Designing the Boltzmann Estimation of Multivariate Normal Distribution: issues, goals and solutions

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

S. Ivvan Valdez
Center for Research in Mathematics
Guanajuato, México
ivvan@cimat.mx

***Abstract*—This paper introduces a new Estimation of Distribution Algorithm (EDA) based on the multivariate Boltzmann distribution. In this work, the design variables and the energy function of the Boltzmann distribution are continuous. Note that since the population has finite size, it can only approximate a continuous Boltzmann distribution with some error. In order to tackle this issue, the parameter estimators for the mean vector and covariance matrix of a Multivariate Normal Density that approximate the Boltzmann density, are derived by minimizing the Kullback-Leibler divergence. The algorithm introduced here uses one energy function for the mean estimator and another for the covariance matrix estimator. The first function places the probability mass around the most promising regions by assigning larger weights to individuals with higher fitness. However, the second function orients the covariance matrix along improving directions by assigning larger weights to individuals with lower fitness. Our proposal combines the conveniences of linear weights with a simple annealing schedule to regulate the exploration and exploitation of the search process. The resulting algorithm is named the Boltzmann Estimation of Multivariate Normal Algorithm (BEMNA). By applying the developed formulae the BEMNA is capable of adapting the structure of a density model to the promisory search directions. BEMNA is tested with a benchmark of 16 functions and contrasted with the AMaLGaM algorithm, a state of the art EDA. Statistical tests of the experimental data show the competitiveness of the proposed algorithm.**

Keywords*—*Estimation of Distribution Algorithm, Boltzmann distribution, Real valued optimization, Evolutionary computation, Kullback-Leibler divergence

I. Introduction

The Estimation of Distribution Algorithms [1] EDAs are optimization methods based on estimating and sampling a probability distribution of some set of individuals chosen after a quality criteria. The most promising regions are unknown and have to be discovered during the optimization process. The EDA must favor such regions by assigning to them the highest probability values. Hence, the main goal of the EDA is to pose the probability mass around the optima. The common strategy, without loss of generality for a maximization process, is to reinforce the sampling of regions where the higher fitness function values have been sampled, and to disregard the regions with the lower values. The most common EDA scheme for continuous optimization is to use a multivariate or univariate Normal distribution [2]–[6]. The parameters of the Normal density are estimated by using maximum likelihood estimators (MLEs) over the selected set, which is usually determined by a selection operator: truncation selection, tournament selection, et cetera. Although these approaches have proven competitive up to some extent, some issues are worth to notice here:

- It is a well-known issue that the variance in estimation of distribution algorithms is often less than required, hence, the MLE variance estimator is not the most adequate technique of searching for the optimum vector [7], [8].
- Not only is it possible that the variance could be less than required, but in addition, the orientation of the variance could be inadequate for sampling better solutions than those already known. Take into consideration the case when the population is not around the optimum region, hence, estimating the structure of the search density from the population will not guide the search to new promising regions. In this case, *the structure of the search density must be oriented according to the improving directions instead of reproducing the population density structure.*

Our purpose is to circumvent the mentioned issues by proposing weighted estimators for a parametric distribution which approximate the Boltzmann. The Boltzmann distribution has been largely used in optimization. For instance, in the context of Estimation of Distribution Algorithms (EDAs), researchers have proposed different approaches, such as the BEDA [9]–[11]. This is a general EDA framework based on the Boltzmann distribution, from which practical EDAs have been derived. For example: the FDA [12] which considers a factorization of the energy function, the Yun Peng et al. [13] and Valdez et al. works [14] which propose different approximations in continuous search spaces by minimizing the Kullbak-Leibler divergence. The unifying characteristic of these approaches is that they intend to set the highest probability on the most promising regions, considering a promising region as the region where the best fitness values have been sampled. This is to say, *the better the fitness function is in a region, the more intense the sampling must be*.

The Gibbs or Boltzmann distribution of an energy function g(x) is defined by

$$p(x) := \int \frac{exp(\beta g(\vec{x}))}{Z}\, d\vec{x}. \tag{1}$$

978-1-4799-7492-4/15/$31.00

As can be noticed in Equation (1), the objective function can be used directly as an energy function. In practical approaches, the Boltzmann distribution cannot be used directly for sampling because it is necessary to determine the objective function in the whole domain. For this reason, the parameters of a density function are computed by minimizing a measure between the parametric distribution and the Boltzmann distribution; for instance, the Kullback-Leibler divergence [13]–[15].

There are remarkable challenges to consider when designing EDAs based on the Boltzmann distribution:

- To choose an adequate β parameter in Equation (1). Usually β depends on the time or is dynamic during the optimization process. The process which controls the β updating each generation is called the *annealing schedule*. The annealing schedule can be used to manage the exploration and convergence of the algorithm.
- To derive robust parameter estimators for the approximated distribution. Some approaches [13], [16] have derived formulae for estimating parameters of a probability function which approximate the Boltzmann, by weighting the population or selected set by exponential functions, similar to Equation (1). Even though competitive results are obtained, these proposals often suffer from premature convergence because the exponential function drastically leads the probability mass to suboptimal positions. This behavior can be avoided by manipulating the β value, but it is not simple to determine a competent manner to do so. A second option is to obtain formulae with an auto-adapted β which do not drastically impact the estimators from one generation to the next.
- The last two issues are also related with the aforementioned variance reduction, which is a common issue in EDAs [7]. In the same vein, the structure of the probability function is related to the orientation of the density, which is defined by the covariance matrix. The structure by itself could be a determinant factor in the covariance reduction. Consider the case of two covariance matrices with the same eigenvalues, the same variance magnitude. They could perform completely differently if they had different eigenvectors; if the eigenvectors are not oriented in an improvement direction the algorithm could get trapped in the region currently covered by the population, even if there is no a local or global optimum inside.

Our proposal intends to tackle the aforementioned challenges by introducing the following features:

- A proposal of annealing schedule to update the β value.
- Robust formulae for the estimators computation, in the sense that they are not impacted as drastically as the exponential function used in other approaches.
- The novel annealing schedules tackle the variance reduction problem, hence, these are mechanisms to avoid the premature convergence of the algorithm.
- *A different energy function for the Bolztmann approximation, which induces an adequate orientation in the covariance matrix, favoring the improving directions.*

The paper is organized as follows: Section II presents the derivation of parameter estimators, it is to say the mean vector and covariance matrix of a Multivariate Normal, which approximates the Boltzmann Density. Section III introduces a remarkable concept regarding the estimation of the covariance matrix, which significantly benefits the search by estimating an adequate structure of the probability search distribution.

Section IV introduces the Boltzmann Estimation of Multivariate Normal Algorithm (BEMNA). Then, Section V is devoted to test the proposed EDA on well-known Benchmark problems versus another state of the art algorithm. Finally, Section VI provides some concluding remarks.

II. Approximating the Boltzmann distribution by the Normal Multivariate distribution

This section introduces the formulae to estimate the mean vector $\vec{\mu}$ and covariance matrix $\mathbf{\Sigma}$ of a Normal multivariate density Q_x, which approximates the multivariate Boltzmann density P_x.

Consider the multivariate Normal density, $Q_x = Q(x; \mu, \Sigma)$, as shown in Equation (2). The corresponding Boltzmann density, defined in the same domain as Q_x, is in Equation (3).

$$Q_x = \frac{1}{\sqrt{(2\pi)^d \left|\mathbf{\Sigma}\right|}} \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu})^t \mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right\} \quad (2)$$

$$P_x = \frac{\exp(\beta g(\vec{x}))}{Z} \quad (3)$$

The procedure to find the parameters of Q_x, which best approximate P_x, consist in minimizing a measure of dissimilarity between density functions. Similar to previous works [13] [14], the Kullback-Leibler Divergence presented in Equation (4), $K_{QP} = D_{KL}(Q_x||P_x)$, is used for this purpose.

$$K_{QP} = \int Q_x \log \frac{Q_x}{P_x}\, d\vec{x} \quad (4)$$

The minimization of K_{QP} for finding the optimal parameters $[\vec{\mu}_*, \mathbf{\Sigma}_*]$ can be stated as shown in Equation (5).

$$[\vec{\mu}, \mathbf{\Sigma}] = \arg\min\{K_{QP}\} \quad (5)$$

Notice that K_{QP} can be rewritten as

$$\begin{aligned} K_{QP} &= \int Q_x \log Q_x \, d\vec{x} - \int Q_x \log P_x \, d\vec{x} \\ &= -H(Q_x) - \int Q_x \log P_x \, d\vec{x} \\ &= -\frac{1}{2}\log((2\pi e)^d \left|\Sigma\right|) - \int Q_x \log P_x \, d\vec{x}, \end{aligned} \quad (6)$$

where the term $H(Q_x)$ stands for the entropy of the multivariate Normal density [17]. In order to get the parameters which minimize the Kulback-Leibler Divergence, the partial derivatives are calculated in Equations (7) and (8).

$$\begin{aligned}\frac{\delta K_{QP}}{\delta \vec{\mu}} &= -\int \frac{\delta Q_x}{\delta \vec{\mu}} \log P_x \, d\vec{x} \\ &= -\int Q_x[\mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})] \log P_x \, d\vec{x}\end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\delta K_{QP}}{\delta \mathbf{\Sigma}} &= -\frac{1}{2}\frac{\delta \log(|\mathbf{\Sigma}|)}{\delta \mathbf{\Sigma}} - \int \frac{\delta Q_x}{\delta \mathbf{\Sigma}} \log P_x \, d\vec{x} \\ &= -\frac{1}{2}\int Q_x[\mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^t \mathbf{\Sigma}^{-1}] \log P_x \, d\vec{x} \\ &\quad + \frac{1}{2}\int Q_x \mathbf{\Sigma}^{-1} \log P_x \, d\vec{x} - \frac{1}{2}\mathbf{\Sigma}^{-1}\end{aligned} \tag{8}$$

The optimal estimates for the mean vector and covariance matrix are obtained by forcing the derivatives equal to 0, as in Equations (9) and (10), and solving for $\vec{\mu}$ and $\mathbf{\Sigma}$ respectively.

$$\begin{aligned}0 &= \frac{\delta K_{QP}}{\delta \vec{\mu}} \\ 0 &= \vec{\mu}\int Q_x \log P_x \, d\vec{x} - \int Q_x \vec{x} \log P_x \, d\vec{x} \\ 0 &= \vec{\mu}\beta E_Q[g(\vec{X})] - \vec{\mu}\log Z - E_Q[g(\vec{X})\vec{X}]\beta + E_Q[\vec{X}]\log Z \\ \vec{\mu} &= \frac{E_Q[g(\vec{X})\vec{X}]}{E_Q[g(\vec{X})]}\end{aligned} \tag{9}$$

The following equivalences were used to obtain Equations (9) and (10):

- $\log P_x = \beta g(\vec{x}) - \log Z$,
- $\int Q_x \vec{x} \, d\vec{x} = E_Q[\vec{X}]$, $\int Q_x(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^t \, d\vec{x} = E_Q[(\vec{X}-\vec{\mu})(\vec{X}-\vec{\mu})^t]$, and other similar equations.
- As $\vec{X} \sim Q_x$ then $E_Q[\vec{X}] = \vec{\mu}$ and $E_Q[(\vec{X}-\vec{\mu})(\vec{X}-\vec{\mu})^t] = \mathbf{\Sigma}$.

$$\begin{aligned}0 &= \frac{\delta K_{QP}}{\delta \mathbf{\Sigma}} \\ 0 &= \int Q_x (\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^t \log P_x \, d\vec{x} \\ &\quad - \int Q_x \log P_x \, d\vec{x} \, \mathbf{\Sigma} + \mathbf{\Sigma} \\ \mathbf{\Sigma} &= \frac{E_Q[g(\vec{X})(\vec{X}-\vec{\mu})(\vec{X}-\vec{\mu})^t]}{E_Q[g(\vec{X})] - 1/\beta}\end{aligned} \tag{10}$$

Finally, for estimating the parameters using the observations $\vec{x}^{(1)}, \vec{x}^{(2)}, ..., \vec{x}^{(N)}$ of the random variable $\vec{X} \sim Q_x$, a numerical stochastic approximation by the Monte Carlo method is computed, as shown in Equations (11) and (12). These two equations are the estimators that approximate the parameters of the search distribution.

$$\vec{\mu}_* = \frac{\sum_{i=1}^N g(\vec{x}^{(i)})\vec{x}^{(i)}}{\sum_{i=1}^N g(\vec{x}^{(i)})} \tag{11}$$

$$\mathbf{\Sigma}_* = r_e \cdot \sum_{i=1}^N g(\vec{x}^{(i)})(\vec{x}^{(i)}-\vec{\mu})(\vec{x}^{(i)}-\vec{\mu})^t \tag{12}$$

where

$$r_e = \left(\sum_{i=1}^N g(\vec{x}^{(i)}) - \frac{N}{\beta}\right)^{-1} \tag{13}$$

A. A note about the derived formulae

The estimators in Equations (11) and (12) use weights defined by $\frac{g(\vec{x}^{(i)})}{\sum_{i=1}^N g(\vec{x}^{(i)})}$. In other words, the weighted estimators are computed by using weights proportional to the objective function value of each individual in the selected set. In contrast to similar approaches [13], [16], there are some advantages of these derivations:

- A proportional weight of the estimators avoids drastic changes when the individuals considerably differ in the objective function value. It is to say, if a new individual with a large objective value is sampled, the exponential weights could concentrate the probability mass around this single individual, leading the algorithm to premature convergence [13]. This effect is diminished when using proportional weights.
- A second advantage is that the minimum variance, which is bounded by a $\beta = \infty$, is not 0 for our approach, which is a significant advantage considering that EDAs naturally suffer from premature convergence and variance reduction [13], [16].

The energy function $g(\vec{x})$ must be positive or equal to zero in the domain. However, the objective function $\mathcal{F}(\vec{x})$ might be negative, considering that this is a maximization/minimization problem, in order to construct a valid energy function. Throughout this paper, the $g(\vec{x}^{(i)})$ value is computed as $g(\vec{x}^{(i)}) = -\mathcal{F}(\vec{x}^{(i)}) - \min_i(-\mathcal{F}(\vec{x}^{(i)}))$. The next section analyzes the importance of the energy function and other topics related to evolutionary computation.

B. The annealing schedule

As seen in Equations (11), (12) and (13), the β value only affects the covariance matrix computation. The grade of impact of β over the covariance is highly related with $\sum_{i=1}^N g(\vec{x}^{(i)})$. On one hand, $N/\beta < \sum_{i=1}^N g(\vec{x}^{(i)})$ must hold in order to maintain a positive variance in the diagonal of the covariance matrix. On the other hand, if $N/\beta << \sum_{i=1}^N g(\vec{x}^{(i)})$ then its effect is diminished. As a consequence, the estimator reaches the minimum variance when $\beta \to \infty$ because $N/\beta \to 0$. An interesting remark about this last setting is that the Normal distribution with such a minimum variance is not similar to a Dirac δ, while the corresponding Boltzmann distribution actually is.

Notice that a method to control the β value is needed, but it is not straightforward since β might increase indefinitely. However, from the previous discussion: β is highly related with $\sum_{i=1}^{N} g(\vec{x}^{(i)})$. Hence, it could be written as factor multiplying $\sum_{i=1}^{N} g(\vec{x}^{(i)})$. As a consequence, r_e must be a reciprocal portion of $\sum_{i=1}^{N} g(\vec{x}^{(i)})$. In agreement with the arguments stated above, assume a parametrization of the β value as follows:

$$\beta = N/((1-\gamma)\sum_{i=1}^{N} g(\vec{x}^{(i)})), \tag{14}$$

where $0 < \gamma < 1$, then the scale factor r_e is

$$r_e = \left(\gamma \sum_{i=1}^{N} g(\vec{x}^{(i)})\right)^{-1}. \tag{15}$$

According to this proposal the covariance estimator is rewritten as in Equation (16).

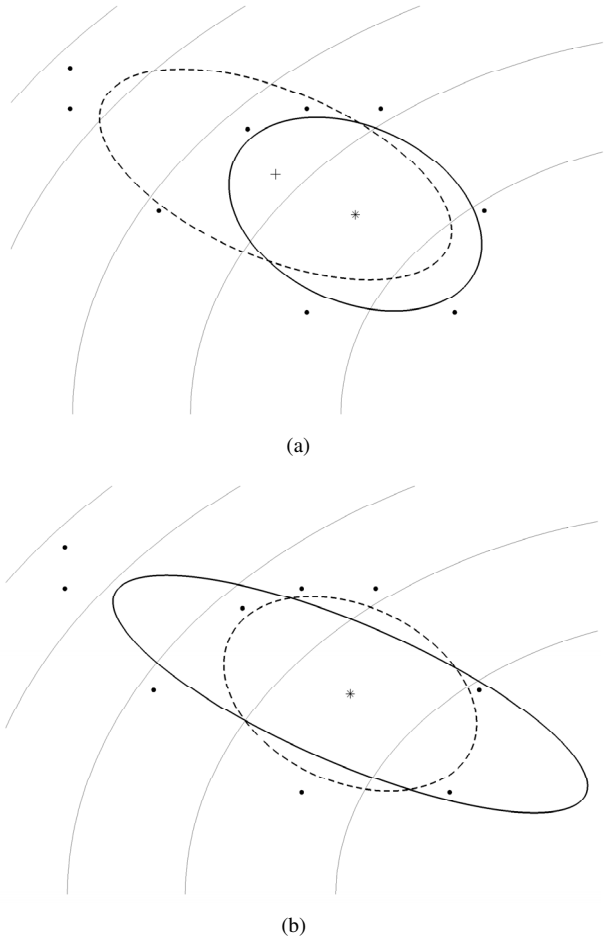
$$\mathbf{\Sigma}_* = \frac{\sum_{i=1}^{N} g(\vec{x}^{(i)})(\vec{x}^{(i)} - \vec{\mu})(\vec{x}^{(i)} - \vec{\mu})^t}{\gamma \cdot \sum_{i=1}^{N} g(\vec{x}^{(i)})} \tag{16}$$

This new equation for $\mathbf{\Sigma}_*$ is more convenient than the one presented in Equation (12) because the scale factor γ is easier to control than the original β parameter. Also γ is more intuitive since it is basically a scaling factor, which can help to regulate the size of the covariance matrix. Additionally, notice that the individual with the highest $g(\vec{x}^{(i)})$ has the greatest impact in the covariance computation.

III. Using a different energy function for the covariance matrix computation

Every individual contribute to compute the mean vector $\vec{\mu}_*$ and the covariance matrix $\mathbf{\Sigma}_*$. Notice that the information of fitness function is introduced via $g(\vec{x}^{(i)})$. The way this information modifies the estimates is worth to analyze. First let us study the mean vector estimator, Equation (11). Most of the probability mass is around the mean vector, which means that this position is crucial to produce samples on promissory regions. In Equation (11) each individual is weighted by $g(\vec{x}^{(i)})/\sum_{i=1}^{N} g(\vec{x}^{(i)})$, and the sum of these weights is equal to one. So, any mean vector computed in this way is a linear combination of the individuals and is inside the convex hull of the actual population [18].

In addition, Figure 1-(a) shows several differences between this proposal and Maximum Likelihood Estimation (MLE). The location of the mean vector via MLE only depends on the density of population whilst the location of $\vec{\mu}_*$ is biased according to the fitness function. Hence, our proposal usually situates the center of the density on promissory regions according to the location of the best individuals. Also, Figure 1-(a) shows a smaller covariance matrix in our proposal compared to the computed one via MLE, when using the fitness function as the energy function. This behaviour could be undesirable because it might lead to the collapse of the population in a suboptimal point after some generations.



(a)

(b)

Fig. 1. Comparison of normal densities. The symbol + shows the position of mean vector via maximum likelihood estimation while the asterisk symbol shows the location of $\vec{\mu}_*$. (a) Maximum likelihood estimators (dashed line) versus our proposal (solid line), computed using Equations (11) and (16). (b) Covariance matrix computed with weights using $g(\vec{x}^{(i)})$ in Equation (16) (dashed line), versus computation using $h(\vec{x}^{(i)})$ in Equation (17) (solid line).

The proposed estimator in Equation (16) multiplies each term $(\vec{x}^{(i)} - \vec{\mu})(\vec{x}^{(i)} - \vec{\mu})^t$ by the weight $g(\vec{x}^{(i)})/\sum_{i=1}^{N} g(\vec{x}^{(i)})$. Since $\vec{\mu}_*$ is close to the best individuals, the vector $\vec{x}^{(i)} - \vec{\mu}_*$ has a shorter length for the best individuals than the worst ones. Then, summation of matrices $(g(\vec{x}^{(i)})/\sum_{i=1}^{N} g(\vec{x}^{(i)})) \cdot (\vec{x}^{(i)} - \vec{\mu})(\vec{x}^{(i)} - \vec{\mu})^t$ induces structures with a smaller variance than the MLE estimator. In order to overcome this issue, we propose to change the energy function $g(\vec{x}^{(i)})$ as shown in Equation (17).

$$\mathbf{\Sigma}_* = \frac{\sum_{i=1}^{N} h(\vec{x}^{(i)})(\vec{x}^{(i)} - \vec{\mu})(\vec{x}^{(i)} - \vec{\mu})^t}{\gamma \cdot \sum_{i=1}^{N} h(\vec{x}^{(i)})} \tag{17}$$

This proposal uses an energy function $h(\vec{x^{(i)}})$, which returns a value between 0.01 and 1 according to the rank of the individual $\vec{x^{(i)}}$. Here, the values are taken from partitions on $[0.01, 1]$. Let j^{th} denote the rank of individual $\vec{x}^{(i)}$ in the population sorted by fitness values, such that if $\vec{x}^{(i)}$ is the

best individual then $j^{th} = 1$, otherwise if $\vec{x}^{(k)}$ is the worst individual then $j^{th} = N$. Thus, the energy function $h(\cdot)$, for the covariance matrix computation, is defined as follows

$$h(\vec{x^{(i)}}) = \frac{1}{N} + \frac{99(j^{th} - 1)}{100(N-1)}, \tag{18}$$

where the first term ensures a non-zero weight for the best individual.

Equations (17) and (18) cause the differences $\vec{x}^{(i)} - \vec{\mu}_*$ of the worst individuals to be multiplied by higher weights than the other individuals. Hence, the proposed formulae for the mean $\vec{\mu}_*$ and covariance $\mathbf{\Sigma}_*$ estimators equip the algorithm with the following features:

- The proposed mean estimator poses the maximum probability mass around the best known region.
- The covariance matrix estimator enlarges the probability density, because usually the farthest an individual is from the mean the greater its weight is.
- Additionally, the enlargement of the covariance matrix aligns the normal multivariate distribution through the direction of maximum improvement; it goes from the worst to the best individuals.

A visual comparison between covariance estimates (16) and (17) is presented in Figure 1-(b). As can be observed our proposal is more prone to avoiding premature convergence than the MLE estimator.

A. Updating the scale factor γ

In order to effectively apply the developed equations into the EDA context, the γ factor must be adapted through the optimization process. Here, the annealing schedule modifies γ in a linear fashion based on the improvements of the fitness values from the parents to the children. Notice that updating β of Equations (12,13) is equivalent to updating γ in Equation (16). Most of the similar previous proposals adapt the β or γ parameter in an exponential manner [13], with a consequent drastic impact from one generation to the next one.

As mentioned $\gamma = 1$ corresponds to the minimum variance (i.e. $\beta = \infty$), thus the limits for this parameter are well defined as $0 < \gamma \leq 1$. The updating of γ proceeds as follows

$$\gamma^{t+1} = \gamma^t - \frac{2}{N}\left(\frac{12M_s}{N} - 1\right), \tag{19}$$

where M_s is the number of new simulations that are preserved from the current generation to the next one, throughout this paper known as the *survivor individuals*. Note that the maximum number of survivor individuals is the sample size S_s. Additionally, two conditions must be added to ensure $0 < \gamma \leq 1$: if $\gamma \leq 0$ then $\gamma = 0.01$, also if $\gamma > 1$ then $\gamma = 1$.

If there are several survivor individuals this rule increases the covariance matrix values; which means that there is still a chance to find better individuals in the vicinity. Otherwise, the covariance is reduced in order to focus on a smaller area. As a consequence, this annealing schedule controls the exploration and exploitation according to the gathered information.

IV. The Boltzmann Estimation of Mutivariate Normal Algorithm

This section introduces our proposal named Boltzmann Estimation of Normal Multivariate Algorithm (BEMNA), which is presented in Figure 2. The algorithm exploits the developed parameter estimators $\vec{\mu}_*$ and $\mathbf{\Sigma}_*$, which allow to model the uncertainty of the optimum location.

The BEMNA starts with a random population, it is evaluated using the fitness function, then the values $g(\vec{x}^{(i)})$ are computed in order to turn the minimization problem to a maximization one. The whole set of individuals $\mathcal{P}^{(t)}$ is used to compute the estimates $\vec{\mu}_*$ and $\mathbf{\Sigma}_*$. Next, new individuals are simulated from the multivariate normal density. The set $\mathcal{P}_S^{(t)}$ could have better fitness values than the current population. The next population $\mathcal{P}^{(t+1)}$ is created by selecting the best individuals between $\mathcal{P}^{(t)}$ and $\mathcal{P}_S^{(t)}$. The number of survivor individuals M_s is used to adapt the scale factor γ according to the gathered information. Finally, the best individual in the population is returned as the best approximation to the global optimum.

1: Initialize $t \leftarrow 0$, N_s, S_s and γ^0
2: $\mathcal{P}^{(t)} \leftarrow \mathcal{U}(Domain)$ ▷ First population
3: Evaluate $\mathcal{F}(\vec{x}^{(i)})$
4: Compute $g(\vec{x}^{(i)}) = -\mathcal{F}(\vec{x}^{(i)}) - \min(-\mathcal{F}(\vec{x}^{(i)}))$
5: **while** (Stop condition is not reached) **do**
6: Estimate $\vec{\mu}_*$ and $\mathbf{\Sigma}_*$ of $\mathcal{P}^{(t)}$, Eq. (11) and (17)
7: $\mathcal{P}_S^{(t)} \leftarrow$ Simulate S_s individuals from $Q(x; \vec{\mu}_*, \mathbf{\Sigma}_*)$
8: $\mathcal{P}_S^{(t)} \leftarrow$ Reinsertion($\mathcal{P}_S^{(t)}$)
9: Evaluate $\mathcal{F}(\vec{x}^{(j)})$ and $g(\vec{x}^{(j)})$ of $\mathcal{P}_S^{(t)}$
10: $\mathcal{P}^{(t+1)} \leftarrow$ Best N_s individuals of $\mathcal{P}^{(t)} \cup \mathcal{P}_S^{(t)}$
11: $M_s \leftarrow |\mathcal{P}^{(t+1)} \cap \mathcal{P}_S^{(t)}|$
12: Compute γ^{t+1} by Eq. (19)
13: **if** $\gamma \leq 0$ **then** $\gamma \leftarrow 0.01$ **end if**
14: **if** $\gamma > 1$ **then** $\gamma \leftarrow 1$ **end if**
15: $t \leftarrow t + 1$
16: **end while**
17: Return the elite individual in $\mathcal{P}^{(t)}$

Fig. 2. Pseudocode of the proposed EDA: Boltzmann Estimation of Multivariate Normal Algorithm (BEMNA).

A. Recommendations about the parameters

The only parameters of the BEMNA are the number of partitions n_p, population size N_s and sample size S_s. After an empirical study on several benchmark problems, some comments/recommendations about the parameters can be provided:

- The factor $2/N$ in Eq. (19) controls the velocity of parameter adaptation. A factor close to 1 speeds up convergence whilst a higher value reduces the convergence velocity. As a consequence the best value for this parameter depends on the problem. However, our

empirical observations suggest that the proposed value $2/N_s$ performs adequately for most of the problems.

- An adequate population size N_s is essential for the BEMNA. If there are too few samples the population collapses, leading to premature convergence. On the contrary, a large sample increases the number of function evaluations and reduces the convergence velocity. Our empirical study suggests that the minimum recommended population size is $N_s = \lfloor 19.92 + 1.35 \cdot d^{1.44} \rfloor$ where d is the number of dimensions. The previous equation was calculated by a regression method applied to the minimum population sizes which delivers successful results in most of the benchmark problems.
- The sample size S_s affects the convergence velocity. If a few samples are simulated then there is less of a chance of intensively exploring the search space, while favoring a fast convergence to a local optimum. On the contrary, a large number of simulations have a greater chance of intensively exploring the search space, avoiding getting trapped in a local optimum; however it takes a larger number of fitness evaluations. According to our experiments, the BEMNA shows a competitive performance with a sample size $S_s = N_s/6$ for most of the problems.

A well known issue in Normal multivariate EDAs is that the covariance matrix could present negative eigenvalues (due to numerical errors [5]). In order to avoid this issue we apply the repairing scheme stated in Figure 3. This repairing method is not utilized most of the time, and it just needs a few iterations to fix the covariance matrix.

1: Let $\boldsymbol{L}$ be the matrix of eigenvectors of covariance matrix $\boldsymbol{\Sigma}$ by columns, d the number of dimensions and $\boldsymbol{\Lambda}$ a diagonal matrix with the corresponding eigenvalues, in decreasing order.
2: $i \leftarrow d$, $flag \leftarrow 0$
3: **while** $i > 0$ **do**
4: **if** $\boldsymbol{\Lambda}_{i,i} < 1e-100$ **then**
5: $\boldsymbol{\Lambda}_{i,i} \leftarrow 1e-100$
6: $flag \leftarrow 1$
7: **else**
8: **break**
9: **end if**
10: $i \leftarrow i-1$
11: **end while**
12: **if** $flag == 1$ **then**
13: $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^t$
14: **end if**

Fig. 3. Repairing scheme for non-positive definite covariance matrix $\boldsymbol{\Sigma}$.

The parameter γ controls the spread of simulated individuals based on the survivors from the current generation to the next one. Since no a priori knowledge is considered to choose an initial value for γ (e.g. the type of problem, survivors individuals, etcetera), a value in the middle of the domain of γ is chosen, it is to say $\gamma^0 = 0.5$. In addition, since the simulation method might generate samples outside the search domain, a re-insertion technique is added in line 8. Let $\zeta_k = l_k^{upper} - l_k^{lower}$ 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 $\vec{y}^{(i)} = (y_1^{(i)}, \cdots, y_k^{(i)}, \cdots, y_D^{(i)})$ is tested/replaced by the following rules:

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

which ensures that any new individual is inside the domain.

V. Experiments

This section is dedicated to test our algorithm versus another state of the art EDA: The Adapted Maximum Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm (AMaLGaM-IDEA) [19]. It is capable of modeling dependencies among variables by using a Multivariate Normal Density. In addition, it adds at least three different rules to avoid premature convergence.

TABLE I. Benchmark minimization problems [2], [4], [14]. Left: unimodal functions. Right: multimodal functions. Further details in section V.

$\mathcal{F}$	Name	Domain	$\mathcal{F}$	Name	Domain
$\mathcal{F}_1$	Sphere	$[-600, 300]^d$	$\mathcal{F}_9$	Rosenbrock	$[-20, 10]^d$
$\mathcal{F}_2$	Different Powers	$[-20, 10]^d$	$\mathcal{F}_{10}$	Ackley	$[-20, 10]^d$
$\mathcal{F}_3$	Schwefel 1.2	$[-20, 10]^d$	$\mathcal{F}_{11}$	Griewangk	$[-600, 300]^d$
$\mathcal{F}_4$	Trid	$[-d^2, d^2]^d$	$\mathcal{F}_{12}$	Levy 8	$[-20, 10]^d$
$\mathcal{F}_5$	Zakharov	$[-20, 10]^d$	$\mathcal{F}_{13}$	Bohachevsky	$[-20, 10]^d$
$\mathcal{F}_6$	Ellipsoid	$[-20, 10]^d$	$\mathcal{F}_{14}$	Rastrigin	$[-20, 10]^d$
$\mathcal{F}_7$	Cigar Tablet	$[-20, 10]^d$	$\mathcal{F}_{15}$	Drop Wave	$[-20, 10]^d$
$\mathcal{F}_8$	Two Axes	$[-20, 10]^d$	$\mathcal{F}_{16}$	Salomon	$[-100, 50]^d$

In order to perform an adequate comparison between both algorithms, a suitable set of problems is chosen as shown in Table I. All of these are minimization problems. For applying the BEMNA these are converted to maximization and translated to positive as follows: $g(\vec{x}) = -\mathcal{F}(\vec{x}) - min(-\mathcal{F}(\vec{x}))$. Where $\mathcal{F}(\vec{x})$ is the objective function and $g(\vec{x})$ is the energy function used in Figure 2. The minimum fitness value of all problems is 0 except for $\mathcal{F}_4$ and $\mathcal{F}_{15}$; where $\mathcal{F}_4^* = -d(d+4)(d-1)/6$ and $\mathcal{F}_{15}^* = -1$.

The experiments contrast the error $\mathcal{F} - \mathcal{F}^*$ reached for each algorithm, where $\mathcal{F}^*$ represents the best fitness value returned by an algorithm. Also this section provides the mean, standard deviation and a non-parametric bootstrap hypothesis test, which elucidates if there is statistical difference between the mean of best objective function values delivered by the algorithms, and the mean of number of function evaluations used to reach the desired precision.

Since there are several variants of the AMaLGaM algorithm, we have chosen the parameter-free version which does not need extra parameter tuning. On the other hand, the parameters of the BEMNA (i.e. population size, number of samples, etcetera) were discussed in the previous section. Both algorithms are stopped when either: a maximum number

of $10000 \cdot d$ evaluations or a precision to the optimum value of $\mathcal{F} - \mathcal{F}^* < 1 \times 10^{-8}$ is reached.

The comparison is summarized in Table II. It contrasts the error $\mathcal{F} - \mathcal{F}^*$ reached for each algorithm in 30 dimensions. Each algorithm is executed 50 times for each benchmark function. For each problem there are three measures: 1) the first row is the percentage of success rate, 2) the second is the mean and standard deviation of the best fitness value reached by the algorithms and 3) the third row is the mean and standard deviation of the number of evaluations of function.

In addition, the difference of the algorithms performance is verified by a non-parametric bootstrap hypothesis test with precision of $\alpha = 0.05$. We test the hypothesis that the BEMNA returns a different mean of the best fitness value than the AMaLGaM as well as a different number of function evaluations. So, if the null hypothesis $H_0 : \mu_1 = \mu_2$ is rejected there is statistical evidence to accept differences between both algorithms. This case is marked in bold, as well as the winner of each problem. Observe that the hypothesis test about the fitness value is significant *only if* the desired precision *is not* reached, otherwise it might be meaningless because the algorithm is stopped as soon as it reaches the desired precision. In the same vein, the number of function evaluations only is significant if the algorithm does find the optimum.

Table II-top presents the results of 50 independent executions in 8 unimodal benchmark problems. According to these results we can observe the following evidence:

- Both algorithms reached the precision requested for all test problems. But *there is statistical evidence to say that the BEMNA requires less number of function evaluations than AMaLGaM to reach the desired precision*.
- In 6 out of 8 problems the AMaLGaM has slightly better distance to the optimum fitness than the BEMNA. However this might not be of great importance because both algorithms have already reached the desired precision.
- The BEMNA requires a smaller population size than the AMaLGaM.

50 independent executions in the other 8 benchmark multimodal problems, are shown in Table II-bottom. These present the following evidence:

- Both algorithms resulted in 3 out of 8 problems remaining unsolved: Rastrigin, Drop Wave and Salomon.
- Considering the 5 solved problems, the AMaLGaM has a slightly better success rate in 2 of them, presenting a considerable difference only in one of the problems: Bohachevsky.
- The BEMNA reaches slightly better fitness values in 3 out of 5 unsolved problems.
- The BEMNA requires fewer evaluations than the AMaLGaM in 4 out of 5 solved problems.

TABLE II. COMPARISON BETWEEN BEMNA AND AMALGAM METHODS IN 30 DIMENSIONS. THE WINNER IS MARKED IN BOLDFACE ACCORDING TO A NON-PARAMETRIC BOOTSTRAP TEST. FURTHER DETAILS IN SECTION V.

$\mathcal{F}$	$BEMNA$	$AMaLGaM$	ρ
$\mathcal{F}_1$	**100.00**	**100.00**	
	8.76e-9±1.02e-9	8.55e-9±1.10e-9	3.20e-1
	5.26e+4±6.99e+2	8.29e+4±1.11e+4	**1.00e-4**
$\mathcal{F}_2$	**100.00**	**100.00**	
	8.39e-9±1.12e-9	**6.59e-9±2.20e-9**	**1.00e-4**
	2.77e+4±2.82e+3	4.86e+4±5.34e+3	**1.00e-4**
$\mathcal{F}_3$	**100.00**	**100.00**	
	8.88e-9±9.72e-10	**8.40e-9±1.20e-9**	**3.09e-2**
	4.25e+4±6.81e+2	7.35e+4±1.16e+4	**1.00e-4**
$\mathcal{F}_4$	**100.00**	**100.00**	
	8.84e-9±9.19e-10	**8.17e-9±1.27e-9**	**3.20e-3**
	5.41e+4±7.60e+2	8.70e+4±1.06e+4	**1.00e-4**
$\mathcal{F}_5$	**100.00**	**100.00**	
	8.89e-9±1.01e-9	**8.17e-9±1.07e-9**	**1.20e-3**
	4.30e+4±9.51e+2	7.44e+4±1.04e+4	**1.00e-4**
$\mathcal{F}_6$	**100.00**	**100.00**	
	9.10e-9±8.05e-10	**8.39e-9±1.12e-9**	**6.00e-4**
	5.70e+4±1.34e+3	9.14e+4±1.14e+4	**1.00e-4**
$\mathcal{F}_7$	**100.00**	**100.00**	
	8.86e-9±1.08e-9	8.50e-9±1.29e-9	1.30e-1
	5.93e+4±1.72e+3	9.29e+4±8.15e+3	**1.00e-4**
$\mathcal{F}_8$	**100.00**	**100.00**	
	9.19e-9±8.00e-10	**8.38e-9±1.20e-9**	**4.00e-4**
	5.91e+4±1.74e+3	1.01e+5±9.88e+3	**1.00e-4**
$\mathcal{F}_9$	98.00	**100.00**	
	7.97e-2±5.64e-1	8.49e-9±1.14e-9	1.19e-1
	1.73e+5±0.00e+0	2.70e+5±1.07e+4	**1.00e-4**
$\mathcal{F}_{10}$	**100.00**	**100.00**	
	9.33e-9±6.06e-10	**9.02e-9±6.58e-10**	**1.67e-2**
	6.96e+4±1.09e+3	1.12e+5±1.01e+4	**1.00e-4**
$\mathcal{F}_{11}$	**100.00**	**100.00**	
	9.08e-9±8.95e-10	**8.34e-9±1.20e-9**	**1.60e-3**
	4.80e+4±9.13e+2	7.87e+4±1.14e+4	**1.00e-4**
$\mathcal{F}_{12}$	**100.00**	**100.00**	
	8.98e-9±8.77e-10	**8.33e-9±1.12e-9**	**1.70e-3**
	3.72e+4±1.09e+3	6.35e+4±8.25e+3	**1.00e-4**
$\mathcal{F}_{13}$	92.00	**100.00**	
	5.85e-2±2.20e-1	8.42e-9±1.16e-9	5.83e-2
	6.82e+4±0.00e+0	7.78e+4±9.00e+3	3.21e-1
$\mathcal{F}_{14}$	0.00	**2.00**	
	1.46e+2±8.33e+0	**1.37e+0±1.68e+0**	**1.00e-4**
	3.00e+5±0.00e+0	3.00e+5±0.00e+0	1.00e+0
$\mathcal{F}_{15}$	0.00	0.00	
	1.33e-1±8.07e-2	2.08e-1±1.09e-1	**1.00e-4**
	3.00e+5±0.00e+0	3.00e+5±0.00e+0	1.00e+0
$\mathcal{F}_{16}$	0.00	0.00	
	1.64e-1±6.29e-2	3.63e-1±8.75e-2	**1.00e-4**
	3.00e+5±0.00e+0	3.00e+5±0.00e+0	1.00e+0

This comparison at fixed 50 dimension problems shows that BEMNA have a better performance than the AMaLGaM in most of the benchmark functions. Despite the differences between both methods, we can conclude that the BENMA does not present any inconvenience to adequately adapt the covariance matrix as demanded by the problem.

VI. CONCLUSIONS

This paper proposes to approximate the Multivariate Normal Density to the Boltzmann density by minimizing the Kullback-Leibler divergence. The first contribution is the derivation of parameter estimators, extending previous related work in one dimension to multidimensional problems [14]. The derived formulae for computing the search distribution use the

objective function as well as a ranking value as linear factors for estimating weighted parameters.

The linear weights avoid prematurely collapsing the probability mass around a single solution, preventing premature convergence. In addition, this fashion of parameter estimation produces a softer change in the structure of the covariance matrix between consecutive generations, in contrast to the exponential weights used in similar approaches [13]. The advantage of using linear weights, even with a fixed β value, is well documented in [14].

We propose a change of variable in order to use a parameter γ, instead of the usual β variable, which is easier to control than the first one. Also, it permits the development of a simple but powerful annealing schedule to control the exploration and exploitation.

One of the most important contributions of this work is the formula in Equation (17)*. Usually EDAs intend to estimate a parametric probability distribution which best fits the data. Our proposal is conceptually different, in the sense that it poses the probability mass in the most promising region by using a mean estimator weighted by the objective function, while the structure of the probability function is oriented to the maximum improvement direction.* The conclusions elucidated from this different point of view are the following:

- While the current population or selected set indicates where the most promising regions are, the difference between the worst individuals to the best ones indicates the direction where the population must move to.
- Most other EDAs are conceptually built on the basis that the structure of the adequate distribution must be the same as that in the current population. We propose that the structure of the adequate distribution *could be* inferred from the current population but *it does not follow the same structure.* The conceptual basis of our proposal is that the best structure must be oriented in agreement with the maximum improvement direction.

Statistical results support that BEMNA is competitive with state of the art algorithms, considering that AMaLGaM has been contrasted with other competitive algorithms as well. Furthermore, the results demonstrate that the BEMNA effectively tackles the Rosenbrock problem, which is not solved by similar algorithms [13] and [14].

Future work will contemplate the proposal of additional enhancement techniques to be applied over the current BEMNA for reducing the population size, as well as the number of function evaluations. Moreover, we will explore new ways to use the ideas developed in this article in other evolutionary algorithms.

REFERENCES

[1] H. Müehlenbein, J. Bendisch, and H. M. Voight, "From Recombination of Genes to the Estimation of Distributions, Continuous Parameters," GMD- Fosrchungszentrum Informationstechnik, 53754 Sankt Augustin, Germany, 1996.

[2] P. Larrañaga, "A Review on Estimation of Distribution Algorithms," in *Estimation of Distribution Algorithms*, ser. Genetic Algorithms and Evolutionary Computation, P. Larrañaga and J. A. Lozano, Eds. Springer US, 2002, vol. 2, pp. 57–100.

[3] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization in Continuous Domains by Learning and Simulation of Gaussian Networks," in *Workshop in Optimization by Building and using Probabilistic Models*, ser. GECCO 2000, Las Vegas, Nevada, USA, 2000, pp. 201–204.

[4] I. Segovia-Dominguez, A. Hernandez-Aguirre, and E. V. Diharce, "The Gaussian Polytree EDA with Copula Functions and Mutations," in *EVOLVE*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2013, vol. 447, pp. 123–153.

[5] W. Dong and X. Yao, "Unified Eigen analysis on Multivariate Gaussian based Estimation of Distribution Algorithms," *Information Sciences*, vol. 178, no. 15, pp. 215–247, 2008.

[6] I. Segovia-Dominguez, A. Hernandez-Aguirre, and S. I. Valdez, "A New EDA by a Gradient-driven Density," in *Parallel Problem Solving from Nature – PPSN XIII*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, Eds. Springer International Publishing, 2014, vol. 8672, pp. 352–361.

[7] J. L. Shapiro, "Diversity Loss in General Estimation of Distribution Algorithms," in *Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 92–101.

[8] J. Grahl, P. A. N. Bosman, and S. Minner, "Convergence Phases, Variance Trajectories, and Runtime Analysis of Continuos EDAs," in *GECCO '07: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 516–522.

[9] H. Mühlenbein, "Convergence Theorems of Estimation of Distribution Algorithms," in *Markov Networks in Evolutionary Computation*, ser. Adaptation, Learning, and Optimization, S. Shakya and R. Santana, Eds. Springer Berlin Heidelberg, 2012, vol. 14, pp. 91–108.

[10] T. Mahnig and H. Mühlenbein, "A New Adaptive Boltzmann Selection Schedule SDS," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 183–190.

[11] H. Mühlenbein, T. Mahnig, and A. O. Rodriguez, "Schemata, Distributions and Graphical Models in Evolutionary Optimization," *Journal of Heuristics*, vol. 5, no. 2, pp. 215–247, 1999.

[12] H. Mühlenbein and T. Mahnig, "The Factorized Distribution Algorithm for Additively Decomposed Functions," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1. IEEE, 1999, p. 759.

[13] C. Yunpeng, S. Xiaomin, and J. Peifa, "Probabilistic Modeling for Continuous EDA with Boltzmann Selection and Kullback-Leibeler Divergence," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 389–396.

[14] S. I. Valdez, A. Hernández-Aguirre, and S. Botello, "A Boltzmann based Estimation of Distribution Algorithm," *Information Sciences*, vol. 236, pp. 126–137, 2013.

[15] A. Ochoa, "Opportunities for Expensive Optimization with Estimation of Distribution Algorithms," in *Computational Intelligence in Expensive Optimization Problems*. Springer, 2010, vol. 2, pp. 193–218.

[16] J. Hu, Y. Wang, E. Zhou, M. C. Fu, and S. I. Marcus, "A Survey of Some Model-based Methods for Global Optimization," in *Optimization, Control, and Applications of Stochastic Systems*. Springer, 2012, pp. 157–179.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, ser. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, 2006.

[18] P. M. Gruber, "Convex Polytopes," in *Convex and Discrete Geometry*, ser. A Series of Comprehensive Studies in Mathematics. Springer Berlin Heidelberg, 2007, vol. 336, pp. 243–351.

[19] P. A. N. Bosman, J. Grahl, and D. Thierens, "Benchmarking Parameter-free AMaLGaM on Functions with and without Noise," *Evolutionary Computation*, vol. 21, no. 3, pp. 445–469, Sept 2013.