An Estimation of Distribution Algorithm based on the Natural Gradient and the Boltzmann Distribution

Ignacio Segovia-Domínguez
Center for Research in Mathematics
Guanajuato, México
ijsegoviad@cimat.mx

Arturo Hernández-Aguirre
Center for Research in Mathematics
Guanajuato, México
artha@cimat.mx

ABSTRACT

This paper introduces an Estimation of Distribution Algorithm (EDA), in which the parameters of the search distribution are updated by the natural gradient technique. The parameter updating is guided via the Kullback-Leibler divergence between the multivariate Normal and the Boltzmann densities. This approach makes sense because it is well-known that the Boltzmann function yields a reliable model to simulate particles near to optimum locations.

Three main contributions are presented here in order to build an effective EDA. The first one is a natural gradient formula which allows for an update of the parameters of a density function. These equations are related to an exponential parametrization of the search distribution. The second contribution involves the approximation of the developed gradient formula and its connection to the importance sampling method. The third contribution is a parameter update rule which is designed to control the exploration and exploitation phases of the algorithm.

The proposed EDA is tested on a benchmark of 16 problems and compared versus the XNES and iAMaLGaM algorithms. The statistical results show that the performance of the proposed method is competitive and it is the winner in several problems.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Unconstrained optimization; G.1.2 [**Approximation**]: Special function approximations; G.3 [**Probability and statistics**]: Multivariate statistics

Keywords

Estimation of Distribution Algorithm, Boltzmann Distribution, Natural Gradient, Optimization

1. INTRODUCTION

A common approach of the Evolutionary Algorithm technique (EA) consists in taking simulations from a statistical function, which fits the optimum locations; e.g. EDAs, ES, IGO, et cetera. In order to improve the search process of the EAs based on statistical models, a large effort has been made to enhance the modelling of the search space [6] [11] [16]. The previous use can be tackled by adding new information into the parameter estimation of density functions. This paper introduces contributions in this trend by building a natural gradient based EDA and developing a Boltzmann based natural gradient.

The EDAs adapt the parameters of a search distribution in order to favour the most promising regions. Thus, the EDA focuses the probability mass around optimum locations. However, it is well-known that this approach presents several issues through the optimization process [11]. For instance, the spread of population is reduced too early and the typical parameter estimation assumes that the individuals have equal probability. Our proposal overcomes the previous issues by adapting the parameter estimation via the natural gradient approach.

The natural gradient approach has been successfully applied to Evolutionary Strategies (ES) and Information Geometric Optimization algorithms (IGO) [4] [6] [18]. These methods adapt the parameters of a search distribution via a vector calculated in the parameter space. This vector, so-called natural gradient, provides the steepest ascent direction of a parameter space with a certain underlying structure. So, the parameter updating ensures the simulation of new individuals in promising regions. A typical natural gradient is pointing to the direction of higher expected fitness. Here, we propose another approach based on the Boltzmann distribution of the fitness function.

This paper presents a new EDA capable of simulating individuals in promising regions. This proposal involves the prediction of the exploration and the exploitation phases; i.e. the enlargement or reduction of the spread of the search distribution according to the gathered information. It is controlled via the natural gradient proposed in this paper. Additionally, the way we adapt a natural gradient into the EDA approach might be transferable to other EAs.

The organization of the paper is as follows. Section 2 provides the basics of the natural gradient approach. Section 3 introduces a Boltzmann based natural gradient. Section 4 develops an estimator for the natural gradient proposed in the previous section. Section 5 presents an EDA based on the natural gradient and the Boltzmann distribution. Section 6 is dedicated to testing the proposed EDA against two algorithms from literature. Finally, Section 7 provides some concluding remarks.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

DOI: http://dx.doi.org/10.1145/2739480.2754803

2. THE NATURAL GRADIENT APPROACH

Many stochastic optimisation methods tackle the black-box optimisation problems, $\mathcal{F}(\vec{x})$, by iteratively updating a density function $Q(\vec{x};\vec{\theta})$, so that any new sample has more chances to improve the fitness values of actual individuals. In this context, the natural gradient provides a straightforward way to adapt the parameters $\vec{\theta}$ of the search distribution $Q(\vec{x};\vec{\theta})$.

THEOREM 1. *The steepest ascent direction of a function $J(\vec{\theta})$ in a Riemannian space is given by*

$$\widetilde{\nabla J}(\vec{\theta}) = G^{-1}(\vec{\theta})\nabla J(\vec{\theta}) \tag{1}$$

where G^{-1} is the inverse of the Riemannian metric tensor G and $\nabla J(\vec{\theta})$ is the conventional gradient.

The direction $\widetilde{\nabla J}(\vec{\theta})$ stated in theorem 1 is usually named as the natural gradient of $J(\vec{\theta})$ in the Riemmanian space [5]. So, any vector calculated by (1) provides the steepest direction of a parameter space with a certain underlying structure. It suggests an iterative procedure to search the optimum on $J(\vec{\theta})$ by an ascent algorithm as follows

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \eta\widetilde{\nabla J}(\vec{\theta}_t), \tag{2}$$

where the step size η might be allowed to change at every iteration [2].

Since the stochastic methods use samples from a density function to optimise the objective function $\mathcal{F}(\vec{x})$, the updating rule presented in Eq. (2) must be adapted accordingly. Firstly, it is known that the Riemmanian structure of the parameter space of a statistical model is defined by its Fisher information matrix $\mathbf{F}$ [3]; i.e. the Riemmanian structure G can be replaced by $\mathbf{F}$. Secondly, the function $J(\vec{\theta})$ to be optimized in the density parameter space must be determined. The Natural Evolution Strategie (NES) method applies a straightforward definition for $J(\vec{\theta})$ by

$$J_E(\vec{\theta}) = \mathbb{E}_Q[\mathcal{F}(\vec{x})] = \int \mathcal{F}(\vec{x})Q(\vec{x};\vec{\theta})\,d\vec{x}. \tag{3}$$

Thus, NES uses a natural gradient to update the parameters of the search distribution in the direction of higher expected fitness. Additionally the gradient of $J_E(\vec{\theta})$ can be written as

$$\nabla_{\vec{\theta}} J_E(\vec{\theta}) = \int \mathcal{F}(\vec{x})[\nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})]\, Q(\vec{x};\vec{\theta})\,d\vec{x} \tag{4}$$

and approximated as

$$\nabla_{\vec{\theta}} J_E(\vec{\theta}) \approx \frac{1}{N}\sum_{k=1}^{N} \mathcal{F}(\vec{x}_k)\nabla_{\vec{\theta}} \log Q(\vec{x}_k;\vec{\theta}) \tag{5}$$

because of the so-called log-likelihood trick and the Monte Carlo approximation [17]. It leads to formulate the natural gradient ascent method by

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \eta\mathbf{F}^{-1}\nabla_{\vec{\theta}} J_E(\vec{\theta}), \tag{6}$$

where $\mathbf{F}$ represents the Fisher information matrix.

Because of Eq. (6) is a valid updating rule to any parametric distribution, it is looks tempting to utilize any available density from literature. But, many parametric distributions do not have a closed-form expression of the Fisher information matrix. On the other hand, even the typical multivariate Gaussian function presents an issue to update the covariance matrix via the natural gradient ascent method. However, a technique to avoid the computation of the Fisher matrix of the multivariate Gaussian distribution is emphasized below because our proposal takes advantage of this idea [9].

Let us assume $Q(\vec{x};\vec{\theta})$ is a multivariate normal density $\mathcal{N}(\vec{\mu},\mathbf{\Sigma})$ and $\mathbf{\Sigma} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$. Also, suppose that $\vec{x}_k = \vec{\mu} + \mathbf{C}\vec{z}_k$ where $\vec{z}_k$ is an outcome of $\mathcal{N}(\mathbf{0},\mathbf{I})$. The idea consists in a suitable transformation of the global coordinate system to a local coordinate system. This change of coordinates causes the actual search distribution $\mathcal{N}(\vec{\mu},\mathbf{\Sigma})$ to match with a standard normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$. As a consequence, the updating rule for the parameters is written as

$$(\vec{\delta},\mathbf{L}) \mapsto (\vec{\mu}_{new},\mathbf{C}_{new}) = (\vec{\mu} + \mathbf{C}\vec{\delta}, \mathbf{C}\exp(\mathbf{L}/2)) \tag{7}$$

where 'exp' represents an exponential map between matrices. Note that, if there is no update in the parameter of the density, i.e. $(\vec{\delta} = \mathbf{0}, \mathbf{L} = \mathbf{0})$, then the new parameters are the original ones. Also, the gradient $\nabla_{\vec{\theta}} J_E(\vec{\theta})$ and the natural gradient $\widetilde{\nabla J}_E(\vec{\theta})$ are equal when $\vec{\delta} = \mathbf{0}$ and $\mathbf{L} = \mathbf{0}$. As a consequence, this change of coordinates allows to compute the natural gradient without the calculus of the Fisher information matrix because

$$\nabla_{\vec{\delta}=\mathbf{0}} \log Q(\vec{x};\vec{\delta},\mathbf{L}=\mathbf{0}) = \vec{z} \tag{8}$$

and

$$\nabla_{\mathbf{L}=\mathbf{0}} \log Q(\vec{x};\vec{\delta}=\mathbf{0},\mathbf{L}) = \frac{1}{2}(\vec{z}\vec{z}^{\mathsf{T}} - \mathbf{I}). \tag{9}$$

So, the updating equation for each parameter can be written as follows

$$\vec{\mu}_{new} = \vec{\mu} + \frac{\eta}{N}\mathbf{C}\left(\sum_{k=1}^{N}\mathcal{F}(\vec{x}_k)\cdot\vec{z}\right) \tag{10}$$

$$\mathbf{C}_{new} = \mathbf{C}\exp\left(\frac{\eta}{4N}\sum_{k=1}^{N}\mathcal{F}(\vec{x}_k)\cdot(\vec{z}_k\vec{z}_k^{\mathsf{T}} - \mathbb{I})\right). \tag{11}$$

The previous proposal has been successfully used in evolutionary computation [1] [17] [18]. The next sections develop a new proposal for parameter updating of the Gaussian distribution based on the natural gradient approach and the Boltzmann density.

3. A NATURAL GRADIENT RELATED TO THE BOLTZMANN DENSITY

Equations (10) and (11) produce new parameters according to the direction of higher expected fitness. Additionally this proposal shows interesting characteristics and good performance [15]. However, criticism arises due to the fact that the function $J_E(\vec{\theta})$ can produce unstable estimates mainly because the expected value can be influenced by outliers [4] [6].

Following we propose a function $J(\vec{\theta})$ based on the divergence between the search distribution $Q(\vec{x};\vec{\theta})$ and the Boltzmann density

$$P(\vec{x};\beta) = \frac{\exp(\beta\mathcal{G}(\vec{x}))}{Z_P}. \tag{12}$$

The parameters of Eq. (12) must satisfy that $\beta \geq 0$ and $\mathcal{G}(\vec{x})) \geq 0$. Additionally the normalisation constant Z_P ensures that $P(\vec{x};\beta)$ is a density function.

The Boltzmann density allows for the simulation of samples near optimum locations. It is a very interesting characteristic, but it can not be explicitly used in black-box optimisation because the fitness formula is required in the energy function $\mathcal{G}(\vec{x})$ [11]. A way to overcome this issue consists in approximating the Boltzmann function via a measure between probability distributions. In this context, our proposal is stated in Eq. (13).

$$\begin{aligned} J_D(\vec{\theta}) &= -D_{KL}(Q(\vec{x};\vec{\theta})||P(\vec{x};\beta)) \\ &= -\int Q(\vec{x};\vec{\theta}) \log \frac{Q(\vec{x};\vec{\theta})}{P(\vec{x};\beta)}\, d\vec{x} \end{aligned} \tag{13}$$

The main difference with previous proposals is that another density function is considered. Note that the minus sign is added in order to ensure that $J_D(\vec{\theta})$ is a maximization problem. Although the Kullback-Leibler divergence is used to facilitate the calculus developed below, other measures between probability distributions might be utilized.

The gradient of $J_D(\vec{\theta})$ can be written as

$$\begin{aligned} \nabla_{\vec{\theta}} J_D(\vec{\theta}) &= -\int Q(\vec{x};\vec{\theta}) \nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})\, d\vec{x} \\ &- \int \log Q(\vec{x};\vec{\theta}) \nabla_{\vec{\theta}} Q(\vec{x};\vec{\theta})\, d\vec{x} \\ &+ \int \log P(\vec{x};\beta) \nabla_{\vec{\theta}} Q(\vec{x};\vec{\theta})\, d\vec{x} \\ &= -\int [\nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})] Q(\vec{x};\vec{\theta})\, d\vec{x} \\ &- \int \log Q(\vec{x};\vec{\theta}) [\nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})] Q(\vec{x};\vec{\theta})\, d\vec{x} \\ &+ \int \log P(\vec{x};\beta) [\nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})] Q(\vec{x};\vec{\theta})\, d\vec{x} \end{aligned} \tag{14}$$

when the so-called log-likelihood trick is used. A final reduction shows a few similarities with the gradient stated in Eq. (4), see Eq. (15).

$$\nabla_{\vec{\theta}} J_D(\vec{\theta}) = \int \log \left(\frac{P(\vec{x};\beta)}{\mathrm{e} \cdot Q(\vec{x};\vec{\theta})} \right) [\nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})] Q(\vec{x};\vec{\theta})\, d\vec{x} \tag{15}$$

The logarithmic function in Eq. (15) is applied on the ratio $P(\vec{x};\beta)/(\mathrm{e} \cdot Q(\vec{x};\vec{\theta}))$, where $\mathrm{e} = \exp(1)$. In fact, it shows a connection with importance sampling technique because the importance weight $P(\vec{x};\beta)/Q(\vec{x};\vec{\theta})$ is embedded in the gradient $\nabla_{\vec{\theta}} J_D(\vec{\theta})$. Also, the logarithmic function associates different weights to each simulation according to the probability of being sampled from the Boltzmann density; only importance weights above $1/\mathrm{e}$ have a positive value. These observations suggest that $\nabla_{\vec{\theta}} J_D(\vec{\theta})$ has a completely different performance in comparison with $\nabla_{\vec{\theta}} J_E(\vec{\theta})$.

Although $\nabla_{\vec{\theta}} J_D(\vec{\theta})$ is defined for any search distribution $Q(\vec{x};\vec{\theta})$, a few issues might arise when applying the provided formula. The next section presents an efficient way to estimate the natural gradient w.r.t $J_D(\vec{\theta})$ based on the importance sampling technique, the Monte Carlo method and an exponential parameterization.

4. THE NATURAL GRADIENT ESTIMATION

Despite the described $\nabla_{\vec{\theta}} J_E(\vec{\theta})$ and $\nabla_{\vec{\theta}} J_D(\vec{\theta})$ come from different sources, both gradients show strong similarities and straightforward connections. In fact, the only difference between these is the factor-term which multiplies $\nabla_{\vec{\theta}} \log Q(\vec{x};\vec{\theta})$ $\cdot Q(\vec{x};\vec{\theta})$. Following a natural gradient estimator is proposed, which takes advantage of these connections, and a graphical comparison is performed versus a typical approach.

Before providing an estimate for $\nabla_{\vec{\theta}} J_D(\vec{\theta})$ an issue must be tackled: the normalization constant Z_P of the Boltzmann density is unknown, mainly because the energy function $\mathcal{G}(\vec{x})$ depends on the unknown fitness function $\mathcal{F}(\vec{x})$. In this case, a trick based on importance sampling without normalisation can be used.

Let us write the involved density functions as $P(\vec{x};\beta) = \widetilde{P}(\vec{x};\beta)/Z_P$ and $Q(\vec{x};\vec{\theta}) = \widetilde{Q}(\vec{x};\vec{\theta})/Z_Q$, where $\widetilde{P}(\vec{x};\beta) = \exp(\beta\mathcal{G}(\vec{x}))$, $Q(\vec{x};\vec{\theta}) \propto \widetilde{Q}(\vec{x};\vec{\theta})$, $\int \widetilde{Q}(\vec{x};\vec{\theta})\, d\vec{x} = Z_Q > 0$ and $\int \widetilde{P}(\vec{x};\beta)\, d\vec{x} = Z_P > 0$. Here, the ratio $P(\vec{x};\beta)/Q(\vec{x};\vec{\theta})$ is expressed as

$$\begin{aligned} \frac{P(\vec{x};\beta)}{Q(\vec{x};\vec{\theta})} &= \left(\frac{Z_Q}{Z_P} \right) \widetilde{W}(\vec{x};\vec{\theta},\beta) \\ &= \left(\frac{1}{\int \widetilde{W}(\vec{x};\vec{\theta},\beta) Q(\vec{x};\vec{\theta})\, d\vec{x}} \right) \widetilde{W}(\vec{x};\vec{\theta},\beta) \end{aligned} \tag{16}$$

where $\widetilde{W}(\vec{x};\vec{\theta},\beta) = \widetilde{P}(\vec{x};\beta)/\widetilde{Q}(\vec{x};\vec{\theta})$ [12]. Since the samples $\vec{x}_1, ..., \vec{x}_k, ..., \vec{x}_N \sim Q(\vec{x};\vec{\theta})$ then the Monte Carlo method allows to approximate equation (16) as

$$\frac{P(\vec{x};\beta)}{Q(\vec{x};\vec{\theta})} \approx \frac{\widetilde{W}(\vec{x};\vec{\theta},\beta)}{\frac{1}{N}\sum_{k=1}^{N} \widetilde{W}(\vec{x}_k;\vec{\theta},\beta)}. \tag{17}$$

Eq. (17) provides a simple way to avoid the computation of the normalisation constant Z_P. In the same vein Eq. (15) can be approximated via the Monte Carlo technique because of the last term $Q(\vec{x};\vec{\theta})$ [14]. With this in mind, let us express the approximation for $\nabla J_D(\vec{\theta})$ as follows

$$\nabla_{\vec{\theta}} J_D(\vec{\theta}) \approx \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{N \cdot \widetilde{W}(\vec{x}_i;\vec{\theta},\beta)}{\mathrm{e} \cdot \sum_{k=1}^{N} \widetilde{W}(\vec{x}_k;\vec{\theta},\beta)} \right) \nabla_{\vec{\theta}} \log Q(\vec{x}_i;\vec{\theta}). \tag{18}$$

A first glance over the estimates in Equations (5) and (18) suggests a slight difference in computational complexity. Furthermore the estimate of $\nabla J_D(\vec{\theta})$ introduces an extra parameter β which plays an important role in stochastic optimisation. Below a further analysis is developed by choosing a multivariate normal density as the search distribution $Q(\vec{x};\vec{\theta})$.

4.1 The multivariate Normal distribution

Although the developed approximation for $\nabla J_D(\vec{\theta})$ works to any parametric search distribution, the special case with respect to a multivariate Gaussian function deserves extra attention.

Please assume that $Q(\vec{x};\vec{\theta}) = \mathcal{N}(\vec{x};\vec{\mu},\mathbf{\Sigma})$, then the function $\widetilde{W}(\vec{x};\vec{\mu},\mathbf{\Sigma},\beta)$ can be expressed as

$$\begin{aligned} \widetilde{W}_{\mathcal{N}}(\vec{x}) &= \widetilde{W}(\vec{x};\vec{\mu},\mathbf{\Sigma},\beta) \\ &= \exp\left(\beta\mathcal{G}(\vec{x}) + \frac{1}{2}(\vec{x}-\mu)^{\mathsf{T}} \mathbf{\Sigma}^{-1} (\vec{x}-\mu) \right). \end{aligned} \tag{19}$$

So, the logarithmic function in Eq. (18) is applied on exponential terms. Although it seems that the numerical limits could be exceeded when the exponential term is calculated, a further analysis shows stability computation up to dimension $d = 100$. Let us consider that the maximum exponent-value to avoid numerical issues is above 300, i.e. $\exp(300)$ still is a valid number to be computed, despite it depending on the machine-precision. Since the power in $\widetilde{W}_{\mathcal{N}}(\vec{x})$ consists of $\beta\mathcal{G}(\vec{x})$ and $\frac{1}{2}(\vec{x}-\mu)^{\mathsf{T}}\mathbf{\Sigma}^{-1}(\vec{x}-\mu)$, each term must be separately analysed. Notice that $(\vec{x}-\mu)^{\mathsf{T}}\mathbf{\Sigma}^{-1}(\vec{x}-\mu) \sim \chi^2_{(d)}$; thus if $d = 100$ it is very unlikely to get a value above 300. On the other hand, the energy function $\mathcal{G}(\vec{x})$ is intimately related to the fitness function $\mathcal{F}(\vec{x})$, which means that large values of the fitness function might produce large energy values. To avoid this issue the next energy function is proposed

$$\mathcal{G}(\vec{x}_k) = \frac{g(\vec{x}_k)}{\max(g(\vec{x}_1), ..., g(\vec{x}_N))} \quad \text{where } g(\vec{x}_k) = \mathcal{F}(\vec{x}_k) - \min(\mathcal{F}(\vec{x}_1), ..., \mathcal{F}(\vec{x}_N)). \tag{20}$$

Note that the maximum value for $\mathcal{G}(\vec{x}_k)$ according to Eq. (20) is 1. As a consequence, only the parameter β might produce a power above 300. Thus, we strongly recommend $\beta \leq 50$ in order to avoid numerical issues (anyway in practical applications this value for β is not an issue).

Another consequence of the similarities between the natural gradient discussed in Section 2 and our proposal, is that the latest can also avoid the computation of the Fisher information matrix. So, recalling the discussed change of coordinate system, then the term $\nabla_{\vec{\theta}} \log Q(\vec{x}; \vec{\theta})$ in Eq. (18) is reduced as in Equations (8) and (9). It allows us to write the updating equations for each parameter as

$$\vec{\mu}_{new} = \vec{\mu} + \frac{\eta}{N}\mathbf{C}\left(\sum_{i=1}^{N} \log\left(\frac{N \cdot \widetilde{W}_{\mathcal{N}}(\vec{x}_i)}{\mathrm{e} \cdot \sum_{k=1}^{N} \widetilde{W}_{\mathcal{N}}(\vec{x}_k)}\right) \cdot \vec{z}_i\right) \tag{21}$$

and

$$\mathbf{C}_{new} = \mathbf{C}\exp\left(\frac{\eta}{4N}\sum_{i=1}^{N} \log\left(\frac{N \cdot \widetilde{W}_{\mathcal{N}}(\vec{x}_i)}{\mathrm{e} \cdot \sum_{k=1}^{N} \widetilde{W}_{\mathcal{N}}(\vec{x}_k)}\right)(\vec{z}_i\vec{z}_i^{\mathsf{T}} - \mathbf{I})\right) \tag{22}$$

where $\mathbf{\Sigma} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$, $\vec{z}_i = \mathbf{C}^{-1}(\vec{x}_i - \vec{\mu})$, $\widetilde{W}_{\mathcal{N}}(\vec{x}_i)$ is defined in Eq. (19) and the step size should satisfy $\eta \geq 0$.

4.2 A graphical comparison

This section is dedicated to graphically analysing the behaviour of two natural gradients: 1) the developed by expected fitness, and 2) our proposal based on the divergence Kullback-Leibler between the Gaussian and the Boltzmann distributions.

Among the typical natural gradient approaches in literature, the exponential natural evolution strategy (XNES) is selected for a visual comparison [9]. This method computes the natural gradient via an exponential parameterization of the multivariate Gaussian distribution. Also, it utilises a rank-based utility function $\mathcal{U}(\vec{x})$ instead of the usual fitness function $\mathcal{F}(\vec{x})$; which is a notable difference versus our proposal.

The experiments are performed on the Sphere problem; notice that a change of sign must be applied in order to effectively use the Eq. (20). The simulation procedure randomly generates 300 particles, i.e. the sample size is $N = 300$.

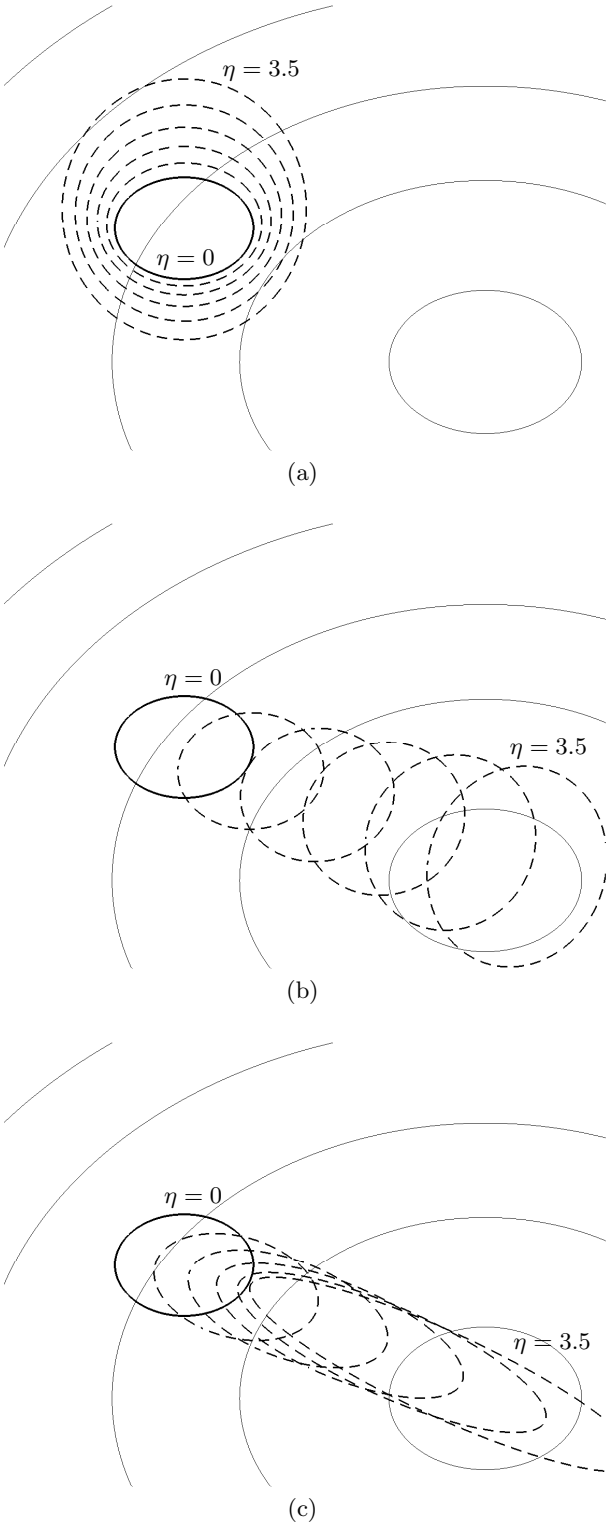

Figure 1: Examples of Gaussian Densities obtained by two different Natural Gradients. Top two Figures come from our approach. (a) $\widetilde{\nabla J}_D(\vec{\theta})$ with $\beta = 1/10$. (b) $\widetilde{\nabla J}_D(\vec{\theta})$ with $\beta = 10$. (c) $\widetilde{\nabla J}_E(\vec{\theta})$ from xNES.

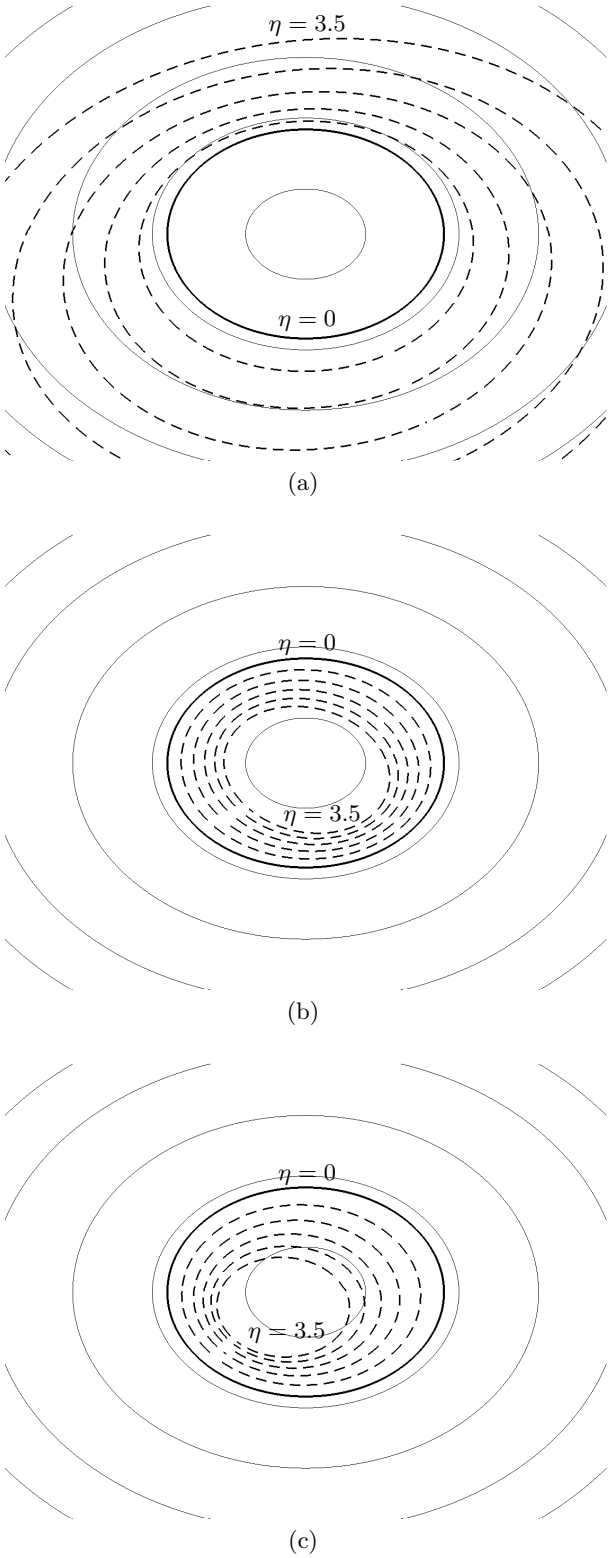

Figure 2: Examples of Gaussian Densities obtained by two different Natural Gradients. Top two Figures come from our approach. (a) $\widetilde{\nabla J_D(\vec{\theta})}$ with $\beta = 1/10$. (b) $\widetilde{\nabla J_D(\vec{\theta})}$ with $\beta = 10$. (c) $\widetilde{\nabla J_E(\vec{\theta})}$ from xNES.

Two different experiments were executed: 1) sampling at a medium distance from the global optimum, and 2) sampling around the global optimum. Figures 1 and 2 show the obtained parameters using different step sizes in the direction of the natural gradients. Following, the results of both experiments are commented on in detail.

Experiment 1
Initial parameters: $\vec{\mu} = [-5, 3]^{\mathsf{T}}$ and $\mathbf{\Sigma} = \mathbf{I}$.
Figure 1 shows the results of this experiment. Figures 1-(b) and 1-(c) look similar because the parameter $\beta = 10$ encourages the reduction of the spread of the Boltzmann density and tends to focus the probability mass around the best individuals. It is contrary to the observations in Figure 1-(a) where a larger step size enlarges the covariance matrix of the Gaussian distribution. Also, these observations are interesting because they mean that our proposal can modify the parameters in a similar way to the classic natural gradient of expected fitness.

Experiment 2
Initial parameters: $\vec{\mu} = [0, 0]^{\mathsf{T}}$ and $\mathbf{\Sigma} = \mathbf{I}$.
Figure 2 shows the results of this experiment. Once again Figures 2-(b) and 2-(c) show similar behaviour; both are focusing the Gaussian distribution on a smaller area around the global optimum. On the other hand, the parameter $\beta = 1/10$ encourages the increment of the spread of the Boltzmann density. As a consequence, in Figure 2-(a) the multivariate Gaussian distribution covers a bigger area according to the step size. This behaviour might be very useful to avoid being trapped in local optimum locations.

The previous experiments suggest that our proposal is capable of managing the exploration and exploitation phases. Note that, it is a desirable attribute to build effective evolutionary algorithms and maintain the diversity of the population [8] [16]. The next section is dedicated to building an Estimation of Distribution Algorithm (EDA) based on the natural gradient developed through the previous sections.

5. THE NATURAL GRADIENT BASED EDA

The Estimation of Distribution Algorithms (EDAs) fit a multivariate density function on regions near optimum locations; so that any new sample might be near the global optimum. However, it is well-known that the EDA approach might reduces the spread of population too early [13]. The EDA presented below deals with similar issues by effectively using the gathered information through the optimisation process. In addition, some of the ideas presented here can be applied in the general EDA context.

This section introduces our proposal named Natural Gradient based Estimation of Distribution Algorithm (NAGEDA), see Figure 3. The algorithm exploits the developed updating rules for the parameters $\vec{\mu}$ and $\mathbf{\Sigma} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$. The central idea consists in the computation of a search distribution capable of predicting the location of optimum regions. Here, the Natural Gradient estimator developed in sections 3 and 4 plays an important role.

The NAGEDA starts with a random population, which is utilised to compute the fitness values $\mathcal{F}(\vec{x}_i)$ and the energy values $\mathcal{G}(\vec{x}_i)$. Note that Eq. (20) is a valid formula to find the maximum values of the fitness function $\mathcal{F}(\vec{x})$. But, the minimization problems should be attained via $\mathcal{F}(\vec{x}) =$

$-\mathcal{F}(\vec{x})$. Then, the whole set of individuals $\mathcal{P}^{(t)}$ is used to compute the maximum likelihood estimates $\vec{\mu}$ and $\mathbf{\Sigma}$. Additionally, a repair technique is applied to the estimate $\mathbf{\Sigma}$, line 7. This technique replaces the eigenvalues below 10^{-100} by setting the minimum value to 10^{-100}; as a consequence the matrix $\mathbf{\Sigma}$ is built again. Next, the initial matrix $\mathbf{C}$ is computed via a singular value decomposition of $\mathbf{\Sigma}$. It allows for the estimate of the vectors $\vec{z}_i$; note that $\vec{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Hence, the proposed natural gradient is computed to update the parameters $\vec{\mu}$ and $\mathbf{\Sigma}$, line 10. Next, new individuals are simulated from the updated multivariate Gaussian density. Since the set $\mathcal{P}_S^{(t)}$ could have better individuals than the actual population, the next population $\mathcal{P}^{(t+1)}$ is chosen by $\mathcal{P}^{(t)} \cup \mathcal{P}_S^{(t)}$. This step adds the necessary selective pressure to maintain the improvement of the population. This algorithm only needs to update the variables β and η, but an auto-adaptive rule is added for this purpose, lines 16-24.

1: Initialize $t \leftarrow 0$, N, S_s, $\beta \leftarrow 10$ and $\eta \leftarrow 1/10$
2: $\mathcal{P}^{(t)} \leftarrow \mathcal{U}(Domain)$ ▷ First population
3: Evaluate $\mathcal{F}(\vec{x}_i)$
4: Compute $\mathcal{G}(\vec{x}_i)$ by Eq. (20)
5: **while** (Stop condition is not reached) **do**
6: Estimate $\vec{\mu}$ and $\mathbf{\Sigma}$ via maximum likelihood
7: $\mathbf{\Sigma} \leftarrow$ RepairMethod($\mathbf{\Sigma}$)
8: Compute $[U, S^2, U^{\mathsf{T}}] = \mathbf{\Sigma}$ and $\mathbf{C} = U(S^2)^{1/2}$
9: Compute $\vec{z}_i = \mathbf{C}^{-1}(\vec{x}_i - \vec{\mu})$
10: Estimate $\vec{\mu}_{new}$ and $\mathbf{C}_{new}$ by Eq. (21)-(22)
11: $\mathcal{P}_S^{(t)} \leftarrow S_s$ individuals from $Q(x; \vec{\mu}_{new}, \mathbf{C}_{new}\mathbf{C}_{new}^{\mathsf{T}})$
12: $\mathcal{P}_S^{(t)} \leftarrow$ Reinsertion($\mathcal{P}_S^{(t)}$)
13: Evaluate $\mathcal{F}(\vec{x}_j)$ and $\mathcal{G}(\vec{x}_j)$ of $\mathcal{P}_S^{(t)}$
14: $\mathcal{P}^{(t+1)} \leftarrow$ Best N individuals of $\mathcal{P}^{(t)} \cup \mathcal{P}_S^{(t)}$
15: $M_s \leftarrow |\mathcal{P}^{(t+1)} \cap \mathcal{P}_S^{(t)}|$ ▷ Number of survivors
16: $\epsilon = |\, M_s/S_s - 0.5\,|$ ▷ Absolute value-factor
17: **if** $2M_s > S_s$ **then** ▷ Exploitation
18: $\eta = (1 + \epsilon)\eta$
19: $\beta = 10$
20: **else** ▷ Exploration
21: $\eta = \eta/(1 + \epsilon)$
22: $\beta = 1/10$
23: **end if**
24: **if** $\eta \leq 10^{-300}$ **then** $\eta = 1$ **end if**
25: $t \leftarrow t + 1$
26: **end while**
27: Return the elite individual in $\mathcal{P}^{(t)}$

Figure 3: Pseudocode of the proposed EDA: Natural Gradient based Estimation of Distribution Algorithm (NAGEDA).

The auto-adaptive rule for the variables β and η is based on the intersection between the sets $\mathcal{P}^{(t+1)}$ and $\mathcal{P}_S^{(t)}$. The number of survivors M_s is the number of new samples that are preserved from the current generation to the next one. This quantity allows for the control of the exploration and exploitation phases. If most of the new simulations are within the next population, the algorithm changes to the exploitation phase, contrary to an unsuccessful sampling. In addition, the step size of the natural gradient estimate is also adapted according to the number of survivor individuals. Here, the η value is increased in the exploitation phase, opposite to the exploration phase.

Since the simulation method might build samples outside the search domain, a re-insertion technique is added, line 12. Let $\gamma_k = l_k^{upper} - l_k^{lower}$ be the domain length in dimension k, where l_k^{upper} and l_k^{lower} are the upper bound and lower bound in dimension k. For each dimension, the new sample $\vec{y} = (y_1, \cdots, y_k, \cdots, y_d)$ is tested/replaced by

- if $y_k > l_k^{upper}$ then $a = (y_k - l_k^{upper})/\gamma_k$ and $y_k = l_k^{upper} - \gamma_k(a - \lfloor a \rfloor)$
- if $y_k < l_k^{lower}$ then $a = (l_k^{lower} - y_k)/\gamma_k$ and $y_k = l_k^{lower} + \gamma_k(a - \lfloor a \rfloor)$

which ensures any new individual is inside the domain.

An empirical study of NAGEDA suggests that its performance dependens on the population size N. We found an equation to increase N according to the dimension size d. Similar to well-known literature [7], a regression method is applied to obtain the equation $N = \exp(\lambda + 0.01 \cdot d) \cdot d$; where the best value for λ depends on the problem. Here we suggest $\lambda = 1.4$ for unimodal problems, $\lambda = 1.5$ for multimodal problems and $\lambda = 1.9$ for the Rosenbrock problem. Since the sample size is not as critical as the population size, the chosen sample size is $S_s = \lceil M_s/5 \rceil$.

6. EXPERIMENTS

This section is dedicated to testing our algorithm versus two state of the art methods: 1) the exponential Natural Evolution Strategy (xNES), and 2) the Incremental Adapted Maximum-Likelihood Gaussian Model Iterated Density Estimation Evolutionary Algorithm (iAMaLGaM) [9] [7]. The first algorithm applies the natural gradient approach in evolutionary strategies, while the second one is capable of adapting the multivariate Gaussian density in an efficient way to avoid premature convergence.

$\mathcal{F}$	NAME	U/M	$\mathcal{F}(\vec{x}^*)$	x_i^*	DOMAIN
$\mathcal{F}_1$	Sphere	U	0	0	$x_i \in [-600, 300]^d$
$\mathcal{F}_2$	Schwefel 1.2	U	0	0	$x_i \in [-20, 10]^d$
$\mathcal{F}_3$	Trid	U	Λ_f	Λ_x	$x_i \in [-d^2, d^2]^d$
$\mathcal{F}_4$	Zakharov	U	0	0	$x_i \in [-20, 10]^d$
$\mathcal{F}_5$	Ellipsoid	U	0	0	$x_i \in [-20, 10]^d$
$\mathcal{F}_6$	Cigar Tablet	U	0	0	$x_i \in [-20, 10]^d$
$\mathcal{F}_7$	Two Axes	U	0	0	$x_i \in [-20, 10]^d$
$\mathcal{F}_8$	Exponential	U	-1	0	$x_i \in [-1, 0.5]^d$
$\mathcal{F}_9$	Rosenbrock	M	0	1	$x_i \in [-20, 10]^d$
$\mathcal{F}_{10}$	Ackley	M	0	0	$x_i \in [-20, 10]^d$
$\mathcal{F}_{11}$	Griewangk	M	0	0	$x_i \in [-600, 300]^d$
$\mathcal{F}_{12}$	Cos. Mixture	M	$-0.1d$	0	$x_i \in [-1, 0.5]^d$
$\mathcal{F}_{13}$	Levy-Montalvo 1	M	0	-1	$x_i \in [-20, 10]^d$
$\mathcal{F}_{14}$	Levy-Montalvo 2	M	0	1	$x_i \in [-20, 10]^d$
$\mathcal{F}_{15}$	Levy 8	M	0	-1	$x_i \in [-20, 10]^d$
$\mathcal{F}_{16}$	Bohachevsky	M	0	0	$x_i \in [-20, 10]^d$

Table 1: Search domain, modality (U/M), global minimum and names of the test problems [10]. The expressions $\Lambda_f = -d(d+4)(d-1)/6$ and $\Lambda_x = i(d+1-i)$ are taken when $\mathcal{F}_3$ is in use.

A suitable set of problems is chosen in order to perform an adequate comparison, see Table 1. The source code is provided by the homepage of the authors. So, the parameters of each competitor-algorithm are the default ones. The experiments contrast the error $\mathcal{F} - \mathcal{F}^*$ reached for each method,

$\mathcal{F}$	NAGEDA	xNES	ρ
$\mathcal{F}_1$	**100.00**	**100.00**	
	9.05e-9±8.00e-10	8.81e-9±1.09e-9	1.95e-1
	3.62e+4±2.51e+2	5.98e+4±5.81e+2	**1.00e-4**
$\mathcal{F}_2$	**100.00**	**100.00**	
	8.95e-9±9.47e-10	8.97e-9±9.25e-10	8.96e-1
	2.94e+4±3.52e+2	5.60e+4±6.00e+2	**1.00e-4**
$\mathcal{F}_3$	**100.00**	**100.00**	
	8.87e-9±7.82e-10	9.03e-9±8.15e-10	3.14e-1
	3.73e+4±3.08e+2	6.19e+4±4.63e+2	**1.00e-4**
$\mathcal{F}_4$	**100.00**	**100.00**	
	8.76e-9±9.38e-10	9.07e-9±7.20e-10	6.80e-2
	2.98e+4±4.04e+2	6.80e+4±3.11e+3	**1.00e-4**
$\mathcal{F}_5$	**100.00**	94.00	
	8.72e-9±1.09e-9	1.58e-8±3.38e-8	1.22e-1
	3.84e+4±2.83e+2	1.31e+5±0.00e+0	**1.00e-4**
$\mathcal{F}_6$	**100.00**	0.00	
	9.02e-9±8.39e-10	2.61e+0±3.74e+0	**1.00e-4**
	3.99e+4±3.53e+2	3.00e+5±0.00e+0	**1.00e-4**
$\mathcal{F}_7$	**100.00**	0.00	
	8.84e-9±1.06e-9	1.05e+2±6.16e+1	**1.00e-4**
	3.97e+4±3.49e+2	3.00e+5±0.00e+0	**1.00e-4**
$\mathcal{F}_8$	**100.00**	74.00	
	8.90e-9±1.02e-9	1.58e-1±2.70e-1	**2.00e-4**
	2.13e+4±2.22e+2	1.40e+5±0.00e+0	**1.00e-4**
$\mathcal{F}_9$	**100.00**	**100.00**	
	8.75e-9±9.96e-10	9.13e-9±8.32e-10	**3.96e-2**
	1.18e+5±3.76e+3	**1.06e+5±5.37e+3**	**1.00e-4**
$\mathcal{F}_{10}$	**100.00**	**100.00**	
	9.26e-9±6.02e-10	9.50e-9±5.06e-10	**3.40e-2**
	5.49e+4±2.68e+2	9.13e+4±5.62e+2	**1.00e-4**
$\mathcal{F}_{11}$	**100.00**	**100.00**	
	8.52e-9±1.09e-9	9.11e-9±6.95e-10	**3.00e-3**
	3.67e+4±3.38e+2	5.39e+4±4.30e+2	**1.00e-4**
$\mathcal{F}_{12}$	**100.00**	28.00	
	8.91e-9±1.05e-9	1.70e+0±2.07e+0	**1.00e-4**
	2.89e+4±6.58e+2	2.38e+5±0.00e+0	**1.00e-4**
$\mathcal{F}_{13}$	**100.00**	**100.00**	
	8.76e-9±9.62e-10	9.09e-9±7.72e-10	6.61e-2
	2.65e+4±1.20e+3	4.39e+4±4.71e+2	**1.00e-4**
$\mathcal{F}_{14}$	**100.00**	**100.00**	
	8.99e-9±8.97e-10	8.89e-9±9.86e-10	5.71e-1
	2.92e+4±2.82e+2	4.93e+4±4.85e+2	**1.00e-4**
$\mathcal{F}_{15}$	**100.00**	**100.00**	
	9.12e-9±8.55e-10	8.91e-9±8.95e-10	2.28e-1
	2.91e+4±1.56e+3	4.81e+4±4.41e+2	**1.00e-4**
$\mathcal{F}_{16}$	**100.00**	82.00	
	8.94e-9±7.49e-10	1.52e-1±3.51e-1	**3.50e-3**
	3.64e+4±3.88e+2	1.06e+5±0.00e+0	**1.00e-4**

Table 2: Comparison between NAGEDA (our approach) and xNES methods in 30 dimensions. The winner is marked in boldface according to a non-parametric bootstrap test. Further details in section 6.

$\mathcal{F}$	NAGEDA	iAMaLGaM	ρ
$\mathcal{F}_1$	**100.00**	**100.00**	
	9.05e-9±8.00e-10	8.88e-9±8.03e-10	2.85e-1
	3.62e+4±2.51e+2	3.85e+4±5.11e+2	**1.00e-4**
$\mathcal{F}_2$	**100.00**	**100.00**	
	8.95e-9±9.47e-10	8.92e-9±1.00e-9	8.69e-1
	2.94e+4±3.52e+2	3.15e+4±4.52e+2	**1.00e-4**
$\mathcal{F}_3$	**100.00**	**100.00**	
	8.87e-9±7.82e-10	8.65e-9±1.18e-9	2.65e-1
	3.73e+4±3.08e+2	4.01e+4±6.22e+2	**1.00e-4**
$\mathcal{F}_4$	**100.00**	**100.00**	
	8.76e-9±9.38e-10	8.87e-9±8.47e-10	5.23e-1
	2.98e+4±4.04e+2	3.31e+4±4.61e+2	**1.00e-4**
$\mathcal{F}_5$	**100.00**	**100.00**	
	8.72e-9±1.09e-9	9.05e-9±7.94e-10	8.26e-2
	3.84e+4±2.83e+2	4.54e+4±1.94e+3	**1.00e-4**
$\mathcal{F}_6$	**100.00**	**100.00**	
	9.02e-9±8.39e-10	8.88e-9±8.19e-10	3.75e-1
	3.99e+4±3.53e+2	4.70e+4±2.00e+3	**1.00e-4**
$\mathcal{F}_7$	**100.00**	**100.00**	
	8.84e-9±1.06e-9	8.93e-9±9.44e-10	6.29e-1
	3.97e+4±3.49e+2	5.45e+4±2.13e+3	**1.00e-4**
$\mathcal{F}_8$	**100.00**	**100.00**	
	8.90e-9±1.02e-9	8.75e-9±8.18e-10	4.18e-1
	2.13e+4±2.22e+2	2.25e+4±4.12e+2	**1.00e-4**
$\mathcal{F}_9$	**100.00**	**100.00**	
	8.75e-9±9.96e-10	8.87e-9±9.31e-10	5.25e-1
	1.18e+5±3.76e+3	**1.07e+5±7.64e+3**	**1.00e-4**
$\mathcal{F}_{10}$	**100.00**	**100.00**	
	9.26e-9±6.02e-10	9.51e-9±3.73e-10	**1.44e-2**
	5.49e+4±2.68e+2	**5.19e+4±5.14e+2**	**1.00e-4**
$\mathcal{F}_{11}$	**100.00**	**100.00**	
	8.52e-9±1.09e-9	8.83e-9±8.75e-10	1.19e-1
	3.67e+4±3.38e+2	**3.51e+4±4.79e+2**	**1.00e-4**
$\mathcal{F}_{12}$	**100.00**	**100.00**	
	8.91e-9±1.05e-9	8.88e-9±9.15e-10	8.68e-1
	2.89e+4±6.58e+2	3.22e+4±1.42e+4	1.18e-1
$\mathcal{F}_{13}$	**100.00**	**100.00**	
	8.76e-9±9.62e-10	8.79e-9±1.06e-9	9.00e-1
	2.65e+4±1.20e+3	**2.56e+4±4.30e+2**	**1.00e-4**
$\mathcal{F}_{14}$	**100.00**	**100.00**	
	8.99e-9±8.97e-10	8.82e-9±1.00e-9	3.64e-1
	2.92e+4±2.82e+2	**2.86e+4±4.59e+2**	**1.00e-4**
$\mathcal{F}_{15}$	**100.00**	**100.00**	
	9.12e-9±8.55e-10	8.88e-9±8.47e-10	1.53e-1
	2.91e+4±1.56e+3	**2.80e+4±4.38e+2**	**1.00e-4**
$\mathcal{F}_{16}$	**100.00**	**100.00**	
	8.94e-9±7.49e-10	9.08e-9±7.07e-10	3.45e-1
	3.64e+4±3.88e+2	4.54e+4±2.10e+4	**3.90e-3**

Table 3: Comparison between NAGEDA (our approach) and iAMaLGaM methods in 30 dimensions. The winner is marked in boldface according to a non-parametric bootstrap test. Further details in section 6.

where $\mathcal{F}^*$ is the best fitness value found by an algorithm. The methods are executed 50 times for each test problem. Also, the algorithms are stopped when either: a maximum number of $10000 \cdot d$ evaluations or an error $\mathcal{F} - \mathcal{F}^* < 1 \times 10^{-8}$ is reached. Additionally the population size and sample size of the NAGEDA are modified by the equations provided in section 5.

Tables 2 and 3 summarize the comparison between our proposal and the competitors. These tables present the percentage of success rate, reached fitness values and needed number of evaluations, mean and standard deviation for each algorithm in 30 dimensions. The last column shows a nonparametric bootstrap test with precision $\alpha = 0.05$ by comparing the mean values. If ρ is boldface, there is statistical evidence of the difference between two algorithms. Furthermore, the winner is marked in boldface.

NAGEDA versus xNES

The results in Table 2 show that the NAGEDA has better performance than the xNES, even in the unimodal problems. Also, it is confirmed by a non-parametric bootstrap test on the number of function evaluations. A notable difference in the mean values can be confirmed by a visual comparison. This surprising result is mainly because the developed natural gradient is capable of adapting the parameters of

the search distribution towards exploration and exploitation phases.

NAGEDA versus iAMaLGaM

Table 3 shows that both algorithms reached the precision requested for all test problems. However, there are significant differences in several problems. There is statistical evidence to confirm that the NAGEDA requires fewer number of function evaluations than iAMaLGaM to reach the desired precision in unimodal problems. On the other hand, in 6 out of 8 functions the iAMaLGaM shows better performance than the NAGEDA for the multimodal problems. Despite these facts being supported by the statistical testing, a visual comparison shows a slight difference between the mean values of the needed evaluations.

The comparisons discussed above show that the NAGEDA has better performance than the xNES in all the test functions. Despite the first one requiring a higher population size than the second algorithm, we can conclude that the NAGEDA uses the gathered information in an efficient way. On the other hand, NAGEDA and iAMaLGaM show a similar performance in most of the benchmark problems.

7. CONCLUSIONS

This paper proposes a natural gradient based on the Kullback-Leibler divergence between a search distribution and the Boltzmann distribution. An analysis of the developed formulae show a strong relationship between our proposal and a typical natural gradient based on the expected fitness. This fact allows us to reuse some ideas and methods from literature, mainly from the NES and the Monte Carlo approaches. Additionally the behaviour of our proposal can be explained via the importance sampling technique.

When the search distribution is set to the multivariate Gaussian distribution, the computational complexity of the natural gradient estimation is reduced. This is because it allows us to use an exponential parameterization for avoiding the computation of the Fisher information. Although the developed natural gradient estimate seems more complex than the typical one, a visual study exhibits that the first one has interesting properties related to the exploration and exploitation phases.

A natural gradient based EDA (NAGEDA) is introduced in order to apply the ideas presented in this paper. Also, a comparison is performed versus two successful algorithms: xNES and iAMaLGaM. Statistical results support that NAGEDA is competitive with state of the art algorithms, mainly because the proposed natural gradient allows for the control of the exploration and exploitation phases.

Future work will contemplate an auto-adaptive population size and additional enhancement techniques to reduce the number of function evaluations. Moreover, notice that the main contributions developed in this paper can be extended to other evolutionary algorithms.

8. REFERENCES

[1] Y. Akimoto, A. Auger, and N. Hansen. Comparison-Based Natural Gradient Optimization in High Dimension. In *Genetic and Evolutionary Computation Conference*, Vancouver, Canada, July 2014. ACM.

[2] S. Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2):251–276, Feb. 1998.

[3] S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. Oxford University Press, 2000.

[4] L. Arnold, A. Auger, N. Hansen, and Y. Ollivier. Information-Geometric Optimization Algorithms: A Unifying Picture via Invariance Principles. Technical report, Cornell University Library, 2011.

[5] K. Arwini and C. Dodson. *Information Geometry: Near Randomness and Near Independence*. Number 1953 in Lecture Notes in Mathematics. Springer, 2008.

[6] H. Beyer. Convergence Analysis of Evolutionary Algorithms that are based on the Paradigm of Information Geometry. *Evolutionary Computation*, 22(4):679–709, Dec. 2014.

[7] P. A. N. Bosman, J. Grahl, and D. Thierens. Benchmarking Parameter-free Amalgam on Functions with and Without Noise. *Evololutionary Computation*, 21(3):445–469, Sept. 2013.

[8] T. Friedrich, N. Hebbinghaus, and F. Neumann. Comparison of Simple Diversity Mechanisms on Plateau Functions. *Theoretical Computer Science*, 410(26):2455–2462, June 2009.

[9] T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra, and J. Schmidhuber. Exponential Natural Evolution Strategies. In *Genetic and Evolutionary Computation Conference*, 2010.

[10] M. Jamil and X.-S. Yang. A Literature Survey of Benchmark Functions for Global Optimization Problems. *CoRR*, abs/1308.4008, 2013.

[11] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, Boston, MA, 2002.

[12] A. B. Owen. *Monte Carlo Theory, Methods and Examples*. In progress, available online, 2013.

[13] P. Pošík. Preventing Premature Convergence in a Simple EDA via Global Step Size Setting. In G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, editors, *PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 549–558. Springer Berlin Heidelberg, 2008.

[14] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, Secaucus, NJ, USA, 2005.

[15] T. Schaul. Benchmarking Exponential Natural Evolution Strategies on the Noiseless and Noisy Black-box Optimization Testbeds. In *Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012.

[16] M. Črepinšek, S. Liu, and M. Mernik. Exploration and Exploitation in Evolutionary Algorithms: A Survey. *ACM Comput. Surv.*, 45(3):35:1–35:33, July 2013.

[17] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural Evolution Strategies. *Journal of Machine Learning Research*, 15:949–980, 2014.

[18] S. Yi, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic Search Using the Natural Gradient. In *Proceedings of the ICML*, pages 1161–1168, New York, NY, USA, 2009. ACM.