the Bayesian factorization case the required number of function evaluations doesn't increase by a similar factor for functions to which it is applicable (i.e. Sphere and Ellipsoid for the univariate factorization and the Bayesian factorization and Rosenbrock only for the Bayesian factorization). For the rotated Ellipsoid function, the Bayesian factorization attempts to incorporate as many dependencies as possible, which makes its application similar to the use of the full covariance matrix. It is for this function that an marked increase in required function evaluations can be seen also. Although the results indicate again that SDR–AVS indeed solves the problem of premature convergence, the uneven increase in number of required function evaluations indicates that there may still be a source of inefficiency that comes into play when the full covariance matrix is used that is not tackled effectively by SDR–AVS. There is also an additional increase in required population size for Rosenbrock's function. The reason for this is that Rosenbrock's function is not unimodal. We return to this in Section 4.5.3.

As already mentioned in Section 2.2.2, the deterioration of the results when using the full covariance matrix with ML estimates is worse than is to be expected by the mere shift of the initialization range when going from symmetric to asymmetric initialization. The underlying reason is the same as the reason why the use of the full covariance matrix leads to a higher increase in number of required function evaluations when moving to the far–away initialization range when using SDR–AVS. Very recently this reason was first noted in the literature [30].

An illustrative explanation of the source of the inefficiency at hand can be given using the extension of the linear slope function to two dimensions, i.e. $f(\boldsymbol{x}) = x_0 + x_1$. A direction $\boldsymbol{u}$ of steepest descent is one where $u_0 = u_1$ and $u_i \leq 0$. In the case of a unit direction, this means $\hat{\boldsymbol{u}} = (-\frac{1}{2}\sqrt{2}, -\frac{1}{2}\sqrt{2})$. Moving $\boldsymbol{x}$ by a distance of 1 in that direction reduces fitness by a value of $\sqrt{2}$. Thus, the best choice is to have the ellipsoids that correspond to the contour lines of the density of the estimated normal distribution, parallel to and elongated along the line $x_0 = x_1$ Conversely, the worst alignment would be parallel to and elongated along $x_0 = -x_1$ as along that line there is no improvement to be found in fitness.

In Figure 8 the contours of the estimated probability distribution are shown in the case of the full covariance matrix for the first six subsequent generations on the two–dimensional slope function. A population size of 100 is used. The density contours shown are the 95% error ellipses. Similar to the one–dimensional case, the normal distribution quickly contracts using ML estimates and the search doesn't move far outside the initialization range. There is an important additional observation to be made. Initially, the population is spread uniformly in the initialization square $[-5, 5]\times[-5, 5]$. On a two–dimensional slope the selected solutions will be located mostly in a triangle in the lower–left corner of the initialization square (see also Figure 8). Fitting a normal distribution with ML results in an orientation of the normal kernel more or less perpendicular to the direction of steepest descent. Most of the search is therefore devoted to searching in the least interesting direction. As generations pass, the diversity in fitness is lost further and the solutions in the population line up more and more with the worst alignment of $x_0 = -x_1$. Scaling the covariance matrix obtained with ML estimates in this situation will mostly enhance the search in the futile direction perpendicular to the direction of steepest descent. In order to still be able to obtain improvements in the direction of steepest

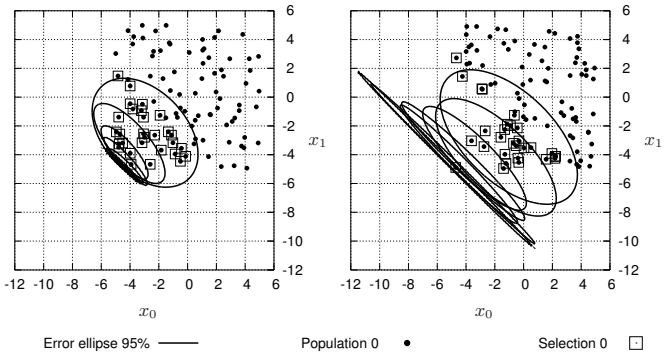

Figure 8: Progression of the normal distribution in the first 6 generations for typical runs of an EDA with ML estimates (*left*) and an EDA with SDR–AVS adaptation of the ML estimates (*right)* on the two–dimensional slope function $f(\boldsymbol{x}) = x_0 + x_1$ with initialization range $[-5, 5]$. The density contours shown are the 95% error ellipses. Also shown are the population and selection in generation 0.

descent, the variance multiplier has to grow to extremely large values.

It was noted in [30] that premature convergence still occurs with AVS. However, in that study, the original AVS suggestions were used in which the value of the variance multiplier is bounded. In a later suggestion (see [6]) and also in this article, there is no upper bound on the variance multiplier. Without the upper bound, the multiplier can become large enough to prevent premature convergence. Still, without rotation of the density toward the direction of descent, the sampling becomes less efficient because many samples are tried far away from the mean but in the wrong direction. Also, this may not be a problem in unconstrained search spaces, but it may be a problem if the search space is constrained. In a bounded search space, the newly sampled solutions are likely to become infeasible for large values of the variance multiplier. The resulting rejection sampling behavior may well lead to a very inefficient construction of new solutions. It is this important to tackle this inefficiency.

As a result of this behavior, optimization of the relatively simple unimodal functions used earlier, i.e. Sphere and Ellipsoid, deteriorates as one moves from symmetric to asymmetric initialization when ML estimates are used. The reason why asymmetric initialization doesn't completely prohibit the EDA from finding the optimum is that given a large enough population, there are enough solutions in the first selection to re-center the search around the optimum. The required number of solutions to this end however grows exponentially with the problem size. It is for this reason that we observe exponential scale–up behavior of the EDA if the full–covariance matrix is used together with asymmetric initialization with ML estimates (Figure 2).

4 Anticipated Mean Shift

Currently, only one EDA approach exists to reshape the density in more than a single dimension [30]. The approach employs minimization of cross–entropy in which both the selected solutions and the population are used. This way, a better matching between the density contours of the normal distribution and the fitness landscape can be obtained. Although first results are promising, the method is rather involved. Also, the resulting

scaling behavior was reported to be inferior to the use of SDR–AVS when symmetric initialization was used. Here, we provide a simple, yet elegant and intuitive alternative way to overcome the inefficiency at hand.

4.1 Definition of AMS

From Section 3 it can be concluded that although the capacity of the model (i.e. the normal distribution with a full–covariance matrix) is large enough to represent an arbitrarily orientated probabilistic search direction, the estimation procedure fails to match the model with the (local) problem structure. The ability of making such a match is essential to achieving efficient optimization [5]. Actually, it is the ability of EDAs to make this match that underlies their success. This ability is also known as exploiting problem structure. Repeatedly estimating a normal distribution with ML however only allows the exploitation of problem structure by means of contraction. In other words, the match between problem structure and the EDA search bias is only achieved if the optimum is located near the mean. By adaptively changing the scale of the covariance matrix we mainly influence the rate of contraction. To be able to exploit problem structure by traversing the search space, we must look toward adaptively changing the mean instead of the covariance matrix.

The difference in the estimated mean between two subsequent generations is an indication of a direction in which the selected solutions are moved to go from one level of fitness to a better level of fitness, assuming improving fitness with generations. Let $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$ denote for generation t the difference between the estimated means in subsequent generations $t-1$ and t, i.e. the mean shift:

$$\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) = \hat{\boldsymbol{\mu}}(t) - \hat{\boldsymbol{\mu}}(t-1) \tag{10}$$

A straightforward anticipation of the direction in which the mean will go in generation $t+1$ is $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$. It is therefore sensible to alter $100\alpha\%$ of all newly sampled solutions $\boldsymbol{x}$ in generation t by moving them a certain fraction δ in the direction of the previously observed the mean shift, i.e.:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \tag{11}$$

This is a very simple operation that is of asymptotically inferior complexity compared to the sampling of $\boldsymbol{x}$. We call this operation Anticipated Mean Shift (AMS).

When already centered over a peak, this this addition doesn't change the existing approach. In that case, $\hat{\boldsymbol{\mu}}(t) \approx \hat{\boldsymbol{\mu}}(t-1)$ and therefore $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \approx \mathbf{0}$, forcing the operation in Equation 11 to leave the originally sampled $\boldsymbol{x}$ unchanged.

When on a slope however, this simple addition can bring about an important change to the shape of the covariance matrix that is still estimated using only ML. In any generation after the first one, selection will choose solutions that basically belong to one of three sets: I) the previously selected solutions (i.e. the elitist solutions), II) the newly generated solutions *without* AMS and III) the newly generated solutions *with* AMS. Since set II is generated from a model that was estimated with ML from set I, these two sets share a similar region. Set III, although similar in shape to set II because of the estimated distribution, is further down the slope and thus pertains to a separate region. If selection now selects solutions from both regions, the shape of the normal distribution estimated in the next generation is automatically partly aligned with the direction of improvement. An illustration of this principle is given in Figure 9 (bottom row). Note that if the search is nearing a peak and AMS overshoots the optimum, the mean shift in the next generation will be much smaller because the mean shift will be caused again only by the non–anticipated solutions, basically resetting the approach. To make proper use of AMS, we still need to choose α and δ.

4.2 Number of adaptations (setting α)

On a slope, all of the $\alpha(1-\tau)n$ altered solutions will be better and therefore get selected. Now, if $\tau \geq \alpha$, this means that only the altered solutions will get selected. Because the solutions were sampled from a normal distribution that is equally shaped to the ML estimate, inefficient density shaping can still occur and the search direction will not get aligned with the direction in which fitness improves. On a slope, the distribution will be advanced down the slope, but the orientation of the distribution will not change. An illustration of this effect is given in Figure 9 (top row) where a value of $\alpha = 1$ is used (i.e. all newly generated solutions are adapted).

For the alignment of the search direction with the direction of fitness improvement to happen, we require the selected solutions to come from both the directly sampled solutions and the solutions altered for anticipation. Ideally, we would like these proportions to be equally sized. Again assuming we are on a slope, this means that we want $\alpha(1-\tau)n = \frac{1}{2}\tau n$, which gives:

$$\alpha = \frac{\tau}{2-2\tau} \tag{12}$$

In addition, we also want keep using the original model without anticipated mean shift to generate new solutions. As using information about the anticipated mean shift is still only predictive, we want to alter no more than 50% of the newly sampled solutions, i.e. $\alpha \leq 0.5$. Using Equation 12 this leads to a restriction on the selection percentile:

$$\alpha \leq 0.5 \quad \Leftrightarrow \quad \frac{\tau}{2-2\tau} \leq 0.5 \quad \Leftrightarrow \quad \tau \leq 0.5 \tag{13}$$

For the value of $\tau = 0.3$ that we use in our experiments throughout this article we thus use $\alpha \approx 0.214$, i.e. we adapt 21.4% of all newly generated solutions.

4.3 Adaptation length (setting δ)

On a slope, with α set as defined in the previous Section, set III (as defined in Section 4) in generation t will constitute 50% of the selected solutions in generation $t+1$. The other 50% will come from sets I and II. The mean of the latter two sets is $\hat{\boldsymbol{\mu}}(t)$. The mean of set III is $\hat{\boldsymbol{\mu}}(t) + \delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$. This means that, for the suggested value of α, the mean of the selected set in generation $t+1$ is[2]:

$$\hat{\boldsymbol{\mu}}(t+1) = \frac{\hat{\boldsymbol{\mu}}(t) + \hat{\boldsymbol{\mu}}(t) + \delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)}{2} = \hat{\boldsymbol{\mu}}(t) + \frac{\delta}{2}\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \tag{14}$$

The mean shift in generation $t+1$ then becomes:

$$\hat{\boldsymbol{\mu}}^{\text{Shift}}(t+1) = \hat{\boldsymbol{\mu}}(t+1) - \hat{\boldsymbol{\mu}}(t) = \frac{\delta}{2}\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \tag{15}$$

Thus, the size of the mean shift in generation $t+1$ is expected to be $\delta/2$ times the size of the mean shift in generation t. Hence,

[2] Note that equality actually only holds in the case of an infinite population size, it is an approximation otherwise.

for any $\delta < 2$ the AMS technique will not contribute much to advancing the mean as the next mean shift is expected to be smaller than the previous one. For too small values of δ, the anticipated solutions will not be far from the non–anticipated solutions. They will especially not be far from the non–anticipated solutions that will be selected in the next generation (i.e. the better ones). The closer the anticipated solutions are (i.e. the smaller δ is), the smaller the change in the shape of the estimated density. This can also be seen from the illustration in Figure 9 (center row) where $\delta = 1$ is used.

For a large enough value of δ, the anticipated solutions will be far enough away from the non–anticipated solutions. Because the newly estimated mean falls in between the two sets (given that α was set appropriately), the density contour will significantly change as an ML estimate captures not only the variance inside the two sets (which corresponds mainly to the previously estimated variance) but also the variance as a result of the distance between the two sets (which corresponds mainly to the direction of improvement). The latter component to the variance causes the density to be aligned more favorably to the direction of descent. As this process is repeated, the density with which the non–anticipated solutions are sampled starts to overlap more with that of the anticipated solutions. It is then no longer always true that all anticipated solutions are better than all other solutions. The re–aligned density can thereby in combination with selection result in a larger mean–shift than what Equation 15 indicates. For this reason, the conclusion drawn from Equation 15 of setting $\delta \geq 2$ is actually an upper bound on the minimal value to use for δ. The minimal value for δ for AMS to work as intended lies between 1 and 2. Indeed, in Figure 9 for $\delta = 2$ (top and bottom rows) the AMS technique results in an increase in mean shift in subsequent generations and the orientation of the distribution changes in combination with the correct value for α (bottom row only). Further change in the orientation of the density contours in subsequent generations is shown in Figure 10. This effect can also be observed in one dimension in Figure 4 where on the one–dimensional slope function the ML–estimated variance increases if AMS is used with $\alpha = \tau/(2 - 2\tau)$ and $\delta = 2$. Hence, we conclude by suggesting to set:

$$\delta = 2 \tag{16}$$

4.4 Combination with AVS: AMaLGaM

If the search is traversing a slope, it makes sense to accelerate the search. The AVS scheme actually already provides a principled way to achieve this. In the AVS scheme, a variance multiplier enlarges the variance if improvements far away from the mean occur in subsequent generations. The direct relation to subsequent improvements allow the multiplier to be seen as a general accelerator. We therefore rename the variance multiplier c^{AVS} to distribution multiplier and denote it by $c^{\text{Multiplier}}$ instead. Not only do we use $c^{\text{Multiplier}}\hat{\boldsymbol{\Sigma}}$ instead of $\hat{\boldsymbol{\Sigma}}$ upon sampling the distribution, we also use

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + c^{\text{Multiplier}}\delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \tag{17}$$

instead of $\boldsymbol{x} \leftarrow \boldsymbol{x}+\delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$ upon applying AMS. In other words, we accelerate the descent down the slope through multiplication with the distribution multiplier. In Figure 10 the effect of combining AVS with AMS can be seen when traversing the slope in two dimensions. The distribution gets rotated and elongated along the direction of improvement much faster than without the use of the distribution multiplier. On the one–dimensional slope the effect on the variance is also markedly different (see Figure 4). Whereas with the use of AMS the variance immediately increases, it continues to increase exponentially, ultimately being overtaken by the AVS scheme. Combining AVS and AMS results in an exponentially accelerating variance that starts immediately.

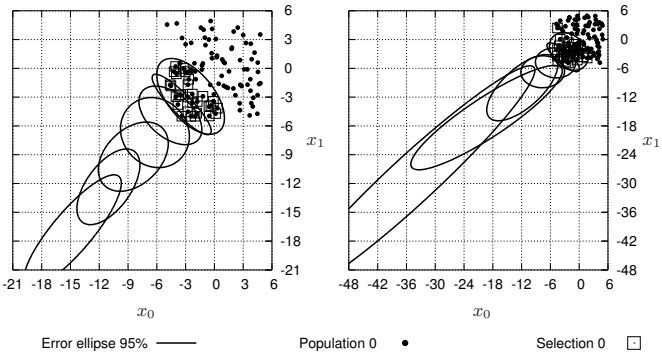

Figure 10: Progression of the normal distribution in the first 6 generations for typical runs of an EDA with AMS (*left*) and with AMaLGaM (*right*) adaptation of the ML estimates on the two–dimensional slope function $f(\boldsymbol{x}) = x_0 + x_1$ with initialization range $[-5, 5]$. The density contours shown are the 95% error ellipses. Also shown are the population and selection in generation 0.

The combination of SDR, AVS and AMS contains a mixture of important ingredients that adaptively changes both the covariance and the mean–shift to prevent inefficient sampling due to fitting the set of selected solutions without considering the direction of descent. We rename this composite AMS–SDR–AVS technique AMaLGaM (Adapted Maximum–Likelihood Gaussian Model). Pseudo–code for the generational loop of the integration of AMaLGaM in the IDEA, resulting in an EDA we call AMaLGaM–IDEA, is given in Figure 11. Source code can be downloaded from the website of the first author.

4.5 EDA Performance

4.5.1 Symmetric initialization

As we have pointed out in Section 2.2.1, running tests with symmetric initialization strongly biases methods of contraction. Also, the AMS technique doesn't contribute if the mean doesn't shift. In the case of symmetric initialization, mean shifts are negligible. We therefore do not run tests with symmetric initialization anymore and move on directly to asymmetric initialization.

4.5.2 Asymmetric initialization

We reconsider the Sphere, Ellipsoid and rotated Ellipsoid function introduced in Section 2 and the Rosenbrock function introduced in Section 2.2.3. The results when AMaLGaM is used to minimize these functions is shown in Figure 12 for the case of asymmetric initialization.

There is a clear reduction in number of required function evaluations compared to when SDR–AVS is used alone. The reduction is not only present for the full covariance matrix variant,

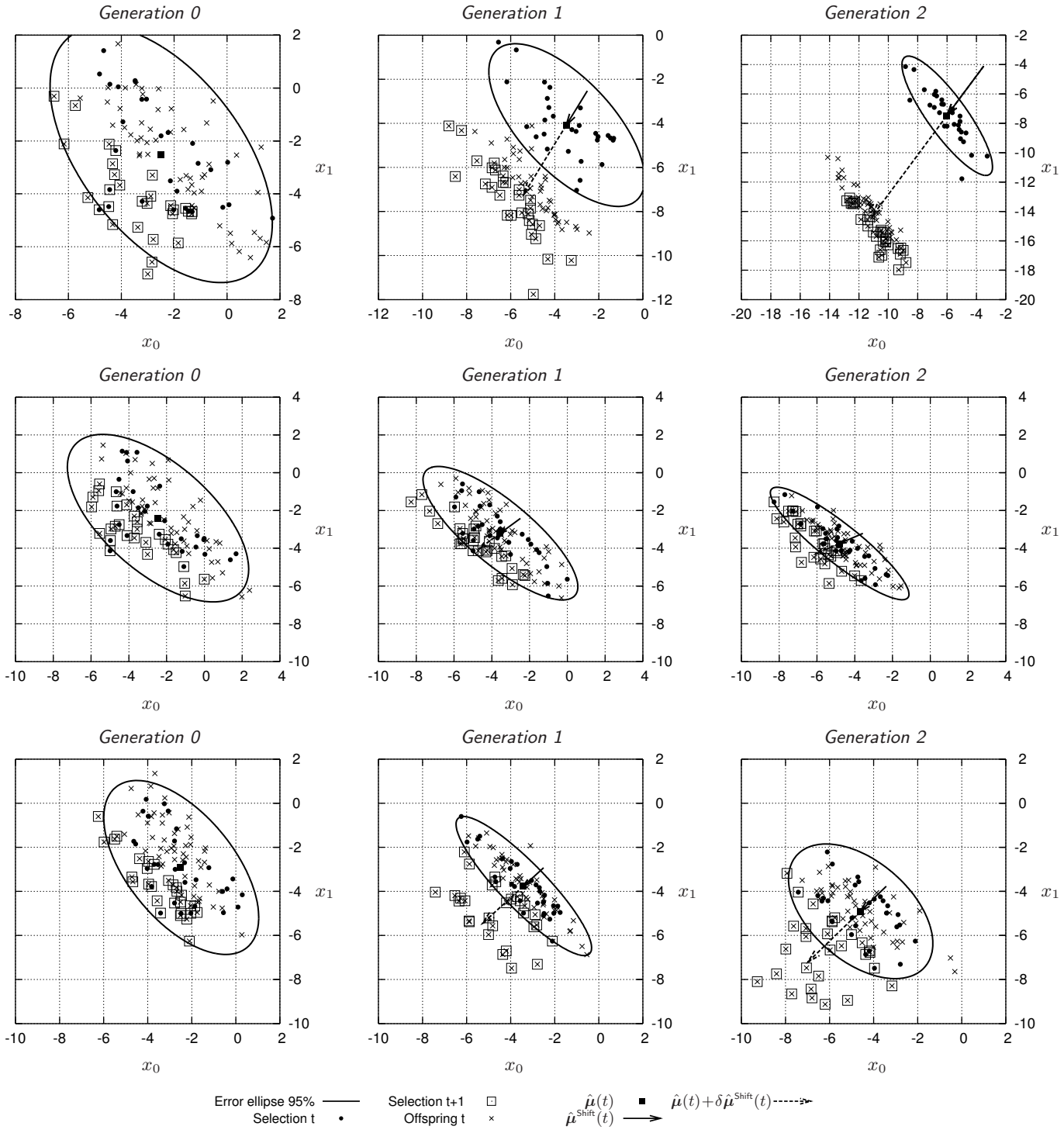

Figure 9: Progression of the normal distribution in the first 3 generations for typical runs of an EDA with AMS on the two–dimensional slope function $f(\boldsymbol{x}) = x_0 + x_1$ with initialization range $[-5, 5]$. The density contours shown are the 95% error ellipses. Also shown are the selection in the current and next generation, the offspring of the current generation, the current estimated mean, the mean shift and the anticipated mean shift. In the top row $\alpha = 1$ and $\delta = 2$. In the center row, $\alpha = 0.214$ and $\delta = 1$. In the bottom row, $\alpha = 0.214$ and $\delta = 2$.

```
 1  τ ← 0.3
 2  θ^SDR ← 1
 3  η^Decrease ← 0.9
 4  η^Increase ← 1/η^Decrease
 5  α ← τ/(2 − 2τ)
 6  δ ← 2
 7  t ← 0
 8  c^Multiplier ← 1
 9  P ← GenerateInitialPopulation(n)
10  for i ← 0 to n − 1 do
10.1   F_i^P ← f(P_i)
11  while ¬TerminationConditionSatisfied() do
11.01  (S, F^S) ← TruncationSelection(τ, P, F^P)
11.02  (π, Σ̂, μ̂) ← EstimateDistributionML(S)
11.03  Σ̂ ← c^Multiplier Σ̂
11.04  for i ← 0 to (n − ⌊τn⌋) − 1 do
11.04.1   O_i ← SampleNewSolution(π, Σ̂, μ̂)
11.05  if t > 0 then
11.05.1   μ̂^Shift ← μ̂ − μ̂^Previous
11.05.2   for i ← 0 to ⌊α(n − ⌊τn⌋)⌋ − 1 do
11.05.2.1    O_i ← O_i + c^Multiplier δ μ̂^Shift
11.06  F^Best ← min_{i=0}^{⌊τn⌋−1} {F_i^S}
11.07  n^Improvement ← 0
11.08  x̄^Improvement ← (0, 0, ..., 0)
11.09  for i ← 0 to (n − ⌊τn⌋) − 1 do
11.09.1   F_i^O ← f(O_i)
11.09.2   if F_i^O < F^Best then
11.09.2.1    n^Improvement ← n^Improvement + 1
11.09.2.2    x̄^Improvement ← x̄^Improvement + O_i
11.10  if n^Improvement > 0 then
11.10.1   x̄^Improvement ← x̄^Improvement / n^Improvement
11.10.2   ρ ← max_{i=0}^{l−1} {|x̄_i^Improvement(t) − μ̆_i(t)| / σ̆_i}
11.10.3   if ρ > θ^SDR then
11.10.3.1    c^Multiplier ← η^Increase c^Multiplier
          else
11.10.1   c^Multiplier ← η^Decrease c^Multiplier
11.11  if c^Multiplier < 1 then
11.11.2   c^Multiplier ← 1
11.12  P ← (S, O)
11.13  F^P ← (F^S, F^O)
11.14  μ̂^Previous ← μ̂
11.15  t ← t + 1
```

Figure 11: Pseudo–code for minimization of an l–dimensional function with AMaLGaM–IDEA using a population size of n. Specific parameter settings used and motivated throughout this article are also given (lines 1–7). Without shaded lines, the above reverts to an EDA with ML estimates. Light shading corresponds to AVS, medium shading corresponds to SDR, dark shading corresponds to AMS.

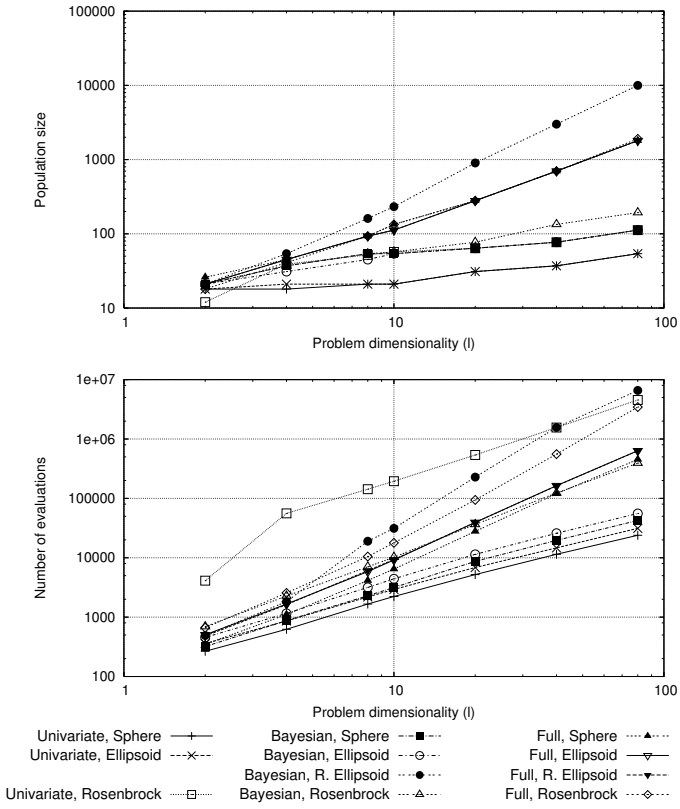

Figure 12: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) required to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-10, 5]$ for each variable (asymmetric initialization). AMaLGaM is used.

but for all variants, indicating that the AMS technique is a beneficial addition in general and does not only serve to re–align the ML–estimated density for more efficient sampling. There is also a further reduction in population size. This reduction is most apparent for the Bayesian factorization on the rotated ellipsoid function. Although it is still the case that many additional resources are required to learn most of the dependencies, the AMS technique clearly reduces the number of required resources. It appears that taking into account the mean shift reduces the necessity for having all dependencies correct in the model, which also becomes clear from the fact that the univariate distribution in combination with AMS is now able to optimize Rosenbrock's function. Rosenbrock's function has only limited dependencies between its problem variables. The rotated functions cannot be optimized using the univariate factorization because the rotations bring about too many dependencies. The AMS technique cannot compensate for the large discrepancy between the model used and the structure of the search space in that case. Only if the full covariance matrix is used are the results on the Ellipsoid function independent of rotation. AMaLGaM–IDEA with use of the full covariance matrix can thus be said to be robust against rotations of the search space.

Figure 3 hints that AMaLGaM–IDEA indeed effectively removes the inefficiency that SDR–AVS wasn't able to tackle. Shifting the initialization range has no effect on the optimal population size. The number of evaluations increases slightly for the simple reason that a larger trajectory is to be traveled to the optimum. The reason why there is no phase transition anymore is that the AMS technique doesn't require a trigger to set it off. It starts to work immediately and performs more work if the initialization range shifts more. For SDR–AVS this only holds if the initialization range is far enough away from the optimum. From an extrapolation of the results with the sliding initialization range, good results are expected if the initialization range moves far away.

4.5.3 Far–away initialization

Figure 13 shows the scalability results on the Sphere, Ellipsoid, rotated Ellipsoid and Rosenbrock functions for an initialization range that is far from the optimum. The similarity with the case of an asymmetric initialization range (Figure 12) is striking. With the exception of Rosenbrock's function, the optimal population size is not significantly affected. Moreover, the difference between the two results with respect to the number of function evaluations is only a small constant factor. The difference is not zero because a larger trajectory must be traveled through the search space. The constant difference factor between asymmetric initialization and far–away initialization is much smaller than in the case of SDR–AVS alone. Also, the difference is no longer bigger for the full covariance matrix variant than for any other variant. Hence, the addition of the AMS technique indeed allows the EDA to traverse slope–like regions of the search space much more efficiently due to an improved alignment of the probabilistic search direction with the direction of descent. AMaLGaM–IDEA can be said to be robust against translations of the search space.

Only the results on Rosenbrock's function are markedly different for the case of far–away initialization compared to the case of asymmetric initialization for dimensionalities between 2 and 20. The required population sizes are larger. The reason for this is that Rosenbrock's function is not unimodal [26]. With

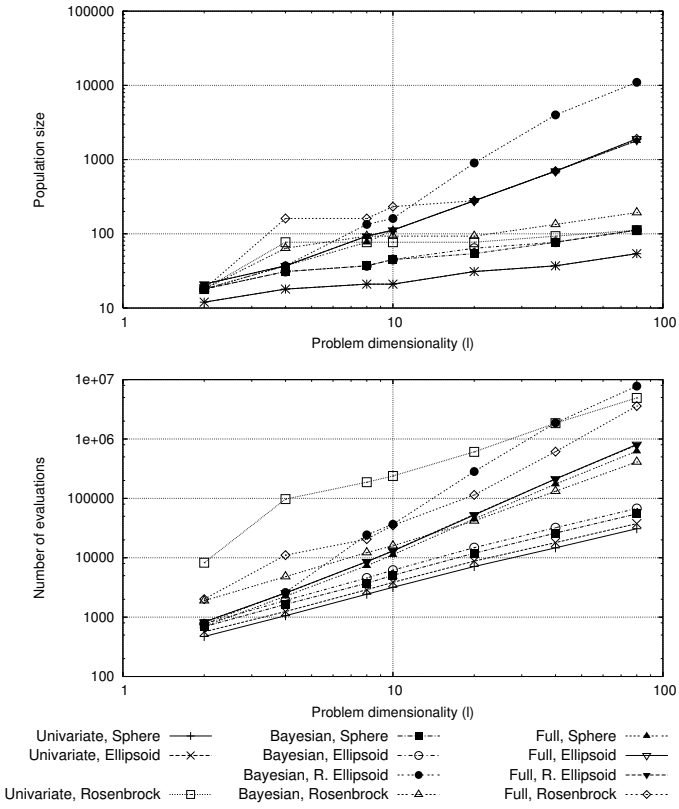

Figure 13: Scaling of the population size (*top*) that leads to the minimum number of evaluations (*bottom*) to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The initialization range is $[-115, -100]$ for each variable (far–away initialization). AMaLGaM is used.

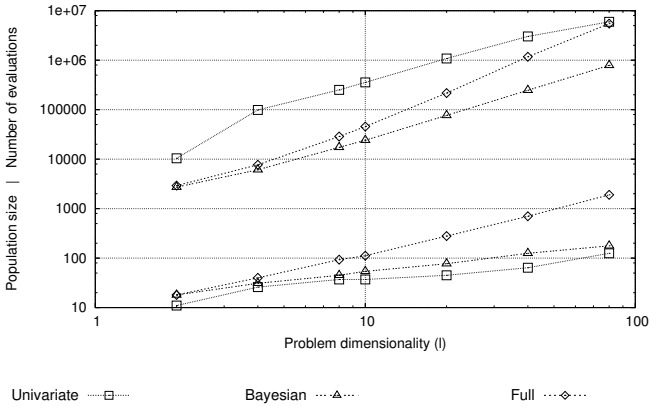

Figure 14: Scaling of the population size that leads to the minimum number of evaluations to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs on Rosenbrock's function. Both the population size and the number of evaluations are plotted. The population size is in the lower part of the graph. The initialization range is $[-115, -100]$ for each variable (far–away initialization). AMaLGaM is used where one run consists of the parallel synchronized execution of two independent EDAs.

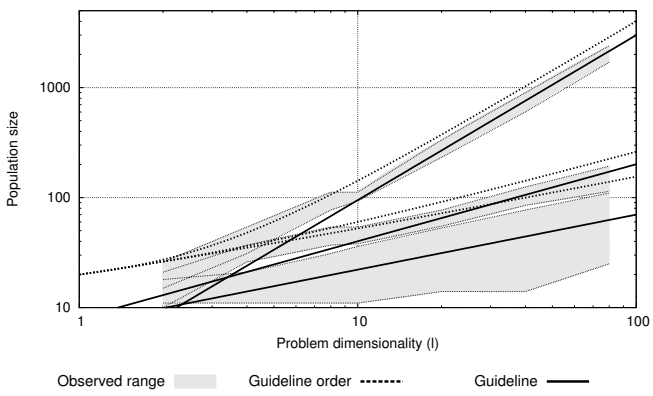

Figure 15: Observed and recommended population size that leads to the minimum number of evaluations for AMaLGaM to reach the VTR for an increasing problem dimensionality, averaged over 100 independent runs. The gray areas are the observed population sizes for all tested problems. The solid lines are the asymptotically recommendations. The thick dotted lines are the final recommendations. Results for the univariate factorization, the Bayesian factorization and the full covariance matrix are shown.

the proposed initialization range $[-115, -100]$ for each problem variable, the probability of finding the suboptimum is far larger than with symmetric or asymmetric initialization. Running the same EDA twice in parallel (i.e. with synchronized generations) and stopping whenever either of the runs reaches the VTR reduces the probability of not finding the optimum from q to q^2. Indeed, in Figure 14 it can be seen that in that case the results for the optimal population revert to similar results as found for the unimodal problems.

5 Guidelines and comparison with CMA–ES

In addition to in–depth theoretical and experimental investigations with respect to the algorithms themselves, it is important to compare with other algorithms. It is equally important to have some guidelines that can be used in subsequent applications and research of the algorithm. We therefore first present a set of practitioner's guidelines here. Subsequently, we use these guidelines to compare the performance of the AMaLGaM–IDEA with CMA–ES, currently the most efficient evolution strategy for continuous optimization.

5.1 Guidelines

To derive guidelines, we use the Sphere, Ellipsoid and Rosenbrock functions defined earlier. We also use the following set of 7 additional benchmark functions:

Name	Definition	VTR
Cigar	$x_0^2 + \sum_{i=1}^{l-1} 10^6 x_i^2$	10^{-10}
Tablet	$10^6 x_1^2 + \sum_{i=1}^{l-1} x_i^2$	10^{-10}
Cigar Tablet	$x_0^2 + \sum_{i=1}^{l-2} 10^4 x_i^2 + 10^8 x_{l-1}^2$	10^{-10}
Two Axes	$\sum_{i=0}^{\lfloor l/2 \rfloor - 1} 10^6 x_i^2 + \sum_{i=\lfloor l/2 \rfloor - 1}^{l-1} x_i^2$	10^{-10}
Different Powers	$\sum_{i=0}^{l-1} \lvert x_i \rvert^{2+10\frac{i}{l-1}}$	10^{-15}
Parabolic Ridge	$-x_1 + 100 \sum_{i=1}^{l-1} x_i^2$	-10^{10}
Sharp Ridge	$-x_1 + 100\sqrt{\sum_{i=1}^{l-1} x_i^2}$	-10^{10}

For each of these 10 functions, we determined the optimal population size for AMaLGaM–IDEA using the naive variant (i.e. univariate factorization), the learning variant (i.e. Bayesian factorization) and the full covariance matrix variant (i.e. unfactorized). Rosenbrock's function is not used for the naive variant. Moreover, for the full covariance matrix variant we used both the unrotated as well as the rotated version. Finally, we ran tests for the symmetric initialization range, the asymmetric initialization range and the far–away initialization range. We combined all resulting scalability plots and determined on the basis thereof a guideline for the population size to be used. These guidelines and the combined scalability plots are shown in Figure 15. We chose the population–sizing guidelines such that a minimal population size of 20 is always ensured. For other parameters we used the guidelines derived and mentioned throughout this paper and given in the first lines of the pseudo–code in Figure 11. The population–sizing guidelines we propose are:

- $n \geq 15 l^{0.5} + 5$ (naive, unfactorized case)
- $n \geq 10 l^{0.7} + 10$ (learning, Bayesian factorized case)
- $n \geq 4 l^{1.5} + 16$ (full covariance matrix, unfactorized case)

5.2 CMA–ES

The CMA–ES is an evolution strategy (ES) that aims to improve upon the design of the original ES [3] by derandomizing the pro-

cess that adapts the covariance matrix [16]. Different from the original ES there is only one covariance matrix associated with the population. This means that at the top level, a normal distribution is used each generation to sample a new population from, which is identical to the EDA approach taken in this article. In CMA–ES the mean is even taken to be the sample mean, identical to the ML approach used in EDAs. The main difference with the approach taken in this article lies in how the covariance matrix is determined. In CMA–ES, the estimate is not directly based on the population. Instead, an evolution path is convoluted over time that describes the direction in which the search has progressed in generations past. The covariance matrix is changed each generation to become more aligned with the evolution path. For details, we refer the interested reader to the relevant literature [16].

5.3 Comparison

We used the guidelines defined in Section 5.1 and ran AMaLGaM–IDEA 100 independent times on each of the benchmark problems, both for the asymmetric initialization range and the far–away initialization range. As mentioned earlier, the symmetric initialization range presents an undesirably big bias. For this reason, we have not included this case in our final experiments. For the parameter settings of the CMA–ES, we used the guidelines provided in the literature also [16].

5.3.1 Computational complexities

From the observed results, we computed a least–squares fit to $\alpha l^{\beta} + \gamma$ for both the number of required evaluations as well as the actual computing time to obtain insight into the computational complexity of the tested algorithms. The fit was always found to be highly accurate. The results are summarized in Figure 16 for the asymmetric initialization range and in Figure 17 for the far–away initialization range. For the evaluations, all regression parameters are presented. For the time, only the main complexity parameter (i.e. β) is presented because the actual number of seconds is machine–dependent.

5.3.2 Comparing naive, learning and full covariance matrix

In general, the naive variant has a smaller computational complexity than does the Bayesian variant, which in turn has a smaller computational complexity than does the variant that uses the full covariance matrix without factorization. With this statement it needs to be mentioned that this only holds for functions that fit the model used. The Bayesian method is not well–suited for rotated variants of the tested functions for instance. On those functions, the full method will outperform the Bayesian method. For the full method, the observed computational complexity is similar if the function is rotated. Hence, using the full covariance matrix in AMaLGaM–IDEA results in an EA that is robust to arbitrary rotations of the search space, a property of CMA–ES also.

In general, the performance in terms of computational complexity of AMaLGaM–IDEA does not change significantly when moving from asymmetric initialization to far–away initialization. This leads to the conclusion that AMaLGaM–IDEA is also robust to translations.

Function	Algorithm	β^{Time}	β^{Eval}	α^{Eval}	γ^{Eval}
Sphere	AMaLGaM–N	2.05	1.36	$1.27{\cdot}10^{2}$	$2.10{\cdot}10^{0}$
	AMaLGaM–L	3.21	1.48	$1.09{\cdot}10^{2}$	$4.01{\cdot}10^{1}$
	AMaLGaM–F	3.81	2.21	$4.16{\cdot}10^{1}$	$1.84{\cdot}10^{2}$
	CMA–ES	1.73	0.94	$1.85{\cdot}10^{2}$	$2.02{\cdot}10^{2}$
Ellipsoid	AMaLGaM–N	2.19	1.34	$1.83{\cdot}10^{2}$	$-1.87{\cdot}10^{1}$
	AMaLGaM–L	3.28	1.45	$1.66{\cdot}10^{2}$	$6.53{\cdot}10^{0}$
	AMaLGaM–F	3.71	2.17	$6.56{\cdot}10^{1}$	$2.18{\cdot}10^{2}$
	CMA–ES	3.76	1.96	$5.37{\cdot}10^{1}$	$1.62{\cdot}10^{3}$
Cigar	AMaLGaM–N	2.15	1.33	$2.11{\cdot}10^{2}$	$-8.70{\cdot}10^{1}$
	AMaLGaM–L	3.23	1.44	$1.89{\cdot}10^{2}$	$-3.19{\cdot}10^{1}$
	AMaLGaM–F	3.67	2.16	$7.45{\cdot}10^{1}$	$1.88{\cdot}10^{2}$
	CMA–ES	2.11	0.91	$6.35{\cdot}10^{2}$	$-1.97{\cdot}10^{2}$
Tablet	AMaLGaM–N	1.94	1.30	$1.65{\cdot}10^{2}$	$3.84{\cdot}10^{1}$
	AMaLGaM–L	3.29	1.42	$1.43{\cdot}10^{2}$	$8.20{\cdot}10^{1}$
	AMaLGaM–F	3.75	2.15	$5.60{\cdot}10^{1}$	$2.66{\cdot}10^{2}$
	CMA–ES	2.83	1.64	$1.14{\cdot}10^{2}$	$1.25{\cdot}10^{3}$
Cigar tablet	AMaLGaM–N	2.14	1.32	$1.98{\cdot}10^{2}$	$-3.31{\cdot}10^{0}$
	AMaLGaM–L	3.30	1.43	$1.82{\cdot}10^{2}$	$1.84{\cdot}10^{1}$
	AMaLGaM–F	3.66	2.15	$7.17{\cdot}10^{1}$	$2.50{\cdot}10^{2}$
	CMA–ES	2.75	1.40	$2.08{\cdot}10^{2}$	$1.20{\cdot}10^{3}$
Two axes	AMaLGaM–N	2.11	1.35	$1.80{\cdot}10^{2}$	$-1.39{\cdot}10^{1}$
	AMaLGaM–L	3.37	1.44	$1.77{\cdot}10^{2}$	$-2.56{\cdot}10^{1}$
	AMaLGaM–F	3.70	2.16	$7.18{\cdot}10^{1}$	$2.08{\cdot}10^{2}$
	CMA–ES	3.52	1.96	$8.79{\cdot}10^{1}$	$1.05{\cdot}10^{3}$
Different powers	AMaLGaM–N	2.44	1.45	$5.79{\cdot}10^{1}$	$6.64{\cdot}10^{1}$
	AMaLGaM–L	3.44	1.54	$5.32{\cdot}10^{1}$	$8.45{\cdot}10^{1}$
	AMaLGaM–F	3.99	2.26	$2.11{\cdot}10^{1}$	$1.62{\cdot}10^{2}$
	CMA–ES	3.24	1.76	$6.44{\cdot}10^{1}$	$9.31{\cdot}10^{2}$
Rosenbrock	AMaLGaM–N	2.09	1.44	$8.58{\cdot}10^{3}$	$-1.61{\cdot}10^{4}$
	AMaLGaM–L	3.46	1.75	$1.92{\cdot}10^{2}$	$4.69{\cdot}10^{1}$
	AMaLGaM–F	3.92	2.54	$5.70{\cdot}10^{1}$	$4.50{\cdot}10^{2}$
	CMA–ES	3.30	1.90	$7.50{\cdot}10^{1}$	$1.37{\cdot}10^{3}$
Parabolic ridge	AMaLGaM–N	1.96	1.07	$1.06{\cdot}10^{2}$	$2.63{\cdot}10^{2}$
	AMaLGaM–L	3.21	1.10	$2.60{\cdot}10^{2}$	$8.49{\cdot}10^{1}$
	AMaLGaM–F	3.51	2.01	$9.22{\cdot}10^{1}$	$3.36{\cdot}10^{2}$
	CMA–ES	2.18	0.98	$4.44{\cdot}10^{2}$	$5.41{\cdot}10^{1}$
Sharp ridge	AMaLGaM–N	1.49	0.97	$1.15{\cdot}10^{2}$	$2.61{\cdot}10^{2}$
	AMaLGaM–L	3.02	1.12	$1.02{\cdot}10^{2}$	$2.63{\cdot}10^{2}$
	AMaLGaM–F	3.57	1.88	$4.88{\cdot}10^{1}$	$3.38{\cdot}10^{2}$
	CMA–ES	1.99	0.86	$2.02{\cdot}10^{3}$	$-6.70{\cdot}10^{3}$

Figure 16: Regression coefficients for scalability on all benchmark problems averaged over 100 independent runs using the recommended settings for the population size. The initialization range is $[-10, 5]$ for each variable (asymmetric initialization).

Function	Algorithm	β^{Time}	β^{Eval}	α^{Eval}	γ^{Eval}
Sphere	AMaLGaM–N	2.00	1.23	$2.74{\cdot}10^{2}$	$9.46{\cdot}10^{0}$
	AMaLGaM–L	3.17	1.34	$2.46{\cdot}10^{2}$	$1.63{\cdot}10^{2}$
	AMaLGaM–F	3.56	2.05	$1.09{\cdot}10^{2}$	$4.08{\cdot}10^{2}$
	CMA–ES	1.68	0.94	$2.38{\cdot}10^{2}$	$3.17{\cdot}10^{2}$
Ellipsoid	AMaLGaM–N	2.03	1.24	$3.33{\cdot}10^{2}$	$8.20{\cdot}10^{0}$
	AMaLGaM–L	3.03	1.36	$2.90{\cdot}10^{2}$	$1.31{\cdot}10^{2}$
	AMaLGaM–F	3.52	2.09	$1.14{\cdot}10^{2}$	$4.87{\cdot}10^{2}$
	CMA–ES	3.33	1.92	$6.40{\cdot}10^{1}$	$1.79{\cdot}10^{3}$
Cigar	AMaLGaM–N	2.12	1.25	$3.40{\cdot}10^{2}$	$-2.30{\cdot}10^{1}$
	AMaLGaM–L	3.06	1.35	$3.20{\cdot}10^{2}$	$4.48{\cdot}10^{1}$
	AMaLGaM–F	3.71	2.08	$1.30{\cdot}10^{2}$	$4.14{\cdot}10^{2}$
	CMA–ES	2.10	0.90	$7.18{\cdot}10^{2}$	$-2.16{\cdot}10^{2}$
Tablet	AMaLGaM–N	1.98	1.22	$2.95{\cdot}10^{2}$	$1.12{\cdot}10^{2}$
	AMaLGaM–L	3.01	1.32	$2.77{\cdot}10^{2}$	$1.80{\cdot}10^{2}$
	AMaLGaM–F	3.51	2.04	$1.13{\cdot}10^{2}$	$5.24{\cdot}10^{2}$
	CMA–ES	2.86	1.64	$1.17{\cdot}10^{2}$	$1.59{\cdot}10^{3}$
Cigar tablet	AMaLGaM–N	2.06	1.22	$3.54{\cdot}10^{2}$	-4.14e-01
	AMaLGaM–L	3.03	1.34	$3.21{\cdot}10^{2}$	$8.52{\cdot}10^{1}$
	AMaLGaM–F	3.50	2.07	$1.23{\cdot}10^{2}$	$4.93{\cdot}10^{2}$
	CMA–ES	2.75	1.40	$2.16{\cdot}10^{2}$	$1.48{\cdot}10^{3}$
Two axes	AMaLGaM–N	2.05	1.27	$3.06{\cdot}10^{2}$	$4.62{\cdot}10^{1}$
	AMaLGaM–L	3.05	1.37	$2.86{\cdot}10^{2}$	$1.18{\cdot}10^{2}$
	AMaLGaM–F	3.54	2.10	$1.11{\cdot}10^{2}$	$5.08{\cdot}10^{2}$
	CMA–ES	3.60	2.00	$7.91{\cdot}10^{1}$	$1.68{\cdot}10^{3}$
Different powers	AMaLGaM–N	2.23	1.39	$1.49{\cdot}10^{2}$	$1.98{\cdot}10^{2}$
	AMaLGaM–L	3.29	1.41	$1.70{\cdot}10^{2}$	$1.94{\cdot}10^{2}$
	AMaLGaM–F	3.55	2.09	$7.75{\cdot}10^{1}$	$3.78{\cdot}10^{2}$
	CMA–ES	2.90	1.65	$1.55{\cdot}10^{2}$	$1.14{\cdot}10^{3}$
Rosenbrock	AMaLGaM–N	2.16	1.55	$5.94{\cdot}10^{3}$	$-7.59{\cdot}10^{3}$
	AMaLGaM–L	3.52	1.70	$2.42{\cdot}10^{2}$	$1.43{\cdot}10^{3}$
	AMaLGaM–F	3.78	2.57	$5.58{\cdot}10^{1}$	$2.35{\cdot}10^{3}$
	CMA–ES	3.32	1.92	$7.25{\cdot}10^{1}$	$2.52{\cdot}10^{3}$
Parabolic ridge	AMaLGaM–N	2.09	1.02	$2.00{\cdot}10^{2}$	$1.57{\cdot}10^{2}$
	AMaLGaM–L	3.21	1.13	$2.75{\cdot}10^{2}$	$1.14{\cdot}10^{2}$
	AMaLGaM–F	3.51	2.01	$1.06{\cdot}10^{2}$	$3.38{\cdot}10^{2}$
	CMA–ES	1.71	1.01	$4.29{\cdot}10^{2}$	$3.43{\cdot}10^{2}$
Sharp ridge	AMaLGaM–N	1.76	0.95	$1.70{\cdot}10^{2}$	$2.02{\cdot}10^{2}$
	AMaLGaM–L	3.11	1.08	$1.57{\cdot}10^{2}$	$2.20{\cdot}10^{2}$
	AMaLGaM–F	3.42	1.87	$7.33{\cdot}10^{1}$	$3.35{\cdot}10^{2}$
	CMA–ES	1.65	0.78	$2.80{\cdot}10^{3}$	$-9.00{\cdot}10^{3}$

Figure 17: Regression coefficients for scalability on all benchmark problems averaged over 100 independent runs using the recommended settings for the population size. The initialization range is $[-115, -100]$ for each variable (far–away initialization).

5.3.3 Comparing AMaLGaM and CMA

The computational complexity of CMA–ES ranges between the results obtained by the different variants of AMaLGaM–IDEA. For some functions (e.g. Sphere), CMA–ES has a better computational complexity than even the naive variant of AMaLGaM–IDEA. For other functions (e.g. Two axes) it has a computational complexity equal to or worse than AMaLGaM–IDEA (i.e. asymmetric initialization and far–away initialization respectively). The computational complexity results of AMaLGaM–IDEA are less variable. As a result, CMA–ES is better on some functions whereas AMaLGaM–IDEA is better on other functions. Overall, it can be concluded that AMaLGaM–IDEA and CMA–ES are competitive, but that CMA–ES has the upper hand in the comparison. This is especially true if we desire the property of rotation invariance. For AMaLGaM–IDEA this property is only obtained if the full covariance matrix is used. For that variant however, AMaLGaM–IDEA does not outperform CMA–ES on any function, but results at most in a similar computational complexity (e.g. Ellipsoid, Two axes).

6 Discussion and Outlook

The results and analysis in this article signal an important shortcoming of early testing. If a symmetric initialization range is used, an undesirably big bias is provided to methods of contraction. Many recombination operators fit this description for continuous search spaces. The results in this article show however that it is not enough to only use asymmetric initialization. For a better understanding of the search dynamics, far–away initialization should be used also. In addition, as was already acknowledged by research on the CMA–ES, it is important to also rotate the search space. Rotation may introduce strong dependencies between the problem variables. Given a new, unseen black–box optimization problem we cannot assume independence between the problem variables. We believe translation of initialization ranges and rotation of test functions to be important to take into account in subsequent EA research for numerical optimization in general.

The applicability of the approaches studied in this article (AMaLGaM–IDEA and CMA–ES) in practice is not independent from the size of the problem to be optimized. From the computational complexity results the applicability of methods that use the full covariance matrix (i.e. AMaLGaM–IDEA with the full covariance matrix and CMA–ES) it follows that these methods are likely not very useful if the problem has many parameters. The required computing time increases quickly with the number of problem variables (roughly cubic or worse). This leaves only methods that take into account only a few dependencies (e.g. allow only one or two parents for each problem variable in the Bayesian factorization) or no dependencies at all (i.e. the naive AMaLGaM–IDEA). Their time scalability is roughly only quadratic. Certainly, if there are many strong dependencies in the problem, the algorithm shall not be able to find the optimum. Still, due to its simplicity, speed, and effectiveness the naive AMaLGaM–IDEA can well serve as a baseline EDA to be used for future comparison with other EAs and for practical applications with many variables.

The efficiency of AMaLGaM–IDEA has substantially improved compared to previous EDAs, especially in the case of using a full covariance matrix. The search dynamics of AMaLGaM–IDEA now share many properties with CMA–ES. The most notable

new similarity is the rotation of the probability distribution to become more aligned with the direction of descent. One of the main differences is that whereas AMaLGaM–IDEA re–estimates the density from the selected solutions each generation anew and only adapts the variance multiplier over multiple generations, the CMA–ES adapts the covariance matrix entirely over multiple generations. Although this might make AMaLGaM–IDEA better suited for dynamically changing functions than CMA–ES, it generally means that AMaLGaM–IDEA also requires more function evaluations. Each generation there need to be enough solutions to support a proper ML estimate. As CMA–ES convolutes its covariance matrix over multiple generations, it can greatly reduce the required population size. We therefore believe that a promising future research direction is to reduce the required population size for AMaLGaM–IDEA. This will directly lead to less required function evaluations. To this end is is interesting to study the optimal values for the parameters of the normal distribution. One interesting result in recent research shows what the optimal value for the variance is, given a certain value for the mean and a certain function [13]. Since this value is independent of the population, this line of research may yet provide important new insights, especially when comparing these results to the use of ML estimates.

7 Conclusion

In this article, we have focused on the use of the normal distribution in EDAs. Specifically, we have focused on the use of maximum–likelihood (ML) estimates for setting the parameters of the normal distribution. Using ML estimates, the EDA risks premature convergence and can only perform optimization properly if the initialization range nicely brackets the optimum. Optimization then mainly proceeds through contraction. Methods of adaptive variance scaling (AVS) provide a way to control the rate of contraction and even turn it into expansion. As a result, the EDA can look well beyond the region of the selected solutions and is it no longer prone to premature convergence. Because ML estimates of the normal distribution attempt to shape the density similar to the configuration of the selected solutions, the density can however be orientated such that it is not aligned with the direction of descent in the fitness landscape. The variance then needs to be scaled to excessively large values to still make progress along the direction of descent. We have proposed a simple, yet effective way called anticipated mean shift (AMS) that removes this inefficiency. AMS proceeds by not only adaptively scaling the variance of the ML estimate, but also adaptively changing the mean to also draw some solutions down the line of descent instead of only surrounding the estimated mean. We analyzed this technique and provided rational settings its parameters. We called the EDA with ML estimates and adaptive changes thereof Adapted Maximum–Likelihood Gaussian Model — Iterated Density–Estimation Evolutionary Algorithm (AMaLGaM–IDEA). By means of an experimental scalability analysis we showed that the AMaLGaM–IDEA is competitive with CMA–ES and that AMaLGaM–IDEA is robust to rotations and translations of the search space.

The research in this article has thus shown that ML estimates of the normal distribution are not only likely to lead the EDA to premature convergence, efficiently removing this limitation requires online adaptation of *both* the estimated covariance matrix and the estimated mean. The techniques used in AMaLGaM–IDEA do exactly this in a simple, but principled and effective manner. AMaLGaM–IDEA therefore makes an important step in the progression of understanding and improving continuous EDAs for numerical optimization.

References

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. Dover Publications, New York, New York, 1972.

[2] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons Inc., New York, New York, 1958.

[3] Th. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[4] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 38–46, Madison, Wisconsin, 1995. Morgan Kauffman.

[5] P. A. N. Bosman and J. Grahl. Matching inductive search bias and problem structure in continuous estimation–of–distribution algorithms. *European Journal of Operational Research*, 185(3):1246–1264, 2008.

[6] P. A. N. Bosman, J. Grahl, and F. Rothlauf. SDR: A better trigger for adaptive variance scaling in normal EDAs. In D. Thierens et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2007*, pages 492–499, New York, New York, 2007. ACM Press.

[7] P. A. N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In M. Pelikan et al., editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference — GECCO–2000*, pages 197–200, San Francisco, California, 2000. Morgan Kaufmann.

[8] P. A. N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In M. Schoenauer et al., editors, *Parallel Problem Solving from Nature — PPSN VI*, pages 767–776, Berlin, 2000. Springer–Verlag.

[9] P. A. N. Bosman and D. Thierens. Advancing continuous IDEAs with mixture distributions and factorization selection metrics. In M. Pelikan and K. Sastry, editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference — GECCO–2001*, pages 208–212, San Francisco, California, 2001. Morgan Kaufmann.

[10] P. A. N. Bosman and D. Thierens. The naive MIDEA: a baseline multi–objective EA. In C. A. Coello Coello et al., editors, *Evolutionary Multi–Criterion Optimization — EMO–2005*, pages 428–442, Berlin, 2005. Springer–Verlag.

[11] P. A. N. Bosman and D. Thierens. Numerical optimization with real–valued estimation–of–distribution algorithms. In M. Pelikan et al., editors, *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer–Verlag, Berlin, 2006.

[12] C. González, J. A. Lozano, and P. Larrañaga. Mathematical modelling of UMDAc algorithm with tournament selection. behaviour on linear and quadratic functions. *International Journal of Approximate Reasoning*, 31(3):313–340, 2002.

[13] J. Grahl, P. A. N. Bosman, and S. Minner. Convergence phases, variance trajectories, and runtime analysis of continuous EDAs. In D. Thierens et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2007*, pages 516–522, New York, New York, 2007. ACM Press.

[14] J. Grahl, P. A. N. Bosman, and F. Rothlauf. The correlation–triggered adaptive variance scaling IDEA. In M. Keijzer et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2006*, pages 397–404, New York, New York, 2006. ACM Press.

[15] J. Grahl, S. Minner, and F. Rothlauf. Behaviour of UMDAc with truncation selection on monotonous functions. In D. Corne et al., editors, *Proceedings of the IEEE Congress on Evolutionary Computation — CEC–2005*, pages 2553–2559, Piscataway, New Jersey, 2005. IEEE Press.

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

[17] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In M. Pelikan et al., editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference — GECCO–2000*, pages 201–204, San Francisco, California, 2000. Morgan Kaufmann.

[18] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic, London, 2001.

[19] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*. Springer–Verlag, Berlin, 2006.

[20] H. Mühlenbein and R. Höns. The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27, 2005.

[21] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. binary parameters. In A. E. Eiben et al., editors, *Parallel Problem Solving from Nature – PPSN V*, pages 178–187, Berlin, 1998. Springer–Verlag.

[22] J. Ocenasek, S. Kern, N. Hansen, S. Müller, and P. Koumoutsakos. A mixed bayesian optimization algorithm with variance adaptation. In X. Yao et al., editors, *Parallel Problem Solving from Nature — PPSN VIII*, pages 352–361, Berlin, 2004. Springer–Verlag.

[23] M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer–Verlag, Berlin, 2006.

[24] S. Rudlof and M. Köppen. Stochastic hill climbing with learning by vectors of normal distributions. In T. Furuhashi, editor, *Proceedings of the First Online Workshop on Soft Computing — WSC1*, pages 60–70, Nagoya, 1996. Nagoya University.

[25] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In A. E. Eiben et al., editors, *Parallel Problem Solving from Nature — PPSN V*, pages 418–427, Berlin, 1998. Springer–Verlag.

[26] Y.-W. Shang and Y.-H. Qiu. A note on the extended Rosenbrock function. *Evolutionary Computation*, 14(1):119–126, 2006.

[27] M. M. Tatsuoka. *Multivariate Analysis: Techniques for Educational and Psychological Research*. John Wiley & Sons Inc., New York, New York, 1971.

[28] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer–Verlag, Berlin, 1995.

[29] B. Yuan and M. Gallagher. A mathematical modelling technique for the analysis of the dynamics of a simple continuous EDA. In G. G. Yen et al., editors, *Proceedings of the IEEE Congress on Evolutionary Computation — CEC–2006*, pages 1585–1591, Piscataway, New Jersey, 2006. IEEE Press.

[30] C. Yunpeng, S. Xiaomin, X. Hua, and J. Peifa. Cross entropy and adaptive variance scaling in continuous EDA. In D. Thierens et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO–2007*, pages 609–616, New York, New York, 2007. ACM Press.