CIMAT

Computer Science Department
Center for Research in Mathematics (CIMAT)

A Univariate Boltzmann based Estimation of Distribution Algorithm Using the Natural Gradient for Updating the Parameters

This dissertation is submitted for the degree of
Master of Science in Computing and Industrial Mathematics
by

Mario Iván Jaen Márquez

Advisors
Dr. Arturo Hernández Aguirre
Dr. John A. W. McCall

Guanajuato, Mexico January 2016

Acknowledgements

And I would like to acknowledge ...

Table of contents

List of figures

List of tables

List of Algorithms

Chapter 1

Introduction

Global optimization is concerned with finding the best solution whether it is a single or a set of optimal solutions for a given problem specification. Optimization problems can be either continuous (a.k.a. numerical optimization) or discrete (a.k.a. combinatorial optimization), depending on whether the variables involved are continuous or discrete.

In Black-Box Continuous Optimization the aim is to find an optimal value (minimum or maximum) of a n-dimensional real-parameter function $f : \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}$ that has no accessible analytical form. The black-box scenario means that the only accessible information on f are: the dimension of the problem, the bounds on all variables and the function values of evaluated search points. Mathematically this problem can be defined by:

$$\begin{aligned} \underset{x_1,\ldots,x_n}{\text{minimize}} \quad & f(x_1,\ldots,x_n) \\ \text{subject to} \quad & l_i \leq x_i \leq u_i, \qquad \forall 1 \leq i \leq n \\ & x_i \in \mathbb{R}, \qquad\quad \forall 1 \leq i \leq n \end{aligned} \tag{1.1}$$

These types of problems are commonly found throughout the scientific and engineering fields, however, they are difficult to solve because most of the time there are some limitations in the resources needed, for instance computation time, number of function evaluations, etc.

The challenge of optimizing black-box problems is to develop new algorithms that are able to generate solutions of reasonable quality without taking advantage of the problem structure and employing a reasonable small computational effort.

The term Evolutionary Algorithm (EA) was proposed in 1990 [9] to describe computer-based problem solving systems which use some of the mechanisms of natural evolution (established by Charles Darwin in 1858) as key components in their design and implementation. The terminology used to describe an EA comes from biological terms: the objective function to optimize may be called the *fitness function*, an admissible solution $x \in \Omega$ may be

refered as an *individual* and the collection of such individuals forms the current *population*. The objective function value of each solution may be interpreted as a measure of *fitness* of each individual in the environment (search space).

EAs are applicable to a wide range of problems such as classification, machine learning, optimization, etc. Here we focus on black-box continuous optimization problems however they are not particularly tailored for some application.

With the passing of time, it has been observed that there are at least two common approaches followed by researchers for designing new successfully EAs: 1) taking non-theoretical ideas such as nature-inspired, heuristics and those obtained experimentally which are adjusted in a proper way to understand the behaviour of the proposed algorithms and commonly are justified only by empirical experiments or 2) using solid theoretical insights which are justified by a rigorous mathematical framework.

Most of the EAs of this latter type have been recently categorized in a class known as Model-Based Evolutionary Algorithms (MBEAs). This big group includes those algorithms who share two basic properties: first, the use of a probabilistic model to represent the population of individuals and second, the use of a rule for updating the parameters of the model.

Each of the two characteristics have been analyzed from different point of views. To choose the model for instance, one of them is the use of the Boltzmann distribution to model the population. On the other hand, for the parameter update problem the introduction of the natural gradient has taken relevance in recent years. Both ideas are based on strong theoretical concepts, however, interestingly they have been a little bit analyzed thogether. The work of [22] its the first attempt to follow this path.

In this thesis we continue exploring both ideas from its origins with the objective of designing a new variant algorithm which is simpler and robust in the context of black-box optimization problems. The contents of the thesis is organized as follows: in Chapter 2 the general concept of a MBEA is introduced. Chapter 3 presents a revision of previous work done on MBEAs which use a model based on the Boltzmann distribution. Chapter 4 describes the related work done on MBEA which uses the natural gradient for the parameter update. Chapter 5 introduces the proposal of this work, mixing the main concepts presented in both ideas to end up with a new univariate EDA called BUMDA-NES. In Chapter 5 BUMDA-NES is tested and compared versus other univariate state of the art algorithms. Finally, in Chapter 8 the discussion and conclusions of this research work are remarked.

Chapter 2

Model-Based Evolutionary Algorithm (MBEA)

The term Model-Based Evolutionary Algorithm (MBEA) [27, 33] refers to a class of EA where an explicit parameterized model of the solution space is used to guide the search process towards the optimum. In this sense, the search is performed by focusing on the evolution of the model itself instead of considering the direct evolution of each candidate solution as usually is done in other kinds of EAs.

The model is the key part of this class of EAs, it includes a stochastic mechanism for generating a set of candidate solutions for each generation. This can be a probabilistic model which is commonly represented by a probability distribution or by graphical models (e.g. Bayesian networks, Markov networks), etc.

Generally speaking, the basic structure of a MBEA involves two separate phases: The first one consists in building a model instance by computing the values of the parameter vector, in the second phase new candidate solutions are generated from the current model and evaluated in the objective function. The idea of this two-step search process, represented in Algorithm 1, is to move the model towards optimum locations by generating a path from the start point as is shown graphically in Figure 2.1.

Algorithm 1: MBEA template

input : $f, n \in \mathbb{N}, Q(x;\theta), \theta_0$

repeat

1.- Sample $x_1,\ldots,x_n \sim Q(x;\theta_t)$

2.- Evaluate $x_1,\ldots,x_n$ on $f \rightarrow f(x_1),\ldots,f(x_n)$

3.- Update parameters θ_{t+1}

until *stopping criterion is met*

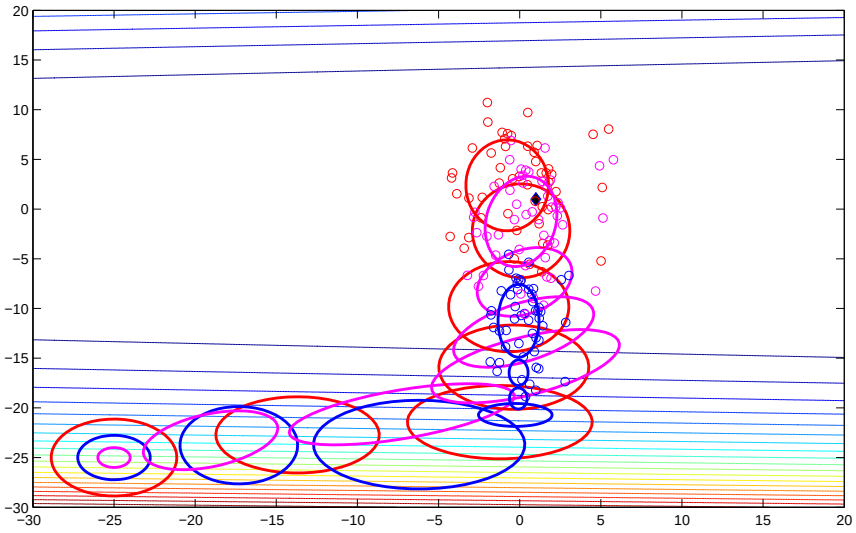


Fig. 2.1 Evolution of the model in three different MBEA on the Rosenbrock function.

Most of the classical EAs such as Genetic Algorithms, Local Search and its variants, Differential Evolution, Particle Swarm Optimization, etc. may be considered instance-based EAs, since they use only the current population to directly generate new candidate solutions.

Basically, the main differences behind every MBEA approach are: 1) which model is chosen to represent the population and 2) how the model parameters are updated every iteration. In fact, the way in which the parameters are updated characterizes the algorithm in a specific MBEA paradigm. Following this idea three different well-known paradigms can be identified: the Estimation of Distribution Algorithm (EDA), the Evolution Strategy (ES) and the Information Geometry Optimization (IGO) approaches, which are briefly described in the next subsections.

2.1 Estimation of Distribution Algorithm (EDA)

The Estimation of Distribution Algorithm (EDA) [14] is a type of MBEA introduced by Muhlenbein and Paab in 1996. From its origin, EDAs have been motivated with some common problems of the Genetic Algorithm (GA): 1) determining the proper parameters for crossover and mutation rates and 2) the difficulty of predicting the movement of individuals in the search space during the optimization process. An EDA maintains selection and variation concepts of traditional EAs, however it replaces the crossover and mutation operators from the simple GA by building and sampling a probabilistic model of solutions.

The main principle of an EDA is to represent each individual $x = \{x_1, x_2, \ldots, x_d\}$ as a set of values which are assumed to be instantiated from a random vector $X = \{X_1, X_2, \ldots, X_d\}$ that has an underlying probability distribution. In this way, an EDA selects a set of candidate solutions which are usually the best ones (e.g. the fittest ones) and, as its name suggest, the aim of an EDA is to estimate the (joint) probability distribution of the selected set to create the model for the next generation. Then the model is sampled to generate the next population as shown in Algorithm 2.

Algorithm 2: EDA approach

input : $f, n \in \mathbb{N}$

$D_0 \leftarrow$ Generate N individuals (initial population) randomly.

repeat

  Evaluate the population in the fitness function $f(x)$.

  $D_t^{Se} \leftarrow$ Select $Se \leq n$ individuals from D_t according to a selection method.

  $\rho(x|D_t^{Se}) \leftarrow$ Estimate the probability distribution of the selected set.

  $D_{t+1} \leftarrow$ Sample N individuals (the new population) from $\rho(x|D_t^{Se})$.

until *stopping criterion is met*

The model in an EDA is created by taking into account the dependencies among the random variables considered. The dependency degree defines the complexity of the model and it is usually categorized in: Univariate, Bivariate or Multivariate. Univariate EDAs assume that each variable of the model is independent therefore, they build univariate marginal models of the solutions variables. Bivariate EDAs assume that at most two variables are interdependent, finally, Multivariate EDAs consider multiple dependencies between variables.

Some common models are: histogram distributions, Gaussian distributions, Gaussian networks, Markov networks, mix of Gaussians, etcetera.

2.2 Evolution Strategies (ES)

Evolution Strategies (ES) [7] are EAs that were developed by Rechenberg and Schwefel in Germany in the 1960s while they were working on applications concerning shape optimization problems. Similar to the Genetic Algorithm, ES are inspired from natural Darwinian evolution, however, they are most commonly applied to continuous search spaces.

The original formulation of ES includes the application of mutation, recombination and selection operators iteratively to move the population of candidate solutions for each generation. The Mutation operator provides the main source of variation to current individuals by adding a perturbation which modifies preliminary candidate solutions. The Recombination

operator selects a set of parents and combines their parts to new solutions to add more variation. The Selection operator induces the bias that gives the evolutionary search direction towards better fitness values.

The mnemonic notation which is usually employed to describe some aspects of the operators in an ES is $(\mu/\rho \overset{+}{,} \lambda)$-ES, where μ indicates the number of individuals in the parent population, ρ represents the parents used for recombination, so $\rho \leq \mu$, λ denotes the number of offspring generated in each iteration and $\overset{+}{,}$ describes the kind of selection applied. "Plus-selection" $(+)$ keeps the best μ individuals from $\mu + \lambda$ while in "Comma-selection" $(,)$ only the offspring survive to the next generation.

Generally, the mutation operator is used as the unique mechanism for inducing diversity in the population (recombination is not commonly used in ES), in fact this operator controls the exploration and exploitation stages of the algorithm by adjusting the so-called strategy parameters or control parameters which define the characteristics of the mutation (e.g. size, orientation, etc.). Adjusting the control parameters to determine a proper mutation mechanism in the current population is key to successfully design ES, this procedure is referred to as self-adaptation of the algorithm.

Algorithm 3 shows the basic template for an ES, D represents the population of individuals, vector $x_k \in \mathbb{R}^d$ is a candidate solution and s_k contains the control parameters.

Algorithm 3: ES approach

input : $\rho, \mu, \lambda \in \mathbb{N}, f$
Initialize $D = \{(x_k, s_k, f(x_k)) | 1 \leq k \leq \mu\}$
repeat
  for $i = 1 \ldots \lambda$ **do**
    $(x_k, s_k) = \text{recombine}(\text{select\_mates}(\rho, D))$
    $s_k \leftarrow \text{mutate\_s}(s_k)$
    $x_k \leftarrow \text{mutate\_x}(s_k, x_k) \in \mathbb{R}^d$
  end
  $D \leftarrow D \cup \{(x_k, s_k, f(x_k)) | 1 \leq k \leq \mu\}$
  $D \leftarrow \text{select\_by\_age}(D)$
  $D \leftarrow \text{select\_}\mu\text{\_best}(\mu, D)$
until *stopping criterion is met*

The model in an ES is regarded as a mutation distribution and for real-valued and unconstrained search spaces the Gaussian distribution has been widely used for many reasons including 1) the flexibility to work with correlated mutations by adjusting the covariances 2) it is the distribution with maximum entropy [8].

2.3 Information Geometry Approaches

Information Geometric Optimization [6] approach is the most recent paradigm in a MBEA. It consists of using a probability distribution as a model and updating its parameters by explicitly taking into account the geometry of the parameter space of probability distributions (i.e. the space where θ lives). This well-defined space is studied by a mathematical research field recently called information geometry, pioneered by S. I. Amari. Roughly speaking, this area suggests considering a non-Euclidean geometry in probability theory by defining a notion of "distance" between two probability distributions.

The novel introduction of IGO is the application of the gradient direction defined in the space of probability distributions, the so-called natural gradient, to minimize an error function by repeatedly taking small steps in parameter space.

Chapter 3

How to choose the model? Approaches using the Boltzmann distribution

As we stated in the previous chapter, the model is the fundamental component of a MBEA. In this chapter the concept of the Boltzmann distribution is presented, as well as the advantages and disadvantages of designing MBEAs by choosing a model based on the Boltzmann distribution. A bibliography review of existing Boltzmann based approaches is also included. Finally a comparative analysis is made of these algorithms.

3.1 The Gibbs-Boltzmann distribution

Inspired by statistical physics the Gibbs-Boltzmann distribution (commonly referred to only as Boltzmann distribution) has been introduced in evolutionary computation to represent the objective function $f(x)$ as a probability distribution. In theory it is a good candidate for a model in MBEAs because the highest probability values are always assigned to global optima regions so that the search towards the best solution is guaranteed (strictly speaking only with $\beta > 0$ values). Its probability density function is defined by:

$$P(x;\beta) = \frac{e^{\beta f(x)}}{\int_{y\in\Omega} e^{\beta f(y)} dy} = \frac{e^{\beta f(x)}}{Z_f(\beta)} \tag{3.1}$$

where $\beta \geq 0$ is the distribution parameter which represents the inverse of the temperature scaled by the Boltzmann constant, that is $\beta = \frac{1}{k_B T}$ (as usually is defined in statistical physics). The term $Z_f(\beta)$ is the normalizing constant also known as the partition function which ensures that the density integrates 1. Lastly, $f(x)$ represents the energy function in

the original distribution, the sign of $f(x)$ indicates if the optimization problem is either a maximization (positive) or minimization (negative) problem.

So, if search points are generated according to the Boltzmann distribution, then points with high fitness values are sampled with higher probability than points with low fitness values as β increases. In the limit case when $\beta \to \infty$ the distribution converges to a uniform distribution on the set of optimal points and 0 elsewhere (non-optimal points), as is shown in Figures 3.1 and 3.2.

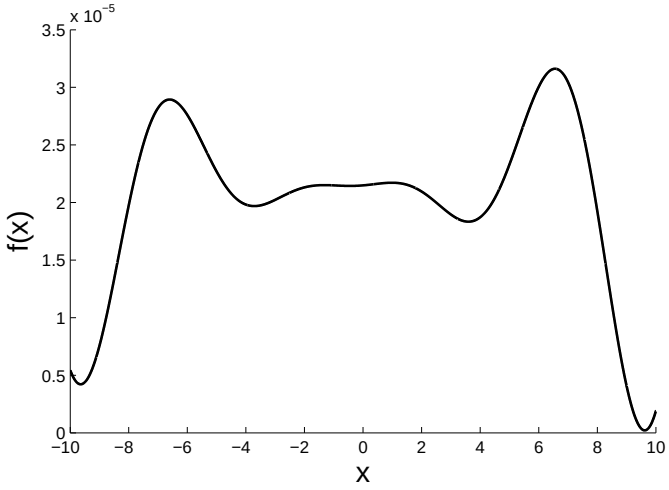


Fig. 3.1 A given function $f(x)$ to maximize.

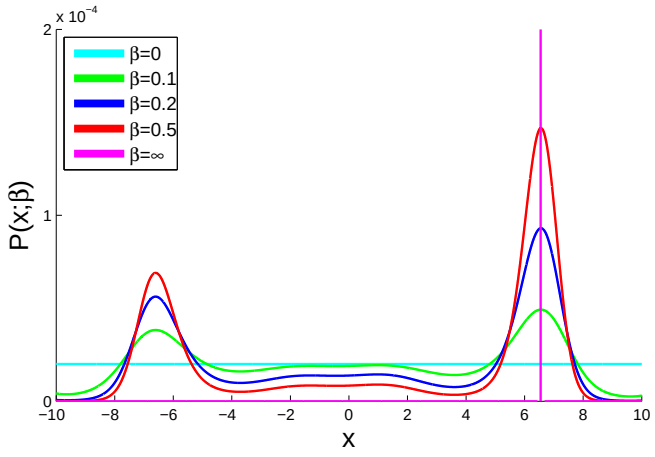


Fig. 3.2 Boltzmann distribution $P(x;\beta)$ associated to $f(x)$ using different β values.

The main problem of using this distribution directly as the model in a MBEA is how to efficiently sample from it. Performing that task is usually impossible because to compute the partition function of 3.1 density is required to know the objective function over the whole

search domain. Even for a relatively small search space, computing $Z_f(\beta)$ by evaluating all possible configurations is computationally infeasible.

In recent years some relevant MBEAs which use the Boltzmann distribution as a mechanism to guide the search towards optimal solutions have been proposed, each one takes a different approach to tackle the sampling problem efficiently. Those approaches are described in next subsections following the chronological order of their publication.

3.2 Boltzmann EDA (BEDA)

Muhlenbein et. al. [18] introduces a conceptual framework for taking advantage of the Boltzmann distribution within an EDA approach for discrete optimization problems. The idea of the proposed algorithm, which was named the Boltzmann Estimation of Distribution Algorithm (BEDA) is to start the search by sampling a Boltzmann distribution with a $\beta = 0$ (which is a uniform distribution) and iteratively updating (increasing) the β value with a certain amount $\Delta\beta_t$ by doing $\beta_{t+1} = \beta_t + \Delta\beta_t$ in order to decrease the variance of the sampling distribution and to converge to the global optimum.

From another point of view, BEDA can be seen as an EDA which incrementally computes the Boltzmann distribution by using Boltzmann selection to control the selection pressure of the population. The BEDA algorithm shown in 4 is a general framework for using the Boltzmann distribution in a MBEA. It has proven convergence (under an infinite number of iterations) however, it is not a practical algorithm because the problem of efficiently computing the distribution $P(x;\beta)$ is not avoided and the annealing schedule for the β parameter is not well-defined.

Algorithm 4: Boltzmann Estimation of Distribution Algorithm (BEDA)

1 $t \Leftarrow 1$. Generate N points according to the uniform distribution $p(\mathbf{x}, 0)$ with $\beta(0) = 0$.
2 **do** {
3 With a given $\Delta\beta(t) > 0$, let
$$p^s(\mathbf{x}, t) = \frac{p(\mathbf{x}, t)e^{\Delta\beta(t)f(\mathbf{x})}}{\sum_{\mathbf{y}} p(\mathbf{y}, t)e^{\Delta\beta(t)f(\mathbf{y})}}.$$
4 Generate N new points according to the distribution $p^s(\mathbf{x}, t)$.
5 $t \Leftarrow t + 1$.
6 } **until** (stopping criterion reached)

3.2.1 Factorized Distribution Algorithm (FDA)

The Factorized Distribution Algorithm (FDA) is an EDA proposed by Muhlenbein et al. [18]. The idea of this proposal is to factorize the Boltzmann distribution to efficiently sample

it assuming that the fitness function is an Additive Descomposable Function (ADF). That is to say, the function can be expressed as a summation of the contributions of the parts of the solution and is represented by a sum of m sub-problems (partial fitness functions) which only depend on a few variables S_i.

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{m} f_i(S_i) \tag{3.2}$$

where $(x_1,\ldots,x_n)$ are problem variables (string positions), f_i is the ith sub-problem, and $S_i \subset \{x_1,x_2,\ldots,x_n\}$ is a sub-solution. Note that because the m sub-solutions S_i are not strictly non-overlapping sets ($S_i \cap S_j \neq \emptyset$ for $i \neq j$), an ADF does not need to be a separable function. If the different functions f_i all depend on different subsets of variables (non-overlapping sets) an ADF is called separable.

Given an ADF the idea in this work is to estimate the Boltzmann distribution using a product of marginal and conditionals probabilities of low order (of dependency) to sample from that approximation easily. An ideal class of distributions to accomplish this goal are the acyclic Bayesian Networks (acBN):

$$q(x) = \prod_{i=1}^{n} q(x_i|pa(x_i)) \tag{3.3}$$

They developed a procedure which uses concepts from the theory of decomposable graphs to build the acBN. The parents are chosen in such a way that interactions in the function definition are represented and that a valid probability density is generated.

3.3 Gradient descent based update rule (Berny)

In Berny's work [16] a general framework for combinatorial optimization is proposed. The idea is to change the problem of searching for a minimum over the set of binary strings into the problem of finding (the parameters of) a probability density function $Q(x;\theta)$ which minimizes some statistical criteria over the set of discrete distributions.

He proposed to use the two following criteria: 1) the simple expectation of the fitness function relative to a distribution $\mathbb{E}_Q[f(x)]$ and 2) the Kullback-Leibler divergence between a distribution and the Boltzmann distribution $D_{KL}(Q(x;\theta) \parallel P(x;\beta))$ at a fixed temperature. As a result, he introduces two stochastic algorithms which iteratively update density parameters by minimizing each of two statistical criteria using gradient techniques.

Following the scope of this chapter the Berny's proposal is described focusing only on the Boltzmann based algorithm developed, however, in Chapter 5 the relevance of the first

criteria is analyzed as well. It is important to notice that Berny uses a Boltzmann distribution parameterized by T, however in order to preserve our notation $\beta = 1/T$ is used instead.

A discrete form of the Kullback-Leibler divergence and Boltzmann distribution is used (replacing integrals by sums), under the assumptions that: 1) the temperature parameter is β and 2) $\sum_{x \in S} Q(x;\theta) = 1$ where S is the set of all possible bit strings x. Replacing $P(x;\beta)$ in Kullback-Leibler definition they get:

$$\begin{aligned} KLD(Q(x;\theta) \parallel P(x;\beta)) &= \sum_x Q(x;\theta) \log \frac{Q(x;\theta)}{P(x;\beta)} dx && (3.4)\\ &= \log Z - \beta \sum_x f(x) Q(x;\theta) + \sum_x Q(x;\theta) \log Q(x;\theta) && (3.5)\\ &= \log Z + \beta \left(- \sum_x f(x) Q(x;\theta) - \frac{1}{\beta} H(Q(x;\theta)) \right) && (3.6)\\ &= \log Z + \beta \left(E - \frac{1}{\beta} H \right) && (3.7) \end{aligned}$$

The quantity in the parentesis $F = E - \frac{1}{\beta} H$ is called the free energy of the system in thermodynamical systems. So that, if a constant temperature is assumed (a fixed Boltzmann distribution), minimizing KLD reduces to minimizing F:

$$\min_\theta KLD(Q(x;\theta) \parallel P(x;\beta)) = \min_\theta \left(E - \frac{1}{\beta} H \right) = \min_\theta F \tag{3.8}$$

To minimize F, the gradient over θ is computed:

$$\begin{aligned} \nabla_\theta F(\theta) &= \nabla_\theta \left(- \sum_x f(x) Q(x;\theta) + \frac{1}{\beta} \sum_x Q(x;\theta) \log Q(x;\theta) \right) && (3.9)\\ &= \mathbb{E}_x \left[\left(-f(x) + \frac{1}{\beta} (1 + \log Q(x;\theta)) \right) \nabla_\theta \log Q(x;\theta) \right] && (3.10) \end{aligned}$$

Then, the model parameters θ are updated by following a gradient descent based update rule using a single solution $x \sim Q(x;\theta)$ and a learning rate α as is shown in Algorithm 5.

$$\Delta\theta = -\alpha \left(-f(x) + \frac{1}{\beta} [1 + \log Q(x;\theta)] \right) \nabla_\theta \log Q(x;\theta) \tag{3.11}$$

Algorithm 5: Gradient descent based update rule (Berny)

input : θ_0 such that $Q(x;\theta_0)$ is uniform over S

repeat

1) Generate a single solution $x \in S$ from $Q(x;\theta)$.

2) Evaluate fitness function $f(x)$

3) Update $\theta \leftarrow \theta + \alpha \left(f(x) - \frac{1}{\beta}[1 + \log Q(x;\theta)] \right) \nabla_\theta \log Q(x;\theta)$

until *stopping criterion is met*

The algorithm has two general parameters: the learning rate α and the temperature value β which are chosen empirically and are not studied in this work. In order to get an instance of this framework a certain distribution $Q(x;\theta)$ has to be specified (with its logarithm and the log-derivative of its parameters). Two specific forms of $Q(x;\theta)$ are considered in this work: 1) Bernoulli distribution and 2) Multivariate Normal distribution (MVN) which lead to two different update rules. Despite being a general framework for global optimization, only binary search spaces are addressed in this work, so that samples generated from a continuous distribution $Q(x;\theta)$ are mapped to the discrete domain of the problem.

Berny extends this idea to continuous optimization problems by taking a population based approach instead of using a single solution every iteration. However, gradients over the Kullback-Leibler divergence to approximate the Boltzmann distribution are no longer considered.

3.4 Stochastic gradient descent based update rule (Gallagher et al.)

Gallagher et. al. [10] introduce in continuous optimization the idea of using a MBEA with a well-known model which is adapted iteratively by approximating the Boltzmann distribution via the Kullback-Leibler divergence. They propose to minimize the Kullback-Leibler divergence between a univariate Gaussian model (parameterized by mean and variance) and the Boltzmann distribution with a fixed temperature $K_{LD}(Q(x;\mu,v) \parallel P(x;\beta))$ over the parameters of μ, v with the idea of performing gradient descent to update them,

$$\min_\theta K_{LD}(Q(x;\theta) \parallel P(x;\beta)) \tag{3.12}$$

Since $Q(x;\theta)$ is the univariate Gaussian and $P(x;\beta)$ is the Boltzmann distribution:

$$Q(x;\theta) = \frac{1}{(2\pi v)^{1/2}} \exp\left(-\frac{(x-\mu)^2}{2v} \right) \tag{3.13}$$

$$\log Q(x;\theta) = -\frac{(x-\mu)^2}{2v} - \log(\sqrt{2\pi v}), \qquad \log P(x;\beta) = -\log Z + \beta f(x) \tag{3.14}$$

Deriving $K_{P,Q}$ with respect to model parameters θ:

$$\begin{aligned}
\frac{\partial K_{Q,P}}{\partial \theta} &= \frac{\partial}{\partial \theta}\left(\int_x Q(x;\theta)\log\frac{Q(x;\theta)}{P(x;\beta)}dx\right) && (3.15)\\
&= \int_x \left[1+\log\frac{Q(x;\theta)}{P(x;\beta)}\right]\frac{\partial}{\partial\theta}Q(x;\theta)dx && (3.16)\\
&= \int_x \left[1-\frac{(x-\mu)^2}{2v} - \beta f(x) - \log(Z\sqrt{2\pi v})\right]\frac{\partial}{\partial\theta}Q(x;\theta)dx && (3.17)
\end{aligned}$$

Taking the derivative of the model with respect to μ:

$$\frac{\partial}{\partial\mu}Q(x;\theta) = Q(x;\theta)\frac{(x-\mu)}{v} \tag{3.18}$$

Substituting expression 3.18 into 3.17 and using the fact that $(x-\mu)$ is odd about μ some integral terms become zero. Finally they get a Monte Carlo approximation as follows:

$$\begin{aligned}
\frac{\partial K_{Q,P}}{\partial \mu} &= \int_x Q(x)\frac{(x-\mu)}{v}\left[1-\frac{(x-\mu)^2}{2v} - \beta f(x) - \log(z\sqrt{2\pi v})\right]dx && (3.19)\\
&= -\frac{\beta}{v}\int_x Q(x;\theta)f(x)(x-\mu)dx && (3.20)\\
&\approx -\frac{\beta}{nv}\sum_{i=1}^{n} f(x_i)(x_i-\mu) && (3.21)
\end{aligned}$$

This work proposes to use the approximation 3.21 to compute mean parameter by following a stochastic gradient descent based rule with a learning rate α proportional to $\frac{\beta}{nv}$:

$$\mu \leftarrow \mu + \alpha \sum_{x_i \in X}(x_i-\mu)f(x_i) \tag{3.22}$$

They developed an iterative algorithm named P_{KLD} shown in 6 which performs stochastic gradient descent over $K_{Q,P}$ with respect to μ. A weakness of this approach is the lack of

a gradient based rule for the variance parameter, so they decided to take the variance as a constant user-defined value instead.

Algorithm 6: P_{KLD}: Stochastic gradient descent based update rule (Gallagher et al.)

input : User-defined variance value: v
repeat
  for $i = 1 \dots n$ **do**
    Generate sample $x_i \sim Q(x; \mu, v)$.
    Evaluate fitness $f(x_i)$
  end
  Update mean: $\mu \leftarrow \mu + \alpha \sum_{i=1}^{n} (x_i - \mu) f(x_i)$
until *stopping criterion is met*

The adjustable parameters are: 1) learning rate α and 2) population size which are not investigated in this approach.

3.5 Boltzmann Gaussian EDA (BGEDA)

Similar to previous approaches, in Yunpeng et. al.'s work [32] the difficulty of computing the exact Boltzmann distribution is circumvented by approximating it with the Gaussian model that minimizes the Kullback-Leibler divergence between both distributions.

However, in contrast to them, three important differences are noted: 1) they choose to minimize the Kullback-Leibler divergence between the Boltzmann distribution and a normal distribution (in that strict sense), that is $D_{KL}(P(x;\beta) \parallel Q(x;\theta))$, 2) considering a temperature-increasing Boltzmann distribution instead of a fixed one and 3) using both a univariate $Q(x;\mu,v)$ and a multivariate $Q(x;\mu,\Sigma)$ Gaussian distribution where their parameters are updated with estimators instead of following gradient based rules.

In this work, they choose to rewrite the Kullback-Leibler divergence as;

$$KLD(P(x;\beta) \parallel Q(x;\theta)) = \int_x P(x;\beta) \log \frac{P(x;\beta)}{Q(x;\theta)} dx \tag{3.23}$$

$$= \int_x P(x;\beta) \log P(x;\beta) dx - \int_x P(x;\beta) \log Q(x;\theta) dx \tag{3.24}$$

$$= -H(P(x;\beta)) - \mathbb{E}_P [\log Q(x;\theta)] \tag{3.25}$$

Thus, the initial minimization problem is reduced to:

$$\min_\theta KLD(P(x;\beta) \parallel Q(x;\theta)) = \max_\theta \mathbb{E}_P [\log Q(x;\theta)] \tag{3.26}$$

Since $Q(x;\theta)$ is a Multivariate Gaussian and $P(x;\beta)$ is the Boltzmann distribution:

$$Q(x;\theta) = \frac{1}{(2\pi)^{d/2}\det(\Sigma)^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{3.27}$$

$$\log Q(x;\theta) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(\det(\Sigma)) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \tag{3.28}$$

The expression 3.26 can be rewritten as:

$$\mathbb{E}_P[\log Q(x;\theta)] = -\frac{1}{2}\int_x P(x;\beta)\log(\det(\Sigma))dx - \frac{1}{2}\int_x P(x;\beta)(x-\mu)^T\Sigma^{-1}(x-\mu)dx \tag{3.29}$$

Computing derivatives with respect to distribution parameters, they get:

$$\frac{\partial}{\partial\mu}\mathbb{E}_P[\log Q(x;\theta)] = -\int_x P(x;\beta)\Sigma^{-1}(x-\mu)dx \tag{3.30}$$

$$\frac{\partial}{\partial\Sigma}\mathbb{E}_P[\log Q(x;\theta)] = -\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\int_x P(x;\beta)\Sigma^{-1}(x-\mu)(x-\mu)^T\Sigma^{-1}dx \tag{3.31}$$

To perform the parameter update, they chose to compute the Gaussian parameter estimator which best fits the search points by forcing those previous expressions to zero. Actually, the parameters found are the Maximum Likelihood Estimators (MLE) of the mean vector and covariance matrix where points are drawn according $P(x;\beta)$,

$$\mu = \int_x x\ p(x;\beta)dx = \mathbb{E}_P[x] \tag{3.32}$$

$$\Sigma = \int_x (x-\mu)(x-\mu)^T\ p(x;\beta)dx = \mathbb{E}_P[(x-\mu)(x-\mu)^T] \tag{3.33}$$

To avoid sampling from $P(x;\beta)$ they compute approximations of expressions 3.32 and 3.33 by deriving a method to sample from one Boltzmann distribution with higher β from a group of samples drawn from another Boltzmann distribution with lower β (similar to the Boltzmann selection). In this way, the selection pressure of the population is controlled following a temperature-increasing Boltzmann distribution without knowing the exact form of either one. As a result, the parameter estimators for the normal distribution are:

$$\mu \leftarrow \frac{\sum_{i=1}^n x_i\ e^{\Delta\beta\cdot f(x_i)}}{\sum_{i=1}^n e^{\Delta\beta\cdot f(x_i)}} \tag{3.34}$$

$$\Sigma \leftarrow \frac{\sum_{i=1}^{n}(x-\mu)(x-\mu)^T \; e^{\Delta\beta \cdot f(x_i)}}{\sum_{i=1}^{n} e^{\Delta\beta \cdot f(x_i)}} \tag{3.35}$$

The resulting algorithm 7 was named Boltzmann Gaussian Estimation of Distribution Algorithm (BGEDA). It starts with a population sampled from a uniform distribution (a Boltzmann distribution with $\beta = 0$) and iteratively creates a Gaussian model from the current population using expressions 3.34 and 3.35, then, using the new model a new population is simulated. In each iteration the β parameter is changed according to a given rule.

Algorithm 7: Boltzmann Gaussian EDA (BGEDA)

input : $\Delta\beta \geq 0$

1) Generate a population P uniformly distributed.

repeat

2) With $\Delta\beta$ calculate the parameters θ of the Gaussian model $Q(x;\theta)$

3) Sample an offspring population $P_t \sim Q(x;\theta)$.

4) Update the population P_{t+1} with P_t

5) Update $\Delta\beta$

until *stopping criterion is met*

They developed two versions of this algorithm 1) the Boltzmann Gaussian Univariate Marginal Distribution Algorithm (BG-UMDA) which uses a univariate normal distribution parametrized by (μ, v) and 2) the Estimation of Multivariate Normal Algorithm with Boltzmann Selection (EMNA-B) which adopts a multivariate normal distribution parametrized by (μ, Σ). An annealing schedule is also given to determine the proper increment $\Delta\beta$ in the inverse temperature parameter β to adjust the selection pressure every iteration.

3.6 Boltzmann UMDA (BUMDA)

The most recent Boltzmann based approaches in a MBEA are Valdez et al. [20], [28] and Segovia et al. [24] works, both are based on the idea of minimizing the Kullback-Leibler divergence between a Gaussian model (the first one for the univariate case and the latter one for the multivariate case) and the Boltzmann distribution $D_{KL}(Q(x;\theta) \parallel P(x;\beta))$ to get the Gaussian parameter estimators for the update phase.

Related to Gallagher et al. work, Valdez et al. [28] proposal also uses a univariate Gaussian distribution to minimize $K_{Q,P}$ over μ and v parameters. The partial derivatives are obtained using a similar derivation, however, in this work the analytical expression for derivative with respect to the variance was found, then these derivatives are set equal to zero to get parameter estimators like the Yunpeng et al. approach.

$$\frac{\partial K_{Q,P}}{\partial \mu} = -\frac{\beta}{v}\int_x Q(x;\theta)f(x)(x-\mu)dx = 0 \tag{3.36}$$

$$\mu = \frac{\int_x xf(x)Q(x;\theta)dx}{\int_x f(x)Q(x;\theta)dx} = \frac{\mathbb{E}_Q[xf(x)]}{\mathbb{E}_Q[f(x)]} \tag{3.37}$$

$$\begin{aligned}
\frac{\partial K_{Q,P}}{\partial v} &= \int_x \left[1+\log\frac{Q(x;\theta)}{P(x;\beta)}\right] Q(x;\theta)\left(\frac{(x-\mu)^2}{2v^2}-\frac{1}{2v}\right)dx && (3.38)\\
&= \int_x \left[1-\log(z\sqrt{2\pi v})-\frac{(x-\mu)^2}{2v}+\log Z-\beta f(x)\right]\left(\frac{(x-\mu)^2}{2v^2}-\frac{1}{2v}\right)dx && (3.39)\\
&= -\frac{1}{2v}-\frac{\beta}{2v^2}\int_x f(x)(x-\mu)^2Q(x;\theta)dx+\frac{\beta}{2v}\int_x f(x)Q(x;\theta)dx && (3.40)
\end{aligned}$$

$$v = \frac{\int_x f(x)(x-\mu)^2Q(x;\theta)dx}{-\frac{1}{\beta}+\int_x f(x)Q(x;\theta)dx} \tag{3.41}$$

Parameter estimators 3.36 and 3.37 can be approximated with Monte Carlo using the sampled points. In addition, according to the analysis presented in this work about the influence of the term $-\frac{1}{\beta}$ which is part of the variance estimator formula, they conclude that the term must be greater than zero and it becomes irrelevant as β or $f(x)$ values increase. Due to these considerations they set the term to 1, resulting in the following parameter estimators for the univariate Gaussian distribution:

$$\mu = \frac{\sum_{i=1}^n x_i\bar{f}(x_i)}{\sum_{i=1}^n \bar{f}(x_i)} \tag{3.42}$$

$$v = \frac{\sum_{i=1}\bar{f}(x_i)(x_i-\mu)^2}{1+\sum_{i=1}^n \bar{f}(x_i)} \tag{3.43}$$

Furthermore, they propose to use $\bar{f}(x_i) = f(x_i) - f(x_{wrost}) + 1$ instead of $f(x_i)$ directly to avoid inconsistencies in the parameter values. The resulting algorithm 8 is called Boltzmann UMDA (BUMDA) and just requires the population size as the tunable parameter.

Algorithm 8: Boltzmann Univariate Marginal Estimation of Distribution Algorithm

1. Give the parameter and stopping criterion:
   nsample ← Number of individuals to be sample.
   minvar ← minimum variance allowed.
2. Uniformly generate the initial population P_0, set $t = 0$.
3. While $v > minvar$ for all dimensions
   (a) $t \leftarrow t + 1$
   (b) Evaluate and truncate the population according algorithm in Figure 1.
   (c) Compute the approximation to μ and v (for all dimensions) by using the selected set (of size $nselec$), and Equations (6) and (9), as follows:

   $$\mu \approx \frac{\sum_1^{nselec} x_i \bar{g}(x_i)}{\sum_1^{nselec} \bar{g}(x_i)}, \quad v \approx \frac{\sum_1^{nselec} \bar{g}(x_i)(x_i-\mu)^2}{1+\sum_1^{nselec} \bar{g}(x_i)},$$

   where $\bar{g}(x_i) = g(x) - g(x_{nselec}) + 1$.
   Note: the individuals can be sorted to simplify the computation, and $g(x_{nselec})$ is the minimum (for maximization case) objective value of the selected individuals.
   (d) Generate $nsample - 1$ individuals from the new model $Q(x,t)$, and insert the elite individual.
4. Return the elite individual as the best approximation to the optimum.

3.7 Boltzmann EMNA (BEMNA)

In the Segovia et al. proposal [24] previous work is extended to the multivariate case to capture dependencies between variables. However, unlike BUMDA, in this work a different approach is taken to derive the parameter estimators of the Multivariate Normal distribution (MVN) which minimizes the Kullback-Leibler divergence $K_{Q,P}$. This derivation is based on rewriting the $K_{Q,P}$ as:

$$\begin{aligned} KLD(Q(x;\theta) \parallel P(x;\beta)) &= \int_x Q(x;\theta) \log \frac{Q(x;\theta)}{P(x;\beta)} dx && (3.44)\\ &= \int_x Q(x;\theta) \log Q(x;\theta) dx - \int_x Q(x;\theta) \log P(x;\beta) dx && (3.45)\\ &= -H(Q(x;\theta)) - \int_x Q(x;\theta) \log P(x;\beta) dx && (3.46)\\ &= -\frac{1}{2}\log((2\pi e^d)\det\Sigma) - \int_x Q(x;\theta) \log P(x;\beta) dx && (3.47) \end{aligned}$$

The partial derivatives are calculated as follows:

$$\frac{\partial K_{Q,P}}{\partial \mu} = -\int_x \frac{\partial Q(x;\theta)}{\partial \mu} \log P(x;\beta) dx \tag{3.48}$$

$$= -\int_x Q(x;\theta) \left[\Sigma^{-1}(x-\mu)\right] \log P(x;\beta) dx \tag{3.49}$$

$$\frac{\partial K_{Q,P}}{\partial \Sigma} = -\frac{1}{2}\frac{\partial \log(|\Sigma|)}{\partial \Sigma} - \int_x \frac{\partial Q(x;\theta)}{\partial \Sigma} \log P(x;\beta) dx \tag{3.50}$$

$$= -\frac{1}{2}\int_x Q(x;\theta) \left[\Sigma^{-1}(x-\mu)(x-\mu)^T\Sigma^{-1}\right] \log P(x;\beta) dx \tag{3.51}$$

$$+\frac{1}{2}\int_x Q(x;\theta)\Sigma^{-1} \log P(x;\beta) dx - \frac{1}{2}\Sigma^{-1} \tag{3.52}$$

Finally, setting both derivatives equal to zero and solving for μ and Σ respectively, they get estimators for the mean vector and the covariance matrix, similar to the univariate case.

$$\mu = \frac{\mathbb{E}_Q[xf(x)]}{\mathbb{E}_Q[f(x)]} \approx \frac{\sum_{i=1}^n x_i f(x_i)}{\sum_{i=1}^n f(x_i)} \tag{3.53}$$

$$\Sigma = \frac{\mathbb{E}_Q\left[f(x)(x-\mu)(x-\mu)^T\right]}{-1/\beta + \mathbb{E}_Q[f(x)]} \approx \alpha \cdot \sum_{i=1}^n f(x_i)(x_i-\mu)(x_i-\mu)^T \tag{3.54}$$

They also propose an annealing schedule to control (indirectly) the β parameter by using a reparametrization with a parameter easier to adjust, so that the new parameter α is taken as a scaling factor for the covariance matrix. The resulting algorithm 9 was named Boltzmann EMNA (BEMNA).

Algorithm 9: Boltzmann Estimation of Multivariate Normal Algorithm (BEMNA)

1. $t \leftarrow 0$. Initialize N_{pop}, N_{sel}, S_c and α/γ according to the annealing schedule.
2. $P^{(0)} \leftarrow$ Uniformly distributed population, where $P^{(0)} = \{\vec{x}_1, ..., \vec{x}_{N_{sel}}\}$.
3. Evaluate $\mathcal{F}(\vec{x}_i)$ and $g(\vec{x}_i)$.
4. Compute the estimates $\vec{\mu}_*$ and Σ_* of $P^{(t)}$, by equations (11) and (15).
5. $P_S^{(t)} \leftarrow$ Sampling S_c individuals from $Q(x;\vec{\mu}_*,\Sigma_*)$.
6. Evaluate $\mathcal{F}(\vec{x}_j)$ and $g(\vec{x}_j)$ of $P_S^{(t)}$.
7. Update α/γ according to the annealing schedule.
8. $P^{(t+1)} \leftarrow$ The best N_{sel} individuals from $P^{(t)} \cup P_S^{(t)}$.
9. $t \leftarrow t+1$.
10. If stopping criterion is not reached return to step 4.
11. Return the elite individual in $P^{(t)}$ as the best approximation to the optimum.

3.8 Summary

In this chapter some MBEAs that use the Boltzmann distribution to build the model were analyzed. The similarities and differences between all of them are summarized in Table 3.1. After reviewing these existing algorithms two different approaches to address the problem of sampling from the exact Boltzmann distribution can be observed: 1) assuming a certain form of the objective function to reduce the computation of the distribution or 2) approximate it with a well-known distribution which we can sample easily (e.g. Gaussian distribution).

As can be seen in Table 3.1, the most common approach is this latter one, which is done by minimizing the Kullback-Leibler divergence between both distributions. However, it is not clear what is the proper way to perform this task. In all the proposals the following directions are easily distinguished: 1) choose $D_{KL}(Q \parallel P)$ or $D_{KL}(P \parallel Q)$ and update the parameters using 2) gradient based rules or by computing parameter estimators.

As can be observed, gradient based rules were used in the first Boltzmann based approaches to update the distribution parameters, however in the latter cases parameter estimators have been employed instead. One advantage of using parameter estimators rather than gradient based rules to update the model is that there is no need to adjust extra parameters (such as learning rates), which makes the algorithm more robust to different problem conditions.

Publication	Name	Ω	Model $Q(x;\theta)$	Idea	Update equations	β
Mühlenbein et al. (1999)	FDA	D	$P(x;\beta)$	Assume $f(x) = \sum_{i=1}^{m} f_i(x_{s_i})$	$P_\beta = \prod_{i=1}^{m} P_\beta(x_{b_i} \mid x_{c_i})$	I
Berny (2000)	-	D	Any $Q(x;\theta)$	$\min_\theta K_{LD}(Q \parallel P)$	$\theta \leftarrow \theta + \alpha(f(x) - \frac{1}{\beta}[1 + \log Q(x;\theta)])\nabla_\theta \log Q(x;\theta)$	F
Gallagher et al. (2005)	P_{KLD}	C	UVN	$\min_\theta K_{LD}(Q \parallel P)$	$\mu \leftarrow \mu + \alpha \sum_{i=1}^{n} f(x_i)(x_i - \mu)$ $v \leftarrow$ constant	F
Yunpeng et al. (2006)	BG-UMDA	C	UVN	$\min_\theta K_{LD}(P \parallel Q)$	$\mu \leftarrow \frac{\sum_{i=1}^{n} x_i \, e^{\Delta\beta \cdot f(x_i)}}{\sum_{i=1}^{n} e^{\Delta\beta \cdot f(x_i)}}$ $v \leftarrow \frac{\sum_{i=1}^{n} (x-\mu)^2 \, e^{\Delta\beta \cdot f(x_i)}}{\sum_{i=1}^{n} e^{\Delta\beta \cdot f(x_i)}}$	I
	EMNA-B	C	MVN		$\mu \leftarrow \frac{\sum_{i=1}^{n} x_i \, e^{\Delta\beta \cdot f(x_i)}}{\sum_{i=1}^{n} e^{\Delta\beta \cdot f(x_i)}}$ $\Sigma \leftarrow \frac{\sum_{i=1}^{n} (x-\mu)(x-\mu)^T \, e^{\Delta\beta \cdot f(x_i)}}{\sum_{i=1}^{n} e^{\Delta\beta \cdot f(x_i)}}$	
Valdez et al. (2013)	BUMDA	C	UVN	$\min_\theta K_{LD}(Q \parallel P)$	$\mu \leftarrow \frac{\sum_{i=1}^{n} x_i \, f(x_i)}{\sum_{i=1}^{n} f(x_i)}$ $v \leftarrow \frac{\sum_{i=1}^{n} f(x_i)(x_i - \mu)^2}{1 + \sum_{i=1}^{n} f(x_i)}$	F
Segovia et al. (2014)	BEMNA	C	MVN		$\mu \leftarrow \frac{\sum_{i=1}^{n} x_i \, f(x_i)}{\sum_{i=1}^{n} f(x_i)}$ $\Sigma \leftarrow \alpha \frac{\sum_{i=1}^{n} f(x_i)(x_i - \mu)(x_i - \mu)^T}{\sum_{i=1}^{n} f(x_i)}$	I

Table 3.1 Historical development of MBEAs using the Boltzmann distribution. Problem domain Ω: Continuous (C) or Discrete (D), Temperature value β: Fixed (F) or Incremental (I)

Chapter 4

How to update the model? Approaches using the Natural Gradient

The parameter update phase is an important part of a MBEA since it determines how the search is performed over the solution space. Thus, different ways to update the model parameters have been developed so far as we described in Chapter 1. Among them the natural gradient approach, which has recently been introduced to EC, seems to be an interesting approach to deal with the parameter update problem, due to its solid theoretical foundation from Information Geometry field. Therefore, in this chapter the concept of natural gradient is presented as well as the existing approaches based on this idea.

4.1 The Natural Gradient

The natural gradient concept originated from Information Geometry by S. I. Amari in 1998 [5]. It represents a generalized version of the classical (plain) gradient direction (the steepest ascent direction) of a cost function defined in a more general (i.e. non-Euclidean, curved) n-dimensional space also known as Riemannian space.

The term "natural" [3] refers to the notion of taking into account the geometry structure (curvature) of the parameter space associated by the specific cost function in order to represent the steepest ascent direction of the function in that space. This is accomplished by modifying the plain gradient direction according to the intrinsic (local) geometry of the parameter space using a nxn matrix called the Metric tensor which measures the similarity between any two nearby configurations. So the natural gradient of a cost function $J(x)$ is defined as follows:

$$\tilde{\nabla} J(x) = G^{-1}(x) \nabla J(x) \tag{4.1}$$

which Amari proved to be the steepest ascent direction with respect to the curvature G. Where $G(x) = (G_{ij}(x))$ is a positive definite matrix representing the curvature at the point x relating to the components i and j. Thereby, when the parameter space is Euclidean the natural gradient reduces to the plain gradient because $G(\theta) = I$ is the identity matrix.

Prior knowledge of the parameter space surface is required to determine the natural gradient direction for any particular cost function. However, for the case of a parameterized family of probability densities $\{p(x;\theta)|\theta \in \Theta\}$ with parameter space $\Theta \subseteq \mathbb{R}^n$ one important result from Information Geometry is that the proper metric is given by the Fisher Information Metric [4] which is derived by using the Kullback-Leibler divergence to measure the similarity between two nearby configurations i and j of Θ space.

$$\mathbb{F}(\theta) = \mathbb{E}_p \left[(\nabla \log p(x;\theta)) (\nabla \log p(x;\theta))^T \right] \tag{4.2}$$

Therefore, the natural gradient direction of a cost function $J(\theta)$ defined in a parameter space of probability densities is:

$$\tilde{\nabla} J(\theta) = \mathbb{F}^{-1}(\theta) \nabla J(\theta) \tag{4.3}$$

Where $\mathbb{F}(\theta)$ defines the local quadratic approximation of this divergence, and so gives a local equivalent to the KL divergence in some sense. It measures how much the distribution changes if you move the parameters a little bit in a given direction.

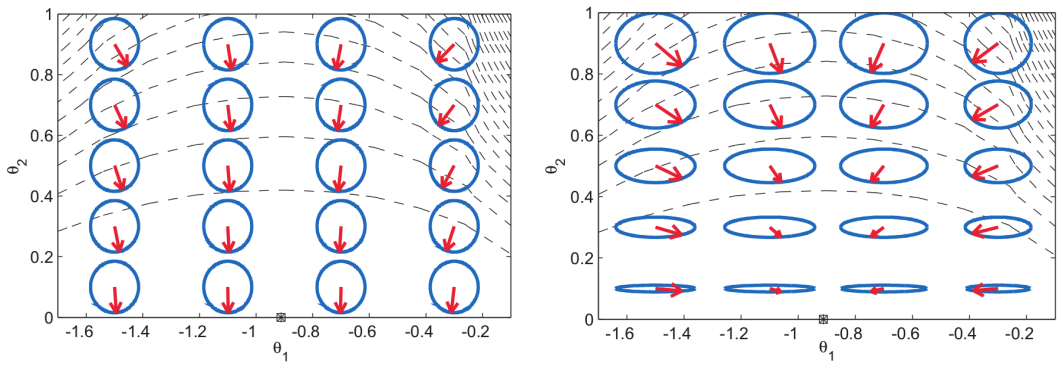
A way to motivate the natural gradient derivation is showing that the plain gradient represents the steepest ascent direction for the Euclidean norm and can be obtained by solving the following constrained optimization problem:

$$\max_{\Delta\theta} J(\theta + \Delta\theta) \quad \text{s. t.} \quad \|\Delta\theta\|_2 \leq \varepsilon \tag{4.4}$$

While natural gradient is the steepest ascent direction considering the natural flow of the geometry in Θ space and can be obtained by solving:

$$\max_{\Delta\theta} J(\theta + \Delta\theta) \quad \text{s. t.} \quad D_{KL}(\theta \parallel \theta + \Delta\theta) \leq \varepsilon \tag{4.5}$$

Graphically, the plain gradient corresponds to the ascent direction on an infinitesimally small hyper-sphere while the natural gradient direction is the ascent direction on an infinitesimally small Fisher hyper-ellipse as is shown in Figure 4.1.



(a) Plain gradient (b) Natural gradient

Fig. 4.1 Geometric interpretation of the plain gradient direction versus the natural gradient on a parameter space. The blue circle in (a) indicates equal distance measured with the Euclidean Metric while the one in (b) shows equal distance measured with the Fisher Information Metric. (Source: [19])

Due to that the Kullback-Leibler is independent of the parametrization taken to represent the distribution $p(x;\theta)$ the natural gradient direction becomes invariant (independent) to the choice of parameter coordinate system.

4.2 Natural Evolution Strategies (NES)

The so-called Natural Evolution Strategy (NES) [31, 29] approach developed in 2008 introduced the idea of using the natural gradient into the EC area. This is a family of optimization algorithms which consider the special case of the natural gradient for Θ-spaces described above (which is denoted by $\tilde{\nabla}_\theta J$) to update the search distribution parameters.

NES uses a parameterized probability distribution $Q(x;\theta)$ which is updated by maximizing a functional $J(\theta)$ every iteration. The cost function employed is the expected value of $f(x)$ under the search distribution which is written in expression 4.6, and is maximized by means of stochastic natural gradient ascent flow on parameters space.

$$J(\theta) = \mathbb{E}_Q[f(x)] = \int_x f(x)\, Q(x;\theta)dx \tag{4.6}$$

In order to get a closed expression for the natural gradient of this cost function with respect to the parameters θ, the plain gradient of $J(\theta)$ and the Fisher Information Metric of $Q(x;\theta)$ has to be computed. The plain gradient direction of $J(\theta)$ which is denoted by ∇_θ is written as:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int_x f(x)\, Q(x;\theta) dx \\
&= \int_x f(x)\, \nabla_\theta Q(x;\theta) dx \\
&= \int_x \Big[f(x)\, \nabla_\theta \log Q(x;\theta) \Big] Q(x;\theta) dx \\
&= \mathbb{E}_Q[f(x)\, \nabla_\theta \log Q(x;\theta)] \qquad (4.7)
\end{aligned}$$

This expectation can be approximated with Monte Carlo using the current sample:

$$\nabla_\theta J(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i)\, \nabla_\theta \log Q(x_i;\theta) \qquad (4.8)$$

The FIM can also be approximated by reusing the log-derivatives already used to compute the plain gradient:

$$\mathbb{F}(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log Q(x_i;\theta) \nabla_\theta \log Q(x_i;\theta)^T \qquad (4.9)$$

Thus, updating the parameters using stochastic natural gradient ascent rather than the plain gradient one lead to the generic version of a NES, as is shown in Algorithm 10. In order to develop a practical algorithm of this family a certain search distribution $Q(x;\theta)$ has to be assumed (i.e. Gaussian, Cauchy, etc).

Algorithm 10: Canonical Natural Evolution Strategies

input : f, θ_0

repeat

  for $i = 1 \ldots n$ **do**

    Draw sample: $x_i \sim Q(x;\theta)$

    Evaluate fitness: $f(x_i)$

    Calculate log-derivatives: $\nabla_\theta \log Q(x_i;\theta)$

  end

  Compute: $\nabla_\theta J(\theta) \leftarrow \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log Q(x_i;\theta) \cdot f(x_i)$

  Compute: $\mathbb{F}(\theta) \leftarrow \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log Q(x_i;\theta) \nabla_\theta \log Q(x_i;\theta)^T$

  Update: $\theta \leftarrow \theta + \eta \cdot \mathbb{F}(\theta)^{-1} \nabla_\theta J$

until *stopping criterion is met*

However, despite having a solid foundation, the original NES algorithm suffers from slow or premature convergence when it is used in practice according to [12]. These problems are due to some instabilities caused in the parameter update rule, such as the empirical FIM may not be invertible, using the fitness value directly to scale the log-gradients, the learning rates η are difficult to control. Consequently, to alleviate some of these common problems some variants to the original algorithm have been developed. The most relevant improvements on NES algorithm are described in the next sections.

4.2.1 Exponential NES (xNES)

Exponential Natural Evolution Strategies (xNES) [12] is the current state of the art algorithm of the NES family. This approach introduces two novel techniques to tackle some of the issues presented in the canonical NES version.

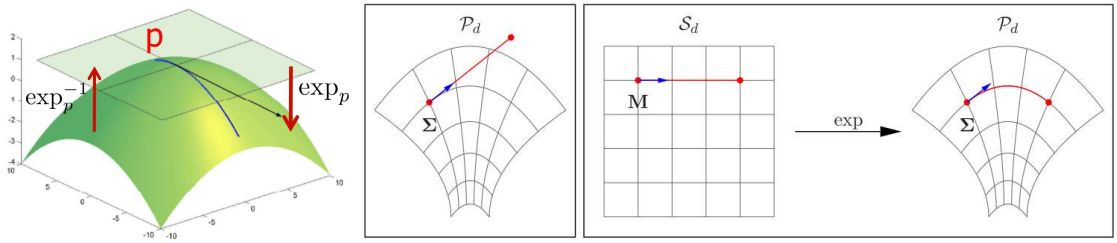
- The first contribution is the use of the exponential parameterization to represent the covariance matrix of the multivariate Gaussian search distribution (which has not been used before in EC area) in order to guarantee *a-priori* that the covariance matrix stays positive-definite after the update phase.

This novel representation of the covariance matrix is based on the exponential map for symmetric matrices [11] which states that *for every symmetric matrix B, the matrix e^B is symmetric positive definite and for every symmetric positive definite matrix A, there is a unique symmetric matrix B such that $A = e^B$.*

If we denote $S_d = \{M \in \mathbb{R}^{dxd}, M^T = M\}$ as the (vector) space of all *dxd* symmetric matrices and $P_d = \{M \in S_d, x^T Mx > 0 \text{ for all } x \in \mathbb{R}^d \setminus \{0\}\}$ as the set (manifold) of all *dxd* positive-definite symmetric matrices respectively, then the exponential map is defined as:

$$\exp : S_d \mapsto P_d, \qquad M \mapsto \sum_{n=0}^{\infty} \frac{M^n}{n!} \tag{4.10}$$

The disadvantage of updating the covariance matrix directly on P_d is that the space P_d is not closed under addition, so that when an additive update is performed directly on the covariance matrix ($\Sigma_{t+1} \leftarrow \Sigma_t + \Delta_t$) the positive-definite constraint can be violated. This well known issue in the covariance matrix can not be avoided working on P_d. Sometimes that problem is circumvented by using repairing methods to fix the negative eigenvalues in the matrix however, an extra computational effort is required to perform this task and more numerical errors are induced in the update.



(a) (b)

Fig. 4.2 Visualizing the exponential mapping between P_d and S_d spaces. The covariance matrix $\Sigma \in P_d$ is represented as $\exp(M)$ with $M \in S_d$ which allows finding a valid covariance matrix after any gradient update direction. Sources ([25], [30])

- The second improvement is a technique to perform a convenient change of coordinates that completely avoids the computation of the FIM (and its inverse) which is required to compute the natural gradient.

This is achieved by performing reparametrizations of θ over time in such a way that the FIM becomes the identity matrix. So that, instead of using the coordinate system of the manifold (global coordinates) to perform the update, an orthogonal basis of parameter space is employed (local coordinates) in such a way that the plain and the natural gradient coincide, as is shown in Figure 4.3.

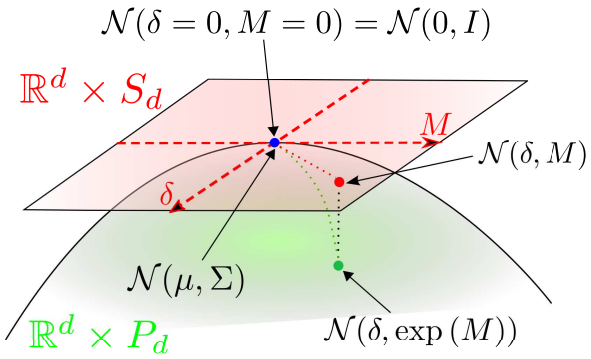


Fig. 4.3 Using the tangent space $T_{(\mu,\Sigma)}(\mathbb{R}^d \times P_d)$ of the parameter manifold $\mathbb{R}^d \times P_d$ (which is isomorphic to the vector space $\mathbb{R}^d \times S_d$) to represent the current search distribution $\mathcal{N}(\mu, \Sigma)$.

An orthogonal local coordinate system (red dashed lines) is used to encode the current search distribution (blue point) for each iteration so that the natural gradient in the (δ, M)-coordinates is equivalent to the plain gradient (the FIM is the identity matrix). By using the local coordinates, the updated search distribution (red point) can be represented in the parameter manifold as $\mathcal{N}(\delta, \exp(M))$ (green point).

Using this change of coordinates the parameter update is performed in local coordinates avoiding the computation of the inverse FIM. After the parameter update the current search distribution encoded as $\mathcal{N}(\mu, \Sigma = AA^T)$ becomes $\mathcal{N}(\delta, \exp(M))$ in local coordinates or $\mathcal{N}(\mu + A\delta, A^T \exp(M)A)$ in global coordinates, which is calculated through the following linear transformation:

$$X = \mathcal{N}(\delta, \exp(M)), \qquad Y = \mu + AX \tag{4.11}$$

$$\mathbb{E}(Y) = \mu + A\delta \tag{4.12}$$

$$\mathrm{Cov}(Y) = A^T \mathrm{Cov}(X)A = A^T \exp(M)A \tag{4.13}$$

So that, the mapping from the local to the global coordinate system can be written as:

$$(\delta, M) \mapsto (\mu_{new}, \Sigma_{new}) = (\mu + A\delta, A^T \exp(M)A) \tag{4.14}$$

On the other hand, if the square root of the covariance matrix A is updated rather than the full covariance matrix, the mapping becomes:

$$(\delta, M) \mapsto (\mu_{new}, A_{new}) = \left(\mu + A\delta, A \exp\left(\frac{1}{2}M\right)\right) \tag{4.15}$$

Using both ideas, the exponential parameterization of the covariance matrix and the local change of coordinates, now the natural gradient of $J(\theta)$ is computed by taking the (δ, M)-parameterization of the MVN, which is:

$$Q(x; \delta, M) = \frac{1}{\sqrt{(2\pi)^d} \det(A) \exp(\frac{1}{2}\mathrm{tr}(M))} \exp\left(-\frac{1}{2}\left\|\exp\left(-\frac{1}{2}M\right) A^{-1}(x - (\mu + A^T\delta))\right\|^2\right) \tag{4.16}$$

As the plain and the natural gradient directions coincide at the point $J(\delta = 0, M = 0)$, the FIM is not necesary anymore. So that closed expressions for $\nabla_\delta J(0,0)$ and $\nabla_M J(0,0)$ are obtained using the log-density of 4.16 and the stochastic approximation as is shown in expression 4.8. The resulting natural gradient formulas for the parameters update take a simple form:

$$\nabla_\delta J \leftarrow \sum_{i=1}^{n} u_i \cdot z_i \tag{4.17}$$

$$\nabla_M J \leftarrow \sum_{i=1}^{n} u_i (z_i z_i^T - I) \tag{4.18}$$

In general, using the fitness values directly to estimate the gradient can introduce undesirable fluctuations in the optimization process depending on the range of the objective function. This problem is reduced by replacing the fitness values $(f(x_i))_{i=1}^n$ by weights $(u_i)_{i=1}^n$ which are obtained according to a relative measure in the fitness values, e.g using the rank of each individual after the fitness evaluation step.

The choice of the weighting function can in fact be seen as a free parameter. For instance,in the first version of NES [31] the following ranking based function $u(x) = i + 5ni^{20}$ was proposed, where i is the relative rank of x_i original fitness values in the population, scaled between $[0,1]$. For this work, they propose to use $u(x) = \frac{\max(0,\log(\frac{n}{2}+1)-\log i)}{\sum_{j=1}^n \max(0,\log(\frac{n}{2}+1)-\log j)} - \frac{1}{n}$.

Assembling all the NES enhancements described above, results in the Algorithm 11 which shows the basic components of xNES algorithm. This algorithm turns out to be better than its predecesors and it constitutes the state-of-the-art approach in the NES family.

Algorithm 11: Exponential Natural Evolution Strategies (xNES)

input : $f, \mu \in \mathbb{R}^d, A \in \mathbb{R}^{dxd}$

repeat

  for $i = 1 \ldots n$ **do**

    $z_i \sim \mathcal{N}(0, I)$

    $x_i = \mu + A \cdot z_i$

    Evaluate fitness: $f(x_i)$

  end

  sort $\{(x_i, z_i)\}$ with respect to $f(x_i)$ and compute the weights u_i

  compute gradients:
$$\nabla_\delta J \leftarrow \sum_{i=1}^n u_i \cdot z_i$$
$$\nabla_M J \leftarrow \sum_{i=1}^n u_i (z_i z_i^T - I)$$

  update parameters:
$$\mu \leftarrow \mu + \eta_\mu A^T \nabla_\delta J$$
$$A \leftarrow A \cdot \exp\left(\frac{\eta_A}{2} \cdot \nabla_M J\right)$$

until *stopping criterion is met*

The tunable parameters of xNES are the population size n, the learning rates η_μ, η_A and the fitness transformation function u_i.

4.2.2 Separable NES (SNES)

After the xNES algorithm one important contribution to the NES family is the introduction of the univariate version of this algorithm, which was called Separable Natural Evolution Strategy [21]. In this variant, the search distribution is restricted to a Gaussian distribution with a diagonal covariance matrix, which corresponds to the following search distribution:

$$Q(x;\theta) = \prod_{i=1}^{d} \mathcal{N}(x_i;\mu_i,\sigma_i) \tag{4.19}$$

Separable NES is shown in algorithm 12. Basically it contains the same steps as in the xNES algorithm, however since the adaptation of each component parameters is independient, the number of parameter in the covariance matrix is reduced from $d(d+1)/2 \in O(d^2)$ to $d \in O(d)$ which is translated in a considerably faster adaptation of algorithm parameters.

Algorithm 12: Separable Natural Evolution Strategies (SNES)

input : $f, \mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^+$

repeat

  for $i = 1 \ldots n$ **do**

    $z_i \sim \mathcal{N}(0, I)$

    $x_i = \mu + \sigma z_i$

    Evaluate fitness: $f(x_i)$

  end

  sort $\{(x_i, z_i)\}$ with respect to $f(x_i)$ and compute utility values u_i

  compute gradients:
  $$\nabla_\delta J \leftarrow \sum_{i=1}^{n} u_i \cdot z_i$$
  $$\nabla_M J \leftarrow \sum_{i=1}^{n} u_i (z_i^2 - 1)$$

  update parameters:
  $$\mu \leftarrow \mu + \eta_\mu \sigma \nabla_\delta J$$
  $$\sigma \leftarrow \sigma \cdot \exp\left(\frac{\eta_\sigma}{2} \cdot \nabla_M J\right)$$

until *stopping criterion is met*

4.3 Summary

This Chapter focuses on describing MBEAs which are based on following the natural gradient direction to update the search distribution parameters. The concept and theory behind the natural gradient was developed in the 1980's however it had not been used before NES approach in EC area. This novel idea introduces some advantages for designing new MBEAs which includes:

- The resulting parameter update is invariant with respect to any particular parameterization of the model. That is because the FIM is computed according to the parametrization chosen to represent the model parameters.
- It allows to design algorithms from a more rigorous perspective and it can lead to a better understanding of the mechanism of some existing approaches.

On the other hand, one of the problems of using this approach is the choice of the proper learning rates to control the size of the updates in each iteration, which affects drastically the performance of the algorithms. The proposals described in this chapter adjust the learning rates by borrowing from related approaches or by empirical investigation.

Furthermore, it is important to notice that the natural gradient has been used only in the context of the ES paradigm. So, it could be interesting to extend this technique to other paradigms as well (i.e. EDAs).

Chapter 5

A Boltzmann based EDA using the natural gradient for the parameter update phase: BUMDA-NES algorithm

This chapter introduces the main contribution of this thesis: a new EDA which was developed by unifying the former mathematical foundations of the two well-known optimization paradigms previously described, the natural gradient concept presented in the NES algorithms and incorporating into it the Boltzmann distribution properties used by some EDAs.

First, the idea of updating the parameters of the search distribution by following a new search direction in parameter space is presented. The proposed direction is derived with the aim to approximate the Boltzmann distribution defined by the objective function.

Next, by assuming a univariate normal distribution as the search distribution closed expressions for the parameters update μ and σ are found. The resulting algorithm called BUMDA-NES is presented as a unifying approach between NES and BUMDA algorithms.

Finally, the behaviour of the β parameter of the Botzmann density takes an interesing role in BUMDA-NES by modifying the traditional parameter update phase presented in NES algorithms. Furthermore, a novel scheme is proposed to adjust this parameter by analyzing the relationship between BUMDA-NES and SNES.

5.1 KLD cost function

The main idea of the NES family of algorithms is to perform optimization on an alternative problem defined in θ-space rather than working on x-space directly. In this way the

search distribution parameters $Q(x;\theta)$ are updated by maximizing a functional following the natural gradient direction in the parameter space.

The cost function, denoted matematically by $J(\theta)$, is useful in this kind of approaches because it is a way to measure how "good" a certain θ value is and it defines the target optimization problem of the algorithm to be solved every iteration. The solution to this problem produces the improvement in the parameters which allows to move the search distribution to sample in optimum locations.

From this point of view, designing a proper cost function in parameter space is key to develop sucessfully MBEAs. As seen in Chapter 4, the common approach in all NES variants developed so far is to employ the expected value of the objective function under the search distribution as the cost function in parameter space, that is $J(\theta) = \mathbb{E}_{Q(x;\theta)}[f(x)]$.

By using that particular cost function the meaning for a good θ value becomes a set of parameters such that the current search distribution location gives better expected fitness in the population than the previous generation. Intuitively, it makes sense to evolve the search distribution using this cost function explicitly because it is similar to the selection method in classical EAs which induces a bias by improving the expected fitness of the current population with respect to the previous one. That is $\mathbb{E}[f(x)]_t \geq \mathbb{E}[f(x)]_{t-1}$, however, in this case the improvement is done implicitly.

In this work we propose to change the direction of the gradient update rule presented in the NES algorithm by defining a different cost function in parameter space. By taking advantage of the Boltzmann distribution properties described in Chapter 3, we use a Boltzmann based cost function where a good θ value gives a search distribution with a better approximation to the Boltzmann distribution of the objective function. The approximation is measured with the Kullback-Leibler divergence as follows:

$$J(\theta) = -K_{LD}(Q(x;\theta) \parallel P(x;\beta)) \tag{5.1}$$

The minus sign in expression 5.1 indicates that it is a maximization problem. Its upper bound is zero, which is the case when both distributions are equal. So that, the more similar the search distribution to the Boltzmann is, a higher divergence value is obtained. In this new cost function $J(\theta)$ we are approximating the search distribution $Q(x;\theta)$ to the Boltzmann PDF in order to focus the search around good fitness values.

In order to interpret how the parameters values of $Q(x;\theta)$ are being ranked according to this new alternative optimization problem, the expression 5.1 is rewritten in terms of the entropy of the search distribution and the expected value of a transformation of the fitness, as presented in the following derivation:

$$
\begin{aligned}
J(\theta) &= -\int_x Q(x;\theta)\log\frac{Q(x;\theta)}{P(x;\beta)}dx \qquad (5.2)\\
&= -\int_x Q(x;\theta)\Big[\log Q(x;\theta)-\log P(x;\theta)\Big]dx\\
&= -\int_x Q(x;\theta)\log Q(x;\theta)dx+\int_x Q(x;\theta)\log P(x;\beta)dx\\
&= H(Q(x;\theta))+\int_x Q(x;\theta)\log P(x;\theta)dx\\
&= H(Q(x;\theta))+\int_x Q(x;\theta)\Big[\beta\cdot f(x)-\log\left(\int_x e^{\beta\cdot f(x)}dx\right)\Big]dx\\
&= H(Q(x;\theta))+\mathbb{E}_Q\Big[\beta\cdot f(x)-\log\left(\int_x e^{\beta\cdot f(x)}dx\right)\Big] \qquad (5.3)
\end{aligned}
$$

In this reformulation, the fitness function appears scaled by the β parameter of the Boltzmann density and the partition function acts as the so-called fitness baseline which is used in [31] and [26] approaches to reduce the estimation variance.

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta H(Q(x;\theta))+\nabla_\theta\mathbb{E}_Q\Big[\beta\cdot f(x)-\log\left(\int_x e^{\beta\cdot f(x)}dx\right)\Big] \qquad (5.4)\\
&= \nabla_\theta H(Q(x;\theta))+\mathbb{E}_Q\Big[\left(\beta\cdot f(x)-\log\left(\int_x e^{\beta\cdot f(x)}dx\right)\right)\nabla_\theta\log Q(x;\theta)\Big]
\end{aligned}
$$

With this change in the cost function, now the idea to guide the search distribution is to perform natural gradient ascent over the Kullback-Leibler divergence to find a set of parameters that locates the search distribution close to the Boltzmann distribution of the objective function. This idea is completely different from the common approaches developed so far.

5.2 Formulae developed

Similar to NES paradigm, where closed expressions for the natural gradient were obtained for that particular cost function. Changing to this new cost function imply to reformulate the natural gradient update formulas by computing first, the plain gradient of $J(\theta)$ with respect to θ and second, the FIM according to the θ parameterization. Furthermore, in order to develop a practical algorithm using the ideas described so far, a univariate normal distribution is assumed as the search distribution of the algorithm, that is $Q(x;\theta)\sim\mathcal{N}(\mu,\sigma)$, therefore:

$$Q(x;\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{5.5}$$

And the entropy of a univariate normal distribution has the following closed form:

$$H(Q(x;\theta)) = \frac{1}{2}\log\left(2\pi e\sigma^2\right) \tag{5.6}$$

Therefore the plain gradient of the cost function $\nabla_\theta J(\theta)$ can also be written as:

$$\begin{aligned} \nabla_\theta J(\theta) &= \frac{1}{2}\nabla_\theta \log(2\pi e\sigma^2) + \beta\nabla_\theta \mathbb{E}_Q[f(x)] - \nabla_\theta \log\left(\int_x e^{\beta\cdot f(x)} dx\right) \\ &= \frac{1}{2}\nabla_\theta \log(2\pi e\sigma^2) + \beta\mathbb{E}_Q[f(x)\nabla_\theta \log Q(x;\theta)] \end{aligned} \tag{5.7}$$

In order to get closed expressions for $\nabla_\theta J(\theta)$ the plain gradients with respect to each parameter, the partial derivatives $\nabla_\mu J(\theta)$ and $\nabla_\sigma J(\theta)$ has to be computed.

First, by using the following equalities:

$$\log Q(x;\theta) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2} \tag{5.8}$$

$$\nabla_\mu \log Q(x;\theta) = \frac{\partial}{\partial\mu}\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{(x-\mu)}{\sigma^2} \tag{5.9}$$

$$\nabla_\sigma \log Q(x;\theta) = -\frac{1}{2}\frac{\partial}{\partial\sigma}\left(\log(2\pi\sigma^2)\right) - \frac{\partial}{\partial\sigma}\left(\frac{(x-\mu)^2}{2\sigma^2}\right) = -\frac{1}{\sigma} + \frac{(x-\mu)^2}{\sigma^3} \tag{5.10}$$

We obtain the 2-dimensional vector corresponding to the plain gradient:

$$\nabla_\theta J(\theta) = \begin{bmatrix} \nabla_\mu J(\theta) \\ \nabla_\sigma J(\theta) \end{bmatrix} = \begin{bmatrix} \frac{\beta}{\sigma^2}\mathbb{E}_Q[f(x)(x-\mu)] \\ \frac{1}{\sigma} - \frac{\beta}{\sigma}\mathbb{E}_Q[f(x)] + \frac{\beta}{\sigma^3}\mathbb{E}_Q[f(x)(x-\mu)^2] \end{bmatrix} \tag{5.11}$$

The inverse of the Fisher Information Matrix (FIM) has to be computed in order to get the natural gradient $\tilde{\nabla}_\theta J(\theta) = \mathbb{F}(\theta)^{-1}\nabla_\theta J(\theta)$. One advantage of assuming a normal distribution on $Q(x;\theta)$ is that the FIM as well as its inverse has a closed form. The exact FIM of the univariate normal distribution using a $\theta = (\mu,\sigma)$ parameterization is:

$$\mathbb{F}(\theta) = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{2}{\sigma^2} \end{bmatrix} \tag{5.12}$$

And because it is a diagonal matrix, the inverse FIM can be calculated by inverting each element of the diagonal:

$$\mathbb{F}(\theta)^{-1} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \frac{\sigma^2}{2} \end{bmatrix} \tag{5.13}$$

Finally, the expression for the natural gradient is:

$$\tilde{\nabla}_\theta J(\theta) = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \frac{\sigma^2}{2} \end{bmatrix} \begin{bmatrix} \frac{\beta}{\sigma^2}\mathbb{E}_Q[f(x)(x-\mu)] \\ \frac{1}{\sigma} - \frac{\beta}{\sigma}\mathbb{E}_Q[f(x)] + \frac{\beta}{\sigma^3}\mathbb{E}_Q[f(x)(x-\mu)^2] \end{bmatrix} \tag{5.14}$$

$$= \begin{bmatrix} \beta\mathbb{E}_Q[f(x)(x-\mu)] \\ \frac{\sigma}{2} - \frac{\sigma\beta}{2}\mathbb{E}_Q[f(x)] + \frac{\beta}{2\sigma}\mathbb{E}_Q[f(x)(x-\mu)^2] \end{bmatrix} \tag{5.15}$$

The resulting formulas for the natural gradient can be approximated with Monte Carlo using the current sample. Furthermore, some terms in the expressions 5.15 y 5.15 can be reduced by writing each observation $x_i \sim \mathcal{N}(\mu, \sigma)$ as $x_i \sim \mu + \sigma z_i$ with $z_i \sim \mathcal{N}(0,1)$ thus, $z_i \sim \frac{x_i - \mu}{\sigma}$ and $z_i^2 \sim \frac{(x_i-\mu)^2}{\sigma^2}$. Finally by considering weights proportional to the fitness instead of using the fitness values directly, the resulting expression for the natural gradient is:

$$\tilde{\nabla}_\theta J(\theta) = \begin{bmatrix} \sigma\beta \sum_{i=1}^{n} u_i \cdot z_i \\ \frac{\sigma}{2} + \frac{\sigma\beta}{2} \sum_{i=1}^{n} u_i \cdot (z_i^2 - 1) \end{bmatrix} \tag{5.16}$$

Employing the natural gradient vector computed above to iterativetly update the parameters of a univariate normal distribution, results in algorithm 13, called BUMDA based NES

(BUMDA-NES). Where $f(x)$ is the fitness function, θ_0 are the initial parameters of the search distribution, n is the population size and d is the problem dimensionality.

Algorithm 13: BUMDA based Natural Evolution Strategies (BUMDA-NES)

input : $d, n \in \mathbb{N}, f : \mathbb{R}^d \to \mathbb{R}, \theta_0 = (\mu_0, \sigma_0)$;

repeat

  for $i = 1 \ldots n$ **do**

    $z_i \sim \mathcal{N}(0, I)$

    $x_i \leftarrow \mu + Bz_i; \quad \mu = \left[\mu^{(1)}, \ldots, \mu^{(d)}\right]^{\mathrm{T}}, B = \begin{bmatrix} \sigma^{(1)} & & \\ & \ddots & \\ & & \sigma^{(d)} \end{bmatrix}$

    evaluate $f(x_i)$

  end

  sort $\{(x_i, z_i)\}$ with respect to $f(x_i)$ and compute weights u_i

  compute gradients

$$\tilde{\nabla}_\mu J \leftarrow \sigma\beta \sum_{i=1}^{n} u_i \cdot z_i$$

$$\tilde{\nabla}_\sigma J \leftarrow \frac{\sigma}{2} + \frac{\sigma\beta}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1)$$

  update parameters

$$\mu \leftarrow \mu + \eta_\mu \tilde{\nabla}_\mu J$$

$$\sigma \leftarrow \sigma + \eta_\sigma \tilde{\nabla}_\sigma J$$

until *stopping criterion is met*

The tunable parameters in BUMDA-NES are: population size n, the utility function u_i, the learning rates η_μ and η_σ and the Boltzmann temperature β value. It is not easy to adjust these parameters, specially to find a proper annealing schedule to adjust the β value during the optimization process. In the next subsection a way to control this parameter is introduced by means of using the relationship of BUMDA-NES with previous approaches.

5.3 Getting a proper annealing schedule for β parameter

As seen in equation 5.16, the β parameter of the Boltzmann distribution is related to the natural gradient computation of BUMDA-NES algorithm. This parameter controls the convergence of the algorithm and it affects both normal distribution parameters: μ and σ. The β parameter can take a fixed value for all the generations, however, to increase robustness it is usually changed over time. Its adaptation process is also known as the annealing schedule.

Choosing an adequate annealing schedule for β is very important to improve the performance of the algorithm, however, it has been a challenging problem since the early Boltz-

mann based approaches. Some common schemes that have been proposed in the literature to control this parameter include:

- Using a gradually increasing value as iterations increase [13] where β is related to the temperature variation of the algorithm.
- To update it according to the actual improvement of the population [23], [24], [32] to control the exploration and exploitation stages in the algorithm.
- Using heuristics that manage the seleccion pressure by proposing a desired increase in the average fitness for the next generation [15],[17].

In a similar way to [1], [2], [12] where is shown how xNES and CMA-ES algorithms are connected in a bidirectional form, in this work it was found out that there is a strong relationship between BUMDA-NES with similar approaches such as SNES and BUMDA. This fundamental link allows to represent BUMDA-NES as a unifying algorithm for univariate NES and Boltzmann based approaches.

In addition, we take advantage of this connection to determine a proper annealing schedule for the β parameter by representing the update equations of SNES as instances of the BUMDA-NES using a specific β value. Following this idea, the proposed annealing schedule is derived by copying the "implicit" annealing schedule presented in SNES algorithm. The closed expression for updating β was obtained by matching the update equations of both algorithms and solving for β, which results in mathematically equivalent parameter update rules.

First, it is important to notice that both BUMDA-NES and SNES algorithms are similar in the sense that they use the same model to guide the search process towards the optimum: a univariate Gaussian search distribution. However, for the parameter update phase, SNES follows the natural gradient direction towards the expected fitness under the search distribution while BUMDA-NES follows the natural gradient direction which minimizes the Kullback-Leibler divergence with respect to the Boltzmann distribution. In Table 6.1 the expressions used for updating the parameters in each algorithm are contrasted.

5.3.1 Determining the β parameter from SNES

The proposed annealing schedule for the β parameter in BUMDA-NES is obtained by relating both parameter update equations presented in the Table 5.1. The aim of this idea is to determine the "hidden" β parameter used in SNES to define a implicit Boltzmann distribution which is approximated by a normal distribution in BUMDA-NES by means of Kullback-Leiber divergence minimization.

	BUMDA-NES	**SNES**
Parameter update equations	$\mu \leftarrow \mu + \eta_\mu \sigma \beta \sum_{i=1}^{n} u_i \cdot z_i$ $\sigma \leftarrow \sigma + \frac{\eta_\sigma \sigma}{2}\left(1 + \beta \sum_{i=1}^{n} u_i \cdot (z_i^2 - 1)\right)$	$\mu \leftarrow \mu + \eta_\mu \sigma \sum_{i=1}^{n} u_i \cdot z_i$ $\sigma \leftarrow \sigma \cdot \exp\left(\frac{\eta_\sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1)\right)$

Table 5.1 Differences between BUMDA and SNES parameter update equations.

The update equations of SNES algorithm are:

$$\mu \leftarrow \mu + \eta_\mu \sigma \sum_{i=1}^{n} u_i \cdot z_i \tag{5.17}$$

$$\sigma \leftarrow \sigma \cdot \exp\left(\frac{\eta_\sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1)\right) \tag{5.18}$$

where u_i are the weights associated to the fitness values $f(x_i)$ and η_μ, η_σ are the learning rates that controls the step size of the gradient direction.

The update equations for the μ parameter are very similar in both algorithms, in fact if we assume a $\beta_\mu = 1$ in BUMDA-NES they become equal. For the σ update equations, because $\exp(x)$ may be defined by the power series:

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \tag{5.19}$$

the exponential term from equation 5.18 can be approximated by its first order Taylor expansion:

$$\exp\left(\frac{\eta_\sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1)\right) \approx 1 + \frac{\eta_\sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1) \tag{5.20}$$

Therefore, the SNES update rule for σ shown in 5.18 becomes approximately:

$$\sigma \leftarrow \sigma + \frac{\eta_\sigma \sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1) \tag{5.21}$$

So that, the σ update rules for BUMDA-NES and SNES shown in **??** and 5.21 respectively are equal if and only if the following equality holds:

$$1+\beta\sum_{i=1}^{n}u_i(z_i^2-1)=\sum_{i=1}^{n}u_i(z_i^2-1) \tag{5.22}$$

This means that by assuming:

$$\beta_\sigma = 1-\frac{1}{\sum_{i=1}^{n}u_i(z_i^2-1)} \tag{5.23}$$

both σ update becomes (approximately) equal.

Therefore, SNES is minimizing the KLD cost function presented in expression 5.1 using the natural gradient direction as well as BUMDA-NES. The μ parameter is updated by approximating a Boltzmann distribution with a fixed $\beta_\mu = 1$ value while the σ parameter is updated by approximating another Boltzmann distribution with a dynamic β_σ value that changes every iteration according to 5.23, which can be rewriten as:

$$\begin{aligned}
\beta_\sigma &= 1-\frac{\sigma^2}{\sum_{i=1}^{n}u_i\left[(x_i-\mu)^2-\sigma^2\right]} && (5.24)\\
&= 1+\frac{\sigma^2}{\sum_{i=1}^{n}u_i\left[\sigma^2-(x_i-\mu)^2\right]} && (5.25)\\
&= 1+\frac{\sigma^2}{\sigma^2\sum_{i=1}^{n}u_i-\sum_{i=1}^{n}u_i(x_i-\mu)^2} && (5.26)
\end{aligned}$$

To discover how the annealing schedule induced by SNES behaves from previous expressions is not easy. However, by assuming that the weighting function $u(f(x))$ fullfils that $\sum_{i=1}^{n}u_i=1$, the term $\hat{\sigma}^2=\sum_{i=1}^{n}u_i(x_i-\mu)^2$ in formula 5.26 can be interpreted as the weighted variance of the current sample and the annealing schedule formula is reduced to:

$$\beta_\sigma = 1+\frac{\sigma^2}{\sigma^2-\hat{\sigma}^2} \tag{5.27}$$

The behaviour of this annealing schedule is interesting in the sense that it depends on the population and weighted-sample variances at the same time. This fact allows to represent β_σ as a d-dimensional vector $\beta_\sigma=[\beta_\sigma^{(1)},\ldots,\beta_\sigma^{(d)}]$ which assign a different value for each dimension of the problem unlike the previous Boltzmann based approaches described in Chapter 3, where the β parameter was assumed to be a scalar value.

So that, to update the β value in each iteration the current variance is needed, which is calculated using the natural gradient update rule computed in the previous generation. it is worth to mention that in some cases the β value can be negative. The reason of this issue can be related to the (out of the common sense) idea of asigning higher weights to the worst

indivuals than others in order to overcome the problem of the collapse of the population in suboptimal points.

A visual comparison of three generations in BUMDA-NES and SNES is presented in Figure 5.1. As can be observed the Boltzmann distribution changes from a negative β value to a positive one as the algorithms reach the optimum locations. Both algoritms start with the same initial contidion in the Gaussian distributions parameters. As iterations increases each algorithm is adjusting their variance according to the Boltzmann distribution. In the last iteration the Gaussian distribution is concentrated around the modes of the objective function.

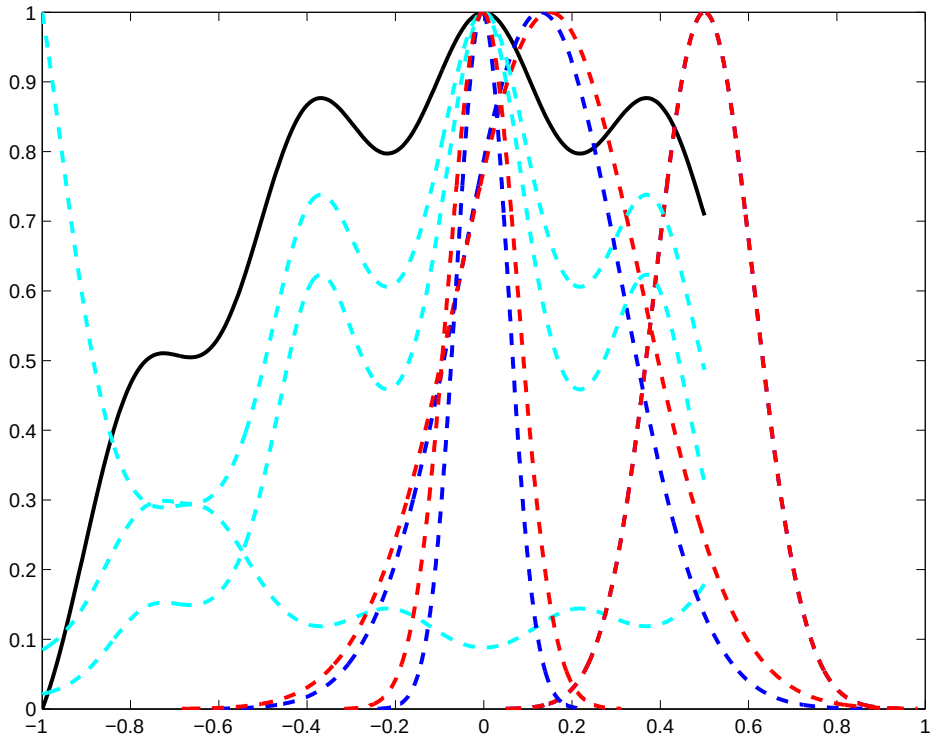


Fig. 5.1 Gaussian distribution of BUMDA-NES (blue) and SNES (red) algorithms using the same initial parameters $\theta_0 : (\mu_0 = 0.5, \sigma_0 = 0.1)$. The objective function (black) and the associated Boltzmann distribution (cyan) indicates the optimum locations

The resulting BUMDA-NES variant using the annealing schedule based on SNES appoach is described in algorithm 14.

Algorithm 14: BUMDA-NES using the annealing schedule based on SNES

input : $d, n \in \mathbb{N}, f : \mathbb{R}^d \to \mathbb{R}, \theta_0 = (\mu_0, \sigma_0)$;

repeat

  for $i = 1 \ldots n$ **do**

    $z_i \sim \mathcal{N}(0, I)$

    $x_i \leftarrow \mu + Bz_i; \quad \mu = \left[\mu^{(1)}, \ldots, \mu^{(d)}\right]^{\mathrm{T}}, B = \begin{bmatrix} \sigma^{(1)} & & \\ & \ddots & \\ & & \sigma^{(d)} \end{bmatrix}$

    evaluate $f(x_i)$

  end

  sort $\{(x_i, z_i)\}$ with respect to $f(x_i)$ and compute weights u_i

  annealing schedule

$$\beta_\mu = 1$$

$$\beta_\sigma = 1 + \frac{\sigma^2}{\sigma^2 - \hat{\sigma}^2} \quad \text{with} \quad \hat{\sigma}^2 = \sum_{i=1}^{n} u_i (x_i - \mu)^2$$

  compute gradients

$$\tilde{\nabla}_\mu J \leftarrow \sigma \beta_\mu \sum_{i=1}^{n} u_i \cdot z_i$$

$$\tilde{\nabla}_\sigma J \leftarrow \frac{\sigma}{2} + \frac{\sigma \beta_\sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1)$$

  update parameters

$$\mu \leftarrow \mu + \eta_\mu \tilde{\nabla}_\mu J$$

$$\sigma \leftarrow \sigma + \eta_\sigma \tilde{\nabla}_\sigma J$$

until *stopping criterion is met*

5.4 Summary

We introduce BUMDA-NES: a univariate Estimation of Distribution Algorithm which is based on the BUMDA and the NES main principles. We consider this new algorithm as a unifying framework which inherits the nice theoretical properties and the well founded mathematical derivation from its predecessors.

BUMDA-NES uses a univariate normal distribution to model the population and its parameters are updated by following the natural gradient in the direction which best approximates the Boltzmann distribution of the objective function. The convenience of assuming a univariate normal model is twofold: first, the complexity of the algorithm is reduced compared with a multivariate model, allowing good scalability as the problem dimensionality increases and second, a closed form of the natural gradient can be obtained without requir-

ing the use of the exponential parameterization. That is because the inverse of the Fisher Information Matrix can be computed exactly.

Finally, this approach adds a new free parameter which is not easy to adjust: the β of the Boltzmann distribution. A proper annealing schedule for this parameter is extracted from other successful approaches such as the SNES algorithm by exploiting the fact that both, BUMDA-NES and SNES can be represented as two approaches derived from the same perspective. This strategy for searching a proper β value in each iteration is simple however it gives a complex behaviour and it impacts the performance of BUMDA-NES as will be observed in the next chapter.

Chapter 6

Experiments

This chapter presents the experiments performed and the results obtained with the aim to evaluate and measure the performance of BUMDA-NES. First, the problem set used to benchmark our algorithm is described. Next, a short analysis is presented to evaluate the impact of some of its parameters such as population size, β annealing schedule, and variances. Finally, the results of comparing BUMDA-NES versus other univariate state of the art algorithms such as SNES and BUMDA are reported.

6.1 Test problems and parameter settings

We empirically validate the performance of our approach by choosing a set of 18 unimodal and multimodal benchmark functions taken from [21], [23], [28] that are commonly used in literature. These functions are described in detail in Table 6.2. All of them are generalized for any number of dimensions. However, in this comparison we report the results obtained only in 2, 5, 10, 20, 40 and 80 dimensions.

Two of state of the art algorithms were selected to compare with BUMDA-NES: BUMDA and SNES. It make sense to contrast our algorithm with respect to these approaches because they are univariate as well. In Table 6.1 the main differences between both BUMDA and SNES are contrasted.

We ran BUMDA-NES, SNES and BUMDA algorithms 30 times on the set of benchmark functions from problem dimensions ranging from 2 to 80 using a target precision of 10^{-8} and $10000d$ maximum number of function evaluations as stopping criteria.

The parameter settings used in BUMDA can be found in [28], however for the functions that are not included in that paper the population size chosen for dimensions from 2 to 40 is 300 and for 80 is 450. The setup for population size and learning rates for SNES is found in [21], the initial parameters $\theta_0 = (\mu_0, \sigma_0)$ for the Gaussian distribution were taken

Characteristic	**BUMDA**	**SNES**
MBEA type	EDA	ES
Weights	$u_i = f(x_i) - f(x_{worst}) + 1$	$u_i = \frac{\max(0, \log(\frac{n}{2}+1) - \log i)}{\sum_{j=1}^{n} \max(0, \log(\frac{n}{2}+1) - \log j)} - \frac{1}{n}$
Population size	Order of $n = 300$	$n = 4 + \lfloor 3\log(d) \rfloor$
Learning rates	Not used	$\eta_\mu = 1$ $\eta_\sigma = \frac{3+\log(d)}{5\sqrt{d}}$
Selection step	Truncation selection method	Not used
Update equations	$\mu \leftarrow \frac{\sum_{i=1}^{n} u_i \cdot x_i}{\sum_{i=1}^{n} u_i}$ $\sigma^2 \leftarrow \frac{\sum_{i=1}^{n} u_i (x_i - \mu)^2}{1 + \sum_{i=1}^{n} u_i}$	$\mu \leftarrow \mu + \eta_\mu \sigma \sum_{i=1}^{n} u_i \cdot z_i$ $\sigma \leftarrow \sigma \cdot \exp\left(\frac{\eta_\sigma}{2} \sum_{i=1}^{n} u_i (z_i^2 - 1) \right)$

Table 6.1 Some relevant differences between BUMDA and SNES algorithms.

as follows: μ_0 is chosen randomly over the search domain and σ_0 is one third the size of the search space.

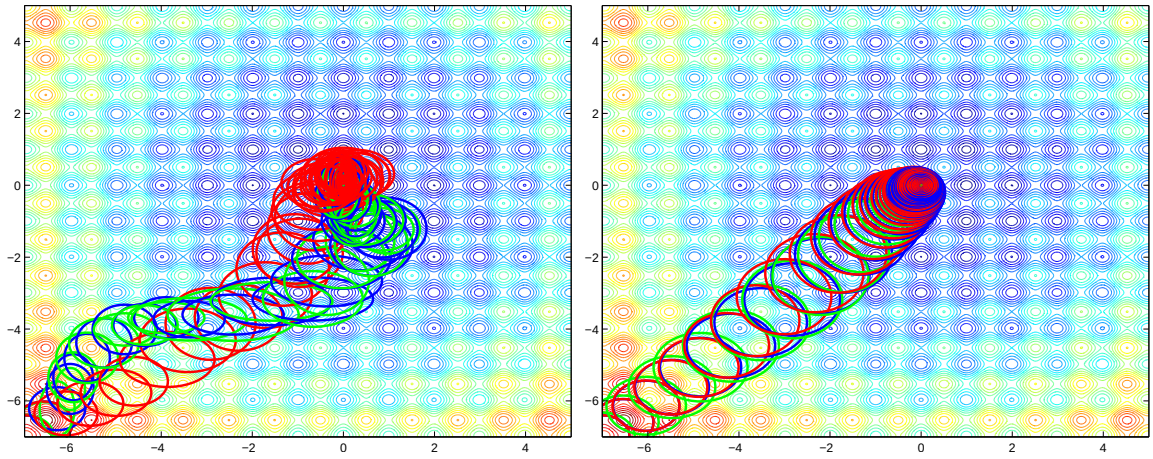
In BUMDA-NES we are using the same population size and learning rates as SNES, the β annealing schedule is the one obtained from SNES, and the weights u_i are taken without the $-\frac{1}{n}$ term in order to fulfill that the sum of all of them is 1.

Name	Type	Definition	Domain	x_i^*	$f(x^*)$
Sphere	U	$\sum_{i=1}^{d} x_i^2$	$[-600,300]^d$	0	0
Ellipsoid	U	$\sum_{i=1}^{d} 10^{6\frac{i-1}{n-1}} x_i^2$	$[-20,10]^d$	0	0
Different Powers	U	$\sum_{i=1}^{d} \lvert x_i \rvert^{2+10\frac{i-1}{d-1}}$	$[-20,10]^d$	0	0
Cigar	U	$x_1^2 + \sum_{i=2}^{d} 10^6 x_i^2$	$[-20,10]^d$	0	0
Tablet	U	$10^6 x_1^2 + \sum_{i=2}^{d} x_i^2$	$[-20,10]^d$	0	0
Cigar Tablet	U	$x_1^2 + \sum_{i=2}^{d-1} 10^4 x_i^2 + 10^8 x_d^2$	$[-20,10]^d$	0	0
Two Axes	U	$\sum_{i=1}^{d/2} 10^6 x_i^2 + \sum_{i=d/2+1}^{d} x_i^2$	$[-20,10]^d$	0	0
Schwefel 1.2	U	$\sum_{i=1}^{d} \left[\sum_{k=1}^{i} x_k\right]^2$	$[-20,10]^d$	0	0
Zakharov	U	$\sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{d} 0.5 i x_i)^2 + (\sum_{i=1}^{d} 0.5 i x_i)^4$	$[-20,10]^d$	0	0
Ackley	M	$20 - 20\exp(-0.2\sqrt{\sum_{i=1}^{n} x_i^2/n}) + \exp(\sum_{i=1}^{n} \cos(2\pi x_i)/n) + \exp(1)$	$[-20,10]^d$	0	0
Rosenbrock	M	$\sum_{i=1}^{d} \left[(1-x_i)^2 + 100(x_{i+1} - x_i^2)^2\right]$	$[-20,10]^d$	1	0
Griewangk	M	$1 + \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i})$	$[-600,300]^d$	0	0
Rastrigin	M	$10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]$	$[-20,10]^d$	0	0
Bohachevsky	M	$\sum_{i=1}^{d-1} [x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7]$	$[-20,10]^d$	0	0
Cosine Mixture	M	$\sum_{i=1}^{d} x_i^2 - 0.1\sum_{i=1}^{d} \cos(5\pi x_i)$	$[-1,0.5]^d$	0	$-0.1d$
Levy 8	M	$\sum_{i=1}^{d-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + \sin^2(\pi y_1) + (y_d - 1)^2$ with $y_i = 1 + 0.25(x_i + 1)$	$[-20,10]^d$	-1	0
Levy Montalvo 1	M	$(\frac{\pi}{d})(10\sin^2(\pi y_1) + (y_d - 1)^2) + (\frac{\pi}{d})\sum_{i=1}^{d-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})]$ with $y_i = 1 + 0.25(x_i + 1)$	$[-20,10]^d$	-1	0
Levy Montalvo 2	M	$0.1(\sin^2(3\pi x_1) + (x_d - 1)^2 [1 + \sin^2(2\pi x_d)]) + 0.1\sum_{i=1}^{d-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})]$	$[-20,10]^d$	1	0

Table 6.2 Name of the function, modality type: U (univariate) and M (multivariate), mathematical definition, search domain and global minimum of the set of 18 benchmark functions to be minimized.

6.2 β parameter analysis

The analysis for the annealing schedule presented in subsection 5.3.1 was performed on the basis on big population sizes because the bigger the population size the more similar both SNES and BUMDA-NES Gaussian distributions are on a typical run, as is contrasted in Figure 6.1. Furthermore, when a big sample is used a more robust natural gradient estimator in parameter space is obtained. So that, the path described by the distribution in the search space tends to be a more consistent route to the global optima location, as is shown in Figure 6.1 (b).



(a) $n = 50$ (b) $n = 5000$

Fig. 6.1 Evolution path generated by the Gaussian distribution of the SNES (red line) and the BUMDA-NES with the annealing schedule based on SNES (blue line). Both algorithms are using the same initial conditions $\theta_0 : \mu_0 = (-7,-7), \sigma_0 = (0.6, 0.6)$, in (b) a much bigger population size is used than (a). The contour lines belong to the 2-d Rastrigin function.

The β variation depends on the population variance σ^2 and the sample weighted variance $\hat{\sigma}^2$. It is important to notice that the β value can be negative when $\hat{\sigma}^2 > \sigma^2$ but also when $\hat{\sigma}^2 < \sigma^2$ is fulfilled. So the β sign depends on the ratio $\gamma = \frac{\hat{\sigma}^2}{\sigma^2}$ as well. In this way, β is negative when $\hat{\sigma}^2 > \sigma^2$ and $\gamma < 2$ or when $\hat{\sigma}^2 < \sigma^2$ and $\gamma > 2$.

In practice it was observed that when the initial variance σ_0^2 of BUMDA-NES is small the condition $\hat{\sigma}^2 > \sigma^2$ is fulfilled only in the initial generations of the algorithm as can be illustrated in Figure 6.2, where the green ellipsoid corresponding to the weighted variance $\hat{\sigma}^2$ is bigger than the blue one in the first 8 generations. In that case the β value is negative because the ratio γ is less than 2, in other words the weighted variance is increased less than two times the population variance. In the latter generations the β value is always positive as can be observed in the left part of Figure 6.3.

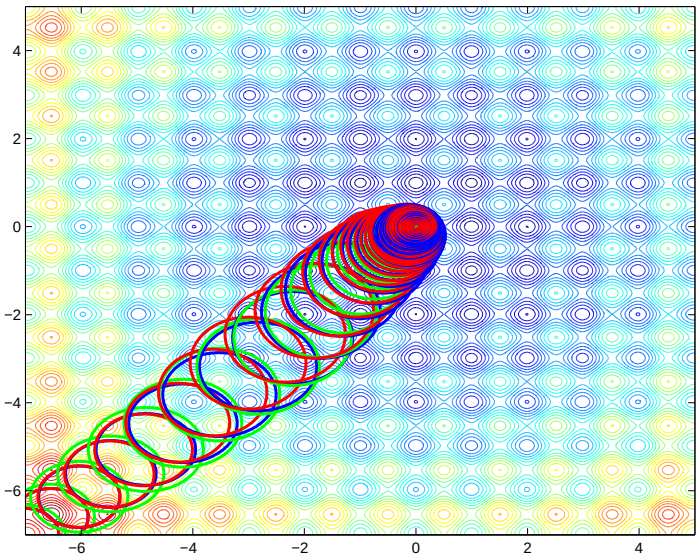


Fig. 6.2 Ellipsoids associated to the Gaussian distributions of BUMDA-NES (blue) and SNES (red) algorithms using the same initial conditions $\theta_0 : \mu_0 = (-7,-7), \sigma_0 = (0.6,0.6)$. The green ellipsoid represents the sample weighted variance $\hat{\sigma}$ of BUMDA-NES. The contour lines belong to the 2-d Rastrigin function.

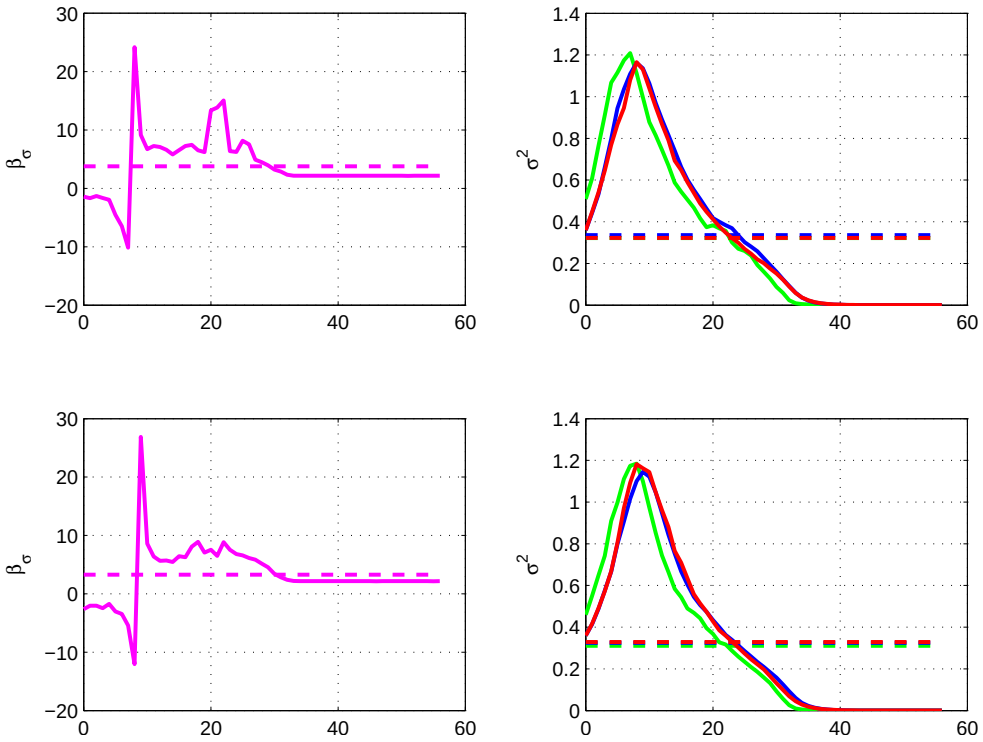


Fig. 6.3 Behaviour of the β parameter of BUMDA-NES using annealing schedule from SNES (left) and the variation of the variances (right) of SNES (red) and BUMDA-NES (blue) in the first dimension (top) and in the second dimension (bottom). The dashed line represents the mean value over all the generations.

However, it is not common to start an EDA using a population with small variance, it is more usual to have a high randomness in the initial solutions to favour the exploration stage and to avoid an early convergence in the algorithm. So that, when σ_0^2 is relatively high the condition $\hat{\sigma}^2 > \sigma^2$ never occurs and the β sign depends only on the γ factor.

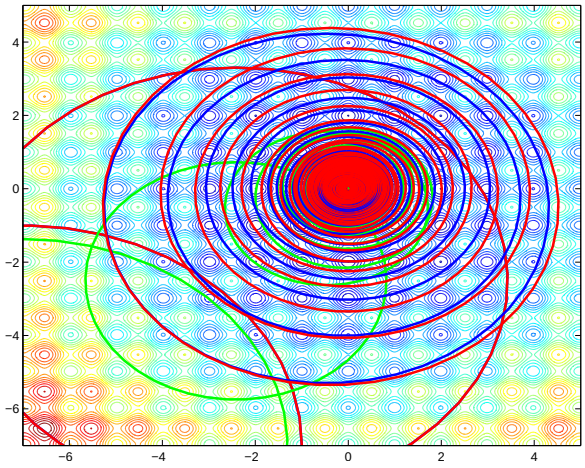


Fig. 6.4 Ellipsoids associated to the Gaussian distributions of BUMDA-NES (blue) and SNES (red) algorithms using the same initial conditions $\theta_0 : \mu_0 = (-7,-7), \sigma_0 = (6,6)$. The green ellipsoid represents the sample weighted variance $\hat{\sigma}$ of BUMDA-NES. The contour line belongs to the 2-d Rastrigin function.

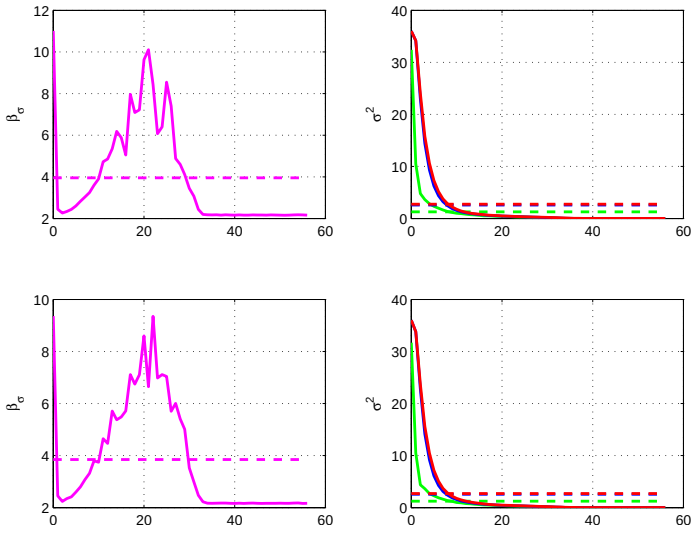


Fig. 6.5 Behaviour of the β parameter of BUMDA-NES using the annealing schedule from SNES (left) and the variation of the variances (right) of SNES (red) and BUMDA-NES (blue) in the first dimension (top) and in the second dimension (bottom). The dashed line represents the mean value over all the generations.

Finally, a third example which illustrates how the β value changes more drastically for each dimension is shown in Figure 6.6.

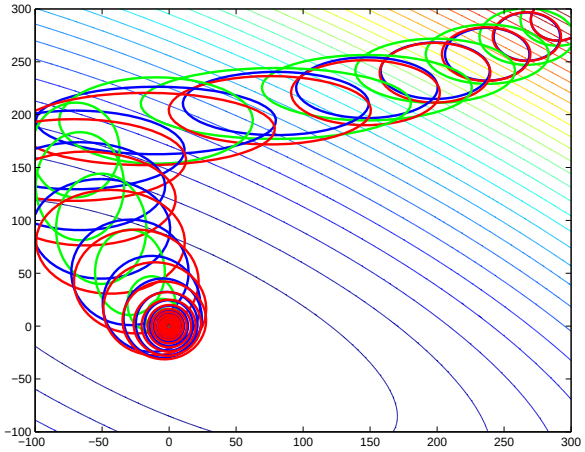


Fig. 6.6 Ellipsoids associated to the Gaussian distributions of BUMDA-NES (blue) and SNES (red) algorithms using the same initial conditions $\theta_0 : \mu_0 = (280, 280), \sigma_0 = (20, 20)$. The green ellipsoid represents the sample weighted variance $\hat{\sigma}$ of BUMDA-NES. The contour lines belong to the 2-d Schwefel 1.2 function.

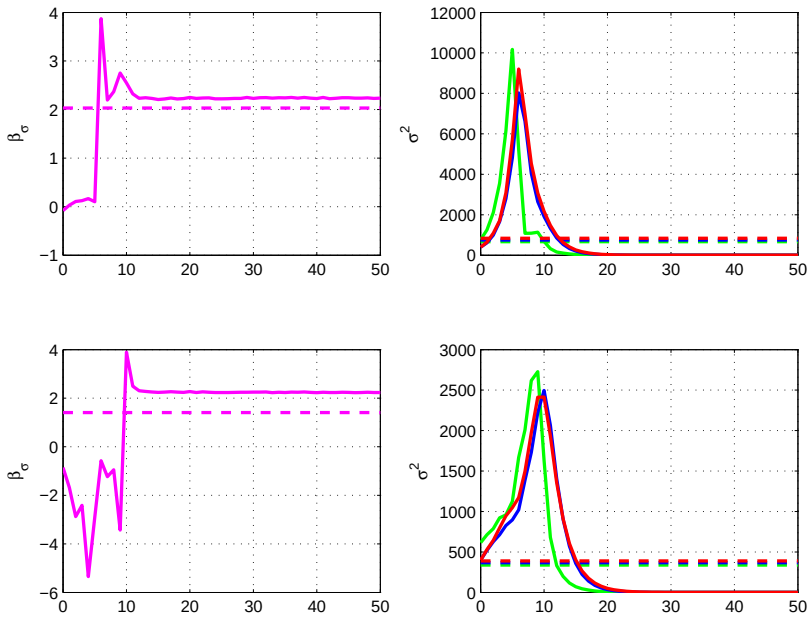


Fig. 6.7 Behaviour of the β parameter of BUMDA-NES using the annealing schedule from SNES (left) and the variation of the variances (right) of SNES (red) and BUMDA-NES (blue) in the first dimension (top) and in the second dimension (bottom). The dashed line represents the mean value over all the generations.

6.3 Experimental results

Tables 6.3, 6.4 and 6.5 summarize the results obtained from the comparison between our approach and the competitors using different problem dimensions. These tables report the percentage of success rate and the mean and standard deviation (over the 30 independent runs) for the best value reached and for the number of function evaluations required for each algorithm in each function of the benchmark set.

As can be seen in these Tables, BUMDA-NES effectively solves most of the univariate problems of the benchmark in low dimensions (2, 5, 10 and 20) using the lower number of fitness evaluations of the three algorithms. However, on higher dimensions (40 and 80) its performance in all the problems is worse than SNES algorithm.

One interesting fact about the performance of BUMDA-NES compared with SNES is that despite having similar update equations for the distribution parameters, the obtained results on this benchmark are sightly different and consistent: BUMDA-NES seems to be better than SNES when the dimensionality of the problem is low and vice versa SNES performed better than BUMDA-NES for dimensions greater than 40.

The explanation of this small but significant difference is that, there are some sources of variation and error which are induced by both algorithms:

- First, in SNES a change in the coordinate system is performed every iteration with the aim to avoid the computation of the inverse of the FIM to estimate the natural gradient direction. On the other hand, the natural gradient computation in BUMDA-NES is not avoiding the calculation of the inverse FIM, in fact we are inverting the exact FIM of the normal distribution model by taking advantage that a closed form for this distribution exists.

- Second, BUMDA-NES seems to be less overfitted to the optimization problem than SNES, because by using the Boltzmann distribution approximation based approach it incorporates an extra parameter which allows us to have a more relaxed problem in parameter space to perform the parameter update, rather than using the simple minimization of the expected fitness values.

- Third, the randomness of the sample induces a difference in the natural gradient approximation, this difference is bigger especially when the sample size is smaller, which is the case for BUMDA-NES and SNES algoritms.

f	BUMDA-NES	SNES	BUMDA
f_1	100.0 4.65e-07±2.76e-07 **4.06e+02±3.89e+01**	100.0 5.25e-07±3.18e-07 1.07e+03±6.22e+01	100.0 **3.86e-07±3.16e-07** 4.54e+03±3.22e+02
f_2	100.0 **3.99e-07±3.13e-07** **4.79e+02±5.41e+01**	100.0 4.73e-07±2.87e-07 1.28e+03±7.24e+01	100.0 4.11e-07±3.01e-07 3.35e+03±2.27e+02
f_3	100.0 3.60e-07±3.24e-07 **2.36e+02±4.91e+01**	100.0 3.05e-07±2.88e-07 5.63e+02±8.33e+01	100.0 **2.81e-07±2.98e-07** 1.51e+03±2.08e+02
f_4	100.0 5.67e-07±2.81e-07 **4.54e+02±3.55e+01**	100.0 5.38e-07±2.54e-07 1.25e+03±9.65e+01	100.0 **4.67e-07±3.16e-07** 3.35e+03±2.29e+02
f_5	100.0 **3.75e-07±2.68e-07** **4.67e+02±3.60e+01**	100.0 5.22e-07±2.84e-07 1.25e+03±9.06e+01	100.0 4.03e-07±3.11e-07 3.32e+03±2.14e+02
f_6	100.0 4.87e-07±3.03e-07 **5.43e+02±9.05e+01**	100.0 5.38e-07±2.64e-07 1.42e+03±9.19e+01	100.0 **4.04e-07±3.41e-07** 3.77e+03±1.75e+02
f_7	100.0 4.66e-07±2.88e-07 **4.52e+02±3.39e+01**	100.0 **4.33e-07±3.06e-07** 1.25e+03±8.37e+01	100.0 4.41e-07±2.41e-07 3.35e+03±2.10e+02
f_8	100.0 4.98e-07±2.66e-07 **3.29e+02±4.14e+01**	100.0 5.26e-07±2.57e-07 8.03e+02±8.62e+01	100.0 **4.81e-07±2.77e-07** 3.96e+03±5.07e+02
f_9	100.0 **3.90e-07±2.76e-07** **3.07e+02±3.86e+01**	100.0 4.92e-07±3.15e-07 8.01e+02±5.90e+01	100.0 4.49e-07±2.97e-07 3.83e+03±3.99e+02
f_{10}	100.0 6.63e-07±2.66e-07 **5.67e+02±4.11e+01**	100.0 6.32e-07±2.06e-07 1.52e+03±1.01e+02	100.0 **5.77e-07±2.77e-07** 7.21e+03±3.65e+02
f_{11}	0.0 1.28e+00±2.90e+00 2.00e+04±0.00e+00	**6.7** 6.12e-01±2.63e+00 **1.87e+04±4.78e+03**	0.0 **2.16e-02±1.84e-02** 2.01e+04±0.00e+00
f_{12}	6.7 6.49e-02±1.02e-01 1.87e+04±4.92e+03	10.0 1.24e-02±1.02e-02 **1.83e+04±5.33e+03**	**46.7** **3.66e-04±8.13e-04** 1.89e+04±1.96e+03
f_{13}	16.7 1.45e+00±1.47e+00 1.68e+04±7.38e+03	33.3 4.60e-01±5.36e-01 1.39e+04±8.78e+03	**100.0** **3.51e-07±2.56e-07** **8.11e+03±9.03e+02**
f_{14}	100.0 4.47e-07±3.05e-07 **3.52e+02±2.88e+01**	100.0 **3.42e-07±2.69e-07** 9.17e+02±1.03e+02	100.0 3.74e-07±2.66e-07 4.24e+03±3.76e+02
f_{15}	93.3 5.34e-03±2.70e-02 1.56e+03±5.01e+03	**100.0** 4.54e-07±2.86e-07 **6.06e+02±5.62e+01**	100.0 **4.25e-07±3.21e-07** 3.01e+03±3.74e+02
f_{16}	100.0 4.94e-07±2.80e-07 **2.69e+02±3.04e+01**	100.0 5.22e-07±3.19e-07 6.72e+02±7.07e+01	100.0 **4.36e-07±2.87e-07** 3.10e+03±3.18e+02
f_{17}	96.7 7.54e-02±4.13e-01 9.72e+02±3.59e+03	**100.0** 4.65e-07±2.73e-07 **8.41e+02±7.77e+01**	100.0 **3.37e-07±2.61e-07** 3.85e+03±2.85e+02
f_{18}	76.7 1.33e-03±3.31e-03 4.94e+03±8.45e+03	96.7 2.26e-06±9.41e-06 **1.49e+03±3.50e+03**	**100.0** **4.05e-07±2.61e-07** 5.11e+03±7.66e+02

(a) dimension 2

f	BUMDA-NES	SNES	BUMDA
f_1	100.0 6.32e-07±2.50e-07 **1.09e+03±6.02e+01**	100.0 6.84e-07±2.27e-07 2.19e+03±7.73e+01	100.0 **5.89e-07±2.49e-07** 9.58e+03±3.14e+02
f_2	100.0 6.70e-07±2.29e-07 **1.20e+03±7.04e+01**	100.0 6.98e-07±1.95e-07 2.38e+03±7.14e+01	100.0 **5.42e-07±2.35e-07** 6.79e+03±2.10e+02
f_3	100.0 5.69e-07±2.51e-07 **5.84e+02±8.37e+01**	100.0 4.69e-07±3.02e-07 1.11e+03±9.73e+01	100.0 **3.58e-07±2.62e-07** 3.04e+03±2.31e+02
f_4	100.0 6.56e-07±2.17e-07 **1.34e+03±7.64e+01**	100.0 **6.19e-07±3.06e-07** 2.66e+03±7.53e+01	100.0 6.90e-07±2.36e-07 1.12e+04±3.58e+02
f_5	100.0 6.52e-07±2.70e-07 **1.11e+03±7.93e+01**	100.0 7.15e-07±2.15e-07 2.20e+03±1.06e+02	100.0 **6.01e-07±2.45e-07** 6.05e+03±2.33e+02
f_6	100.0 7.18e-07±1.99e-07 **1.35e+03±7.44e+01**	100.0 6.93e-07±2.11e-07 2.69e+03±7.46e+01	100.0 **6.69e-07±2.35e-07** 1.10e+04±2.70e+02
f_7	100.0 6.47e-07±2.54e-07 **1.26e+03±8.36e+01**	100.0 6.76e-07±2.11e-07 2.54e+03±8.25e+01	100.0 **5.47e-07±2.51e-07** 7.13e+03±2.00e+02
f_8	**100.0** 8.80e-07±1.49e-07 3.69e+03±4.20e+03	100.0 **6.30e-07±2.77e-07** **2.32e+03±1.09e+02**	3.3 3.63e-03±4.01e-03 4.89e+04±6.74e+03
f_9	86.7 3.19e-02±1.46e-01 8.28e+03±1.67e+04	**100.0** **7.50e-07±2.39e-07** **2.61e+03±1.59e+02**	0.0 4.13e-01±5.90e-01 5.01e+04±0.00e+00
f_{10}	100.0 8.21e-07±1.59e-07 **1.47e+03±5.52e+01**	100.0 8.13e-07±1.84e-07 2.90e+03±8.61e+01	100.0 **7.58e-07±1.76e-07** 1.44e+04±3.26e+02
f_{11}	0.0 1.85e+01±6.12e+01 **5.00e+04±0.00e+00**	0.0 **1.70e-01±7.16e-01** 5.00e+04±0.00e+00	0.0 2.90e+00±1.98e-01 5.01e+04±0.00e+00
f_{12}	0.0 8.19e-02±5.63e-02 5.00e+04±0.00e+00	0.0 4.69e-02±2.86e-02 5.00e+04±0.00e+00	**100.0** **5.43e-07±2.57e-07** **1.55e+04±9.40e+02**
f_{13}	0.0 6.07e+00±5.39e+00 5.00e+04±0.00e+00	3.3 3.55e+00±1.76e+00 4.84e+04±8.51e+03	**100.0** **6.84e-07±2.13e-07** **1.57e+04±1.06e+03**
f_{14}	86.7 9.75e-02±2.79e-01 7.53e+03±1.69e+04	96.7 8.23e-03±4.51e-02 **3.60e+03±8.76e+03**	**100.0** **5.56e-07±2.43e-07** 9.25e+03±2.62e+02
f_{15}	90.0 1.48e-02±4.51e-02 5.63e+03±1.50e+04	**100.0** 6.75e-07±2.25e-07 **1.39e+03±7.47e+01**	100.0 **5.76e-07±2.35e-07** 7.01e+03±2.89e+02
f_{16}	100.0 6.67e-07±1.80e-07 **7.62e+02±5.80e+01**	100.0 6.49e-07±2.23e-07 1.48e+03±7.32e+01	100.0 **6.12e-07±2.47e-07** 7.05e+03±2.81e+02
f_{17}	96.7 4.15e-02±2.27e-01 2.44e+03±8.98e+03	**100.0** 6.63e-07±2.26e-07 **1.53e+03±7.80e+01**	100.0 **5.63e-07±2.54e-07** 7.37e+03±2.69e+02
f_{18}	83.3 2.21e-02±1.13e-01 9.02e+03±1.86e+04	93.3 3.16e-04±1.22e-03 **4.91e+03±1.23e+04**	**100.0** **6.47e-07±2.26e-07** 8.71e+03±3.65e+02

(b) dimension 5

Table 6.3 Comparing BUMDA-NES, SNES and BUMDA using the set of benchmark functions. For each function three characteristics are reported: 1) the success rate over the 30 runs (first row), 2) the average best value reached (second row) and 3) the average number of fitness evaluations (third row). The best value of each function is boldfaced according to each characteristic.

f	BUMDA-NES	SNES	BUMDA
	100.0	100.0	100.0
f_1	7.29e-07±1.61e-07	8.09e-07±1.75e-07	**7.25e-07±1.73e-07**
	2.41e+03±9.38e+01	3.78e+03±1.12e+02	1.54e+04±1.95e+02
	100.0	100.0	100.0
f_2	8.06e-07±1.01e-07	7.78e-07±1.43e-07	**7.40e-07±1.82e-07**
	2.58e+03±8.78e+01	4.06e+03±9.70e+01	1.07e+04±1.75e+02
	100.0	100.0	100.0
f_3	6.95e-07±2.12e-07	6.23e-07±2.53e-07	**5.32e-07±2.63e-07**
	1.32e+03±8.26e+01	1.95e+03±9.96e+01	7.51e+03±2.55e+02
	100.0	100.0	100.0
f_4	7.81e-07±1.61e-07	8.14e-07±1.36e-07	**7.55e-07±1.67e-07**
	3.00e+03±1.33e+02	4.63e+03±1.08e+02	1.82e+04±3.29e+02
	100.0	100.0	100.0
f_5	8.02e-07±1.44e-07	8.03e-07±1.48e-07	**7.13e-07±2.52e-07**
	2.22e+03±9.10e+01	3.49e+03±1.11e+02	9.07e+03±1.78e+02
	100.0	100.0	100.0
f_6	7.86e-07±1.69e-07	**7.32e-07±1.69e-07**	7.74e-07±1.70e-07
	2.82e+03±7.13e+01	4.45e+03±7.65e+01	1.71e+04±2.43e+02
	100.0	100.0	100.0
f_7	7.69e-07±2.17e-07	7.79e-07±1.66e-07	**7.35e-07±1.85e-07**
	2.64e+03±1.36e+02	4.15e+03±9.67e+01	1.65e+04±2.23e+02
	0.0	**100.0**	0.0
f_8	7.28e-03±2.01e-02	**8.66e-07±1.08e-07**	1.76e-01±1.38e-01
	1.00e+05±0.00e+00	**5.93e+03±2.99e+02**	1.00e+05±0.00e+00
	0.0	**100.0**	0.0
f_9	3.46e+01±6.20e+01	**8.33e-07±1.34e-07**	9.38e+00±6.79e+00
	1.00e+05±0.00e+00	**9.85e+03±1.66e+03**	1.00e+05±0.00e+00
	100.0	100.0	100.0
f_{10}	8.93e-07±6.48e-08	9.13e-07±8.63e-08	**8.73e-07±1.08e-07**
	3.14e+03±1.08e+02	4.87e+03±8.00e+01	2.22e+04±3.09e+02
	0.0	0.0	0.0
f_{11}	4.84e+01±1.47e+02	**1.33e-01±1.26e-01**	8.10e+00±7.72e-02
	1.00e+05±0.00e+00	1.00e+05±0.00e+00	1.00e+05±0.00e+00
	26.7	33.3	**100.0**
f_{12}	1.41e-02±1.17e-02	1.26e-02±1.41e-02	**7.30e-07±2.10e-07**
	7.40e+04±4.38e+04	6.80e+04±4.60e+04	**1.70e+04±3.79e+02**
	0.0	0.0	**93.3**
f_{13}	1.25e+01±5.33e+00	6.58e+00±3.03e+00	**6.63e-02±2.52e-01**
	1.00e+05±0.00e+00	1.00e+05±0.00e+00	**3.01e+04±1.92e+04**
	63.3	96.7	**100.0**
f_{14}	2.73e-01±4.75e-01	1.38e-02±7.54e-02	**7.75e-07±1.67e-07**
	3.81e+04±4.79e+04	**6.71e+03±1.76e+04**	1.47e+04±3.24e+02
	60.0	**100.0**	100.0
f_{15}	6.90e-02±1.01e-01	7.78e-07±1.67e-07	**7.59e-07±1.53e-07**
	4.10e+04±4.90e+04	**2.48e+03±7.79e+01**	1.16e+04±2.33e+02
	100.0	100.0	100.0
f_{16}	7.84e-07±1.67e-07	**7.37e-07±1.83e-07**	8.03e-07±1.49e-07
	1.71e+03±9.71e+01	2.64e+03±9.67e+01	1.14e+04±3.15e+02
	93.3	**100.0**	100.0
f_{17}	2.07e-02±7.89e-02	**7.46e-07±1.73e-07**	7.54e-07±1.58e-07
	8.22e+03±2.49e+04	**2.55e+03±8.62e+01**	1.11e+04±2.81e+02
	86.7	96.7	**100.0**
f_{18}	1.47e-03±3.80e-03	3.67e-04±2.01e-03	**7.44e-07±1.95e-07**
	1.49e+04±3.40e+04	**6.06e+03±1.77e+04**	1.28e+04±3.21e+02

(a) dimension 10

f	BUMDA-NES	SNES	BUMDA
	100.0	100.0	100.0
f_1	**8.32e-07±1.35e-07**	8.47e-07±8.77e-08	8.64e-07±1.38e-07
	5.31e+03±1.26e+02	6.52e+03±1.07e+02	2.34e+04±3.05e+02
	100.0	100.0	100.0
f_2	8.51e-07±1.05e-07	8.64e-07±8.74e-08	**8.28e-07±1.29e-07**
	5.65e+03±1.61e+02	6.91e+03±1.33e+02	3.30e+04±4.03e+02
	100.0	100.0	100.0
f_3	8.23e-07±1.47e-07	6.94e-07±2.18e-07	**6.36e-07±2.20e-07**
	2.95e+03±1.96e+02	3.50e+03±1.41e+02	2.43e+04±4.37e+02
	100.0	100.0	100.0
f_4	8.87e-07±7.66e-08	8.67e-07±9.79e-08	**8.22e-07±1.23e-07**
	6.56e+03±1.51e+02	8.10e+03±1.16e+02	3.76e+04±4.69e+02
	100.0	100.0	100.0
f_5	8.61e-07±1.08e-07	8.46e-07±1.26e-07	**8.11e-07±1.14e-07**
	4.60e+03±1.28e+02	5.73e+03±1.09e+02	1.34e+04±2.02e+02
	100.0	100.0	100.0
f_6	8.62e-07±1.01e-07	8.63e-07±8.74e-08	**7.69e-07±1.59e-07**
	6.05e+03±1.43e+02	7.50e+03±1.02e+02	3.48e+04±3.38e+02
	100.0	100.0	100.0
f_7	8.56e-07±9.25e-08	8.32e-07±1.16e-07	**7.94e-07±1.20e-07**
	5.91e+03±2.21e+02	7.11e+03±1.41e+02	4.23e+04±4.04e+02
	0.0	**100.0**	0.0
f_8	4.16e+00±9.01e+00	**9.97e-07±2.81e-09**	2.86e+00±1.24e+00
	2.00e+05±0.00e+00	**5.18e+04±1.11e+04**	2.00e+05±0.00e+00
	0.0	**60.0**	0.0
f_9	3.36e+03±2.39e+03	**1.62e+00±8.43e+00**	7.88e+01±3.10e+01
	2.00e+05±0.00e+00	**1.54e+05±4.55e+04**	2.00e+05±0.00e+00
	100.0	100.0	100.0
f_{10}	9.30e-07±5.10e-08	9.13e-07±7.42e-08	**8.80e-07±8.01e-08**
	6.71e+03±1.60e+02	8.22e+03±1.10e+02	3.36e+04±3.82e+02
	0.0	0.0	0.0
f_{11}	2.99e+01±4.28e+01	**3.90e+00±6.35e+00**	1.81e+01±7.58e-02
	2.00e+05±0.00e+00	2.00e+05±0.00e+00	2.00e+05±0.00e+00
	80.0	80.0	**100.0**
f_{12}	1.97e-03±4.11e-03	2.46e-03±5.82e-03	**8.07e-07±1.29e-07**
	4.39e+04±7.94e+04	4.48e+04±7.89e+04	**2.38e+04±2.99e+02**
	0.0	0.0	**36.7**
f_{13}	2.96e+01±1.06e+01	1.76e+01±5.44e+00	**8.29e-01±7.43e-01**
	2.00e+05±0.00e+00	2.00e+05±0.00e+00	**1.40e+05±8.01e+04**
	33.3	90.0	**100.0**
f_{14}	8.37e-01±8.06e-01	1.05e-01±3.20e-01	**8.09e-07±1.51e-07**
	1.35e+05±9.35e+04	2.55e+04±5.91e+04	**2.25e+04±3.34e+02**
	36.7	86.7	**100.0**
f_{15}	1.18e-01±1.19e-01	1.97e-02±5.11e-02	**7.49e-07±1.55e-07**
	1.28e+05±9.63e+04	3.05e+04±6.76e+04	**1.78e+04±3.03e+02**
	100.0	100.0	100.0
f_{16}	8.63e-07±1.13e-07	8.54e-07±1.11e-07	**8.09e-07±1.19e-07**
	3.87e+03±1.64e+02	4.74e+03±1.21e+02	1.76e+04±2.45e+02
	93.3	**100.0**	100.0
f_{17}	1.04e-02±3.95e-02	8.52e-07±1.21e-07	**8.16e-07±1.17e-07**
	1.67e+04±4.98e+04	**4.33e+03±1.06e+02**	1.65e+04±2.72e+02
	83.3	93.3	**100.0**
f_{18}	1.83e-03±4.16e-03	7.33e-04±2.79e-03	**7.45e-07±1.56e-07**
	3.65e+04±7.44e+04	**1.77e+04±4.95e+04**	1.87e+04±3.01e+02

(b) dimension 20

Table 6.4 Comparing BUMDA-NES, SNES and BUMDA using the set of benchmark functions. For each function three characteristics are reported: 1) the success rate over the 30 runs (first row), 2) the average best value reached (second row) and 3) the average number of fitness evaluations (third row). The best value of each function is boldfaced according to each characteristic.

f	BUMDA-NES	SNES	BUMDA
f_1	100.0 9.05e-07±6.58e-08 1.23e+04±1.58e+02	100.0 9.13e-07±6.55e-08 **1.18e+04±1.51e+02**	100.0 **8.78e-07±9.68e-08** 3.50e+04±3.60e+02
f_2	100.0 9.10e-07±7.00e-08 1.28e+04±2.12e+02	100.0 9.20e-07±8.19e-08 **1.25e+04±1.58e+02**	100.0 **8.71e-07±9.56e-08** 4.96e+04±4.25e+02
f_3	100.0 8.42e-07±1.31e-07 6.97e+03±2.67e+02	100.0 8.14e-07±1.14e-07 **6.67e+03±1.68e+02**	100.0 **7.63e-07±1.91e-07** 7.69e+04±1.48e+03
f_4	100.0 9.02e-07±6.72e-08 1.51e+04±2.86e+02	100.0 9.00e-07±8.58e-08 **1.46e+04±1.64e+02**	100.0 **8.33e-07±1.17e-07** 8.52e+04±5.46e+02
f_5	100.0 9.04e-07±8.22e-08 1.02e+04±1.96e+02	100.0 8.96e-07±8.77e-08 **9.95e+03±1.25e+02**	100.0 **8.62e-07±7.80e-08** 2.99e+04±2.45e+02
f_6	100.0 9.30e-07±5.70e-08 1.37e+04±2.74e+02	100.0 9.09e-07±7.54e-08 **1.34e+04±1.54e+02**	100.0 **9.05e-07±6.32e-08** 7.76e+04±4.01e+02
f_7	100.0 8.94e-07±8.76e-08 1.34e+04±3.86e+02	100.0 9.08e-07±7.77e-08 **1.28e+04±1.72e+02**	100.0 **8.60e-07±1.06e-07** 7.66e+04±5.07e+02
f_8	0.0 9.01e+01±7.25e+01 **4.00e+05±0.00e+00**	0.0 **1.30e-02±3.03e-02** 4.00e+05±0.00e+00	0.0 3.08e+01±9.29e+00 4.00e+05±0.00e+00
f_9	0.0 1.18e+04±4.61e+03 **4.00e+05±0.00e+00**	0.0 3.62e+03±3.21e+03 4.00e+05±0.00e+00	0.0 **3.78e+02±1.35e+02** 4.00e+05±0.00e+00
f_{10}	100.0 9.58e-07±3.79e-08 1.50e+04±2.32e+02	100.0 9.49e-07±5.85e-08 **1.46e+04±1.58e+02**	100.0 **9.41e-07±5.40e-08** 4.92e+04±4.16e+02
f_{11}	0.0 4.04e+01±2.56e+01 **4.00e+05±0.00e+00**	0.0 **2.00e+01±1.61e+01** 4.00e+05±0.00e+00	0.0 3.79e+01±7.98e-02 4.00e+05±0.00e+00
f_{12}	96.7 2.47e-04±1.35e-03 2.38e+04±7.10e+04	96.7 2.47e-04±1.35e-03 **2.35e+04±7.11e+04**	**100.0** **8.74e-07±7.53e-08** 3.42e+04±3.20e+02
f_{13}	0.0 6.57e+01±1.78e+01 **4.00e+05±0.00e+00**	0.0 4.54e+01±8.43e+00 4.00e+05±0.00e+00	0.0 **4.51e+00±1.82e+00** 4.00e+05±0.00e+00
f_{14}	13.3 2.00e+00±1.35e+00 3.48e+05±1.34e+05	73.3 3.64e-01±6.56e-01 1.15e+05±1.75e+05	**100.0** **8.56e-07±8.54e-08** **3.35e+04±3.50e+02**
f_{15}	10.0 3.55e-01±2.18e-01 3.61e+05±1.19e+05	43.3 9.85e-02±9.77e-02 2.30e+05±1.97e+05	**100.0** **8.88e-07±9.75e-08** **2.69e+04±2.68e+02**
f_{16}	100.0 9.10e-07±8.56e-08 9.18e+03±2.19e+02	100.0 9.04e-07±6.60e-08 **8.70e+03±1.77e+02**	100.0 **8.40e-07±1.11e-07** 2.66e+04±3.02e+02
f_{17}	93.3 5.18e-03±1.97e-02 3.43e+04±9.94e+04	96.7 2.59e-03±1.42e-02 **2.08e+04±7.16e+04**	**100.0** **8.71e-07±9.08e-08** 2.40e+04±3.05e+02
f_{18}	90.0 1.10e-03±3.35e-03 4.80e+04±1.19e+05	96.7 3.67e-04±2.01e-03 **2.16e+04±7.15e+04**	**100.0** **8.68e-07±8.74e-08** 2.74e+04±3.31e+02

(a) dimension 40

f	BUMDA-NES	SNES	BUMDA
f_1	100.0 9.47e-07±3.77e-08 2.74e+04±3.11e+02	100.0 9.46e-07±5.75e-08 **2.06e+04±2.29e+02**	100.0 **9.10e-07±8.18e-08** 5.17e+04±3.91e+02
f_2	100.0 9.34e-07±6.69e-08 2.87e+04±4.32e+02	100.0 9.17e-07±6.17e-08 **2.17e+04±2.49e+02**	100.0 **9.07e-07±7.41e-08** 9.19e+04±5.52e+02
f_3	100.0 9.16e-07±7.65e-08 1.61e+04±4.07e+02	100.0 9.10e-07±8.24e-08 **1.26e+04±3.80e+02**	100.0 **8.06e-07±1.31e-07** 2.42e+05±1.84e+03
f_4	100.0 9.44e-07±4.27e-08 3.38e+04±4.43e+02	100.0 9.40e-07±3.67e-08 **2.55e+04±2.17e+02**	100.0 **9.21e-07±6.28e-08** 1.89e+05±6.98e+02
f_5	100.0 9.39e-07±5.30e-08 2.24e+04±3.82e+02	100.0 9.41e-07±5.83e-08 **1.69e+04±2.24e+02**	100.0 **9.00e-07±7.27e-08** 5.86e+04±3.87e+02
f_6	100.0 9.39e-07±4.23e-08 3.03e+04±2.98e+02	100.0 **9.16e-07±6.02e-08** **2.29e+04±1.80e+02**	100.0 9.23e-07±5.35e-08 1.72e+05±7.75e+02
f_7	100.0 9.29e-07±5.29e-08 2.97e+04±5.16e+02	100.0 9.31e-07±7.23e-08 **2.24e+04±4.11e+02**	100.0 **9.24e-07±5.92e-08** 1.33e+05±6.78e+02
f_8	0.0 2.31e+03±8.35e+02 **8.00e+05±0.00e+00**	0.0 **8.78e+01±7.07e+01** 8.00e+05±0.00e+00	0.0 1.58e+02±3.40e+01 8.00e+05±0.00e+00
f_9	0.0 3.25e+04±1.26e+04 **8.00e+05±0.00e+00**	0.0 2.71e+04±1.22e+04 8.00e+05±0.00e+00	0.0 **1.19e+03±2.36e+02** 8.00e+05±0.00e+00
f_{10}	100.0 9.70e-07±2.86e-08 3.30e+04±4.73e+02	100.0 9.61e-07±3.02e-08 **2.49e+04±2.45e+02**	100.0 **9.56e-07±3.33e-08** 1.09e+05±4.96e+02
f_{11}	0.0 7.64e+01±2.02e+01 **8.00e+05±0.00e+00**	0.0 **6.06e+01±2.86e+01** 8.00e+05±0.00e+00	0.0 7.75e+01±1.36e-01 8.00e+05±0.00e+00
f_{12}	96.7 4.11e-04±2.25e-03 4.98e+04±1.42e+05	96.7 2.47e-04±1.35e-03 **4.40e+04±1.43e+05**	**100.0** **9.15e-07±6.02e-08** 7.47e+04±3.41e+02
f_{13}	0.0 1.71e+02±3.19e+01 **8.00e+05±0.00e+00**	0.0 1.05e+02±1.67e+01 8.00e+05±0.00e+00	0.0 **5.67e+00±2.85e+00** 8.00e+05±0.00e+00
f_{14}	0.0 5.75e+00±2.21e+00 8.00e+05±0.00e+00	3.3 2.21e+00±1.08e+00 7.74e+05±1.42e+05	**100.0** **9.19e-07±6.16e-08** **7.51e+04±4.43e+02**
f_{15}	0.0 1.02e+00±3.75e-01 8.00e+05±0.00e+00	0.0 4.48e-01±2.25e-01 8.00e+05±0.00e+00	**100.0** **9.18e-07±6.79e-08** **6.07e+04±5.00e+02**
f_{16}	100.0 9.22e-07±5.07e-08 2.12e+04±4.66e+02	100.0 9.40e-07±6.46e-08 **1.57e+04±2.82e+02**	100.0 **8.90e-07±7.67e-08** 5.97e+04±4.96e+02
f_{17}	83.3 9.07e-03±2.21e-02 1.49e+05±2.96e+05	96.7 1.30e-03±7.10e-03 **3.97e+04±1.44e+05**	**100.0** **9.18e-07±6.02e-08** 5.28e+04±3.86e+02
f_{18}	86.7 1.80e-03±4.94e-03 1.24e+05±2.70e+05	93.3 7.33e-04±2.79e-03 6.73e+04±1.99e+05	**100.0** **9.17e-07±6.79e-08** **6.11e+04±4.75e+02**

(b) dimension 80

Table 6.5 Comparing BUMDA-NES, SNES and BUMDA using the set of benchmark functions. For each function three characteristics are reported: 1) the success rate over the 30 runs (first row), 2) the average best value reached (second row) and 3) the average number of fitness evaluations (third row). The best value of each function is boldfaced according to each characteristic.

For a nicer visualization, in Figures 6.8, 6.9, 6.10 and 6.11 a graphical comparison between the three algorithms is reported using log-log plots of the problem dimensionality versus the average number of fitness evaluations and the best value reached in each problem.

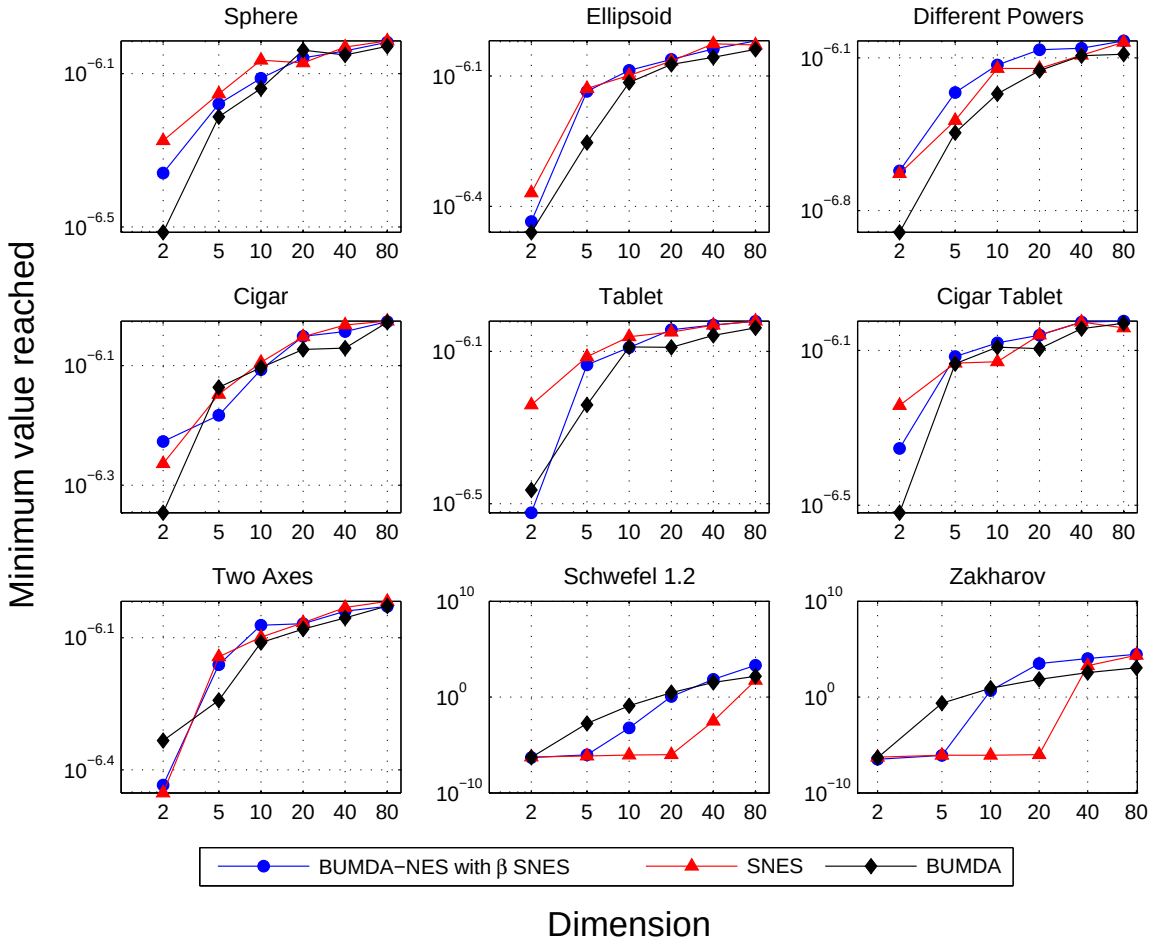


Fig. 6.8 Log-log plot of the average of the minimum value reached for BUMDA-NES, BUMDA and SNES over the 30 runs using the set of unimodal benchmark functions on dimensions 2 to 80.

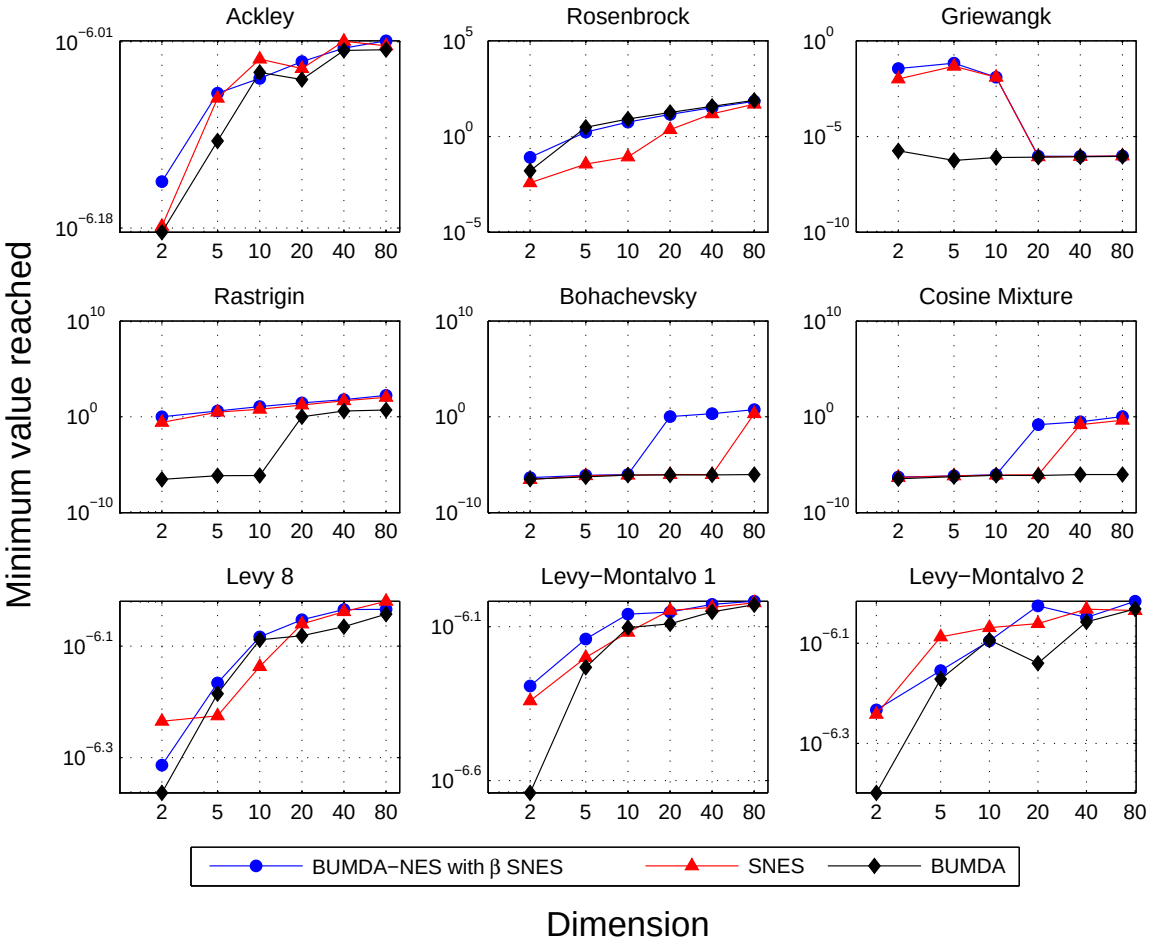


Fig. 6.9 Log-log plot of the average of the minimum value reached for BUMDA-NES, BUMDA and SNES over the 30 runs using the set of multimodal benchmark functions on dimensions 2 to 80.

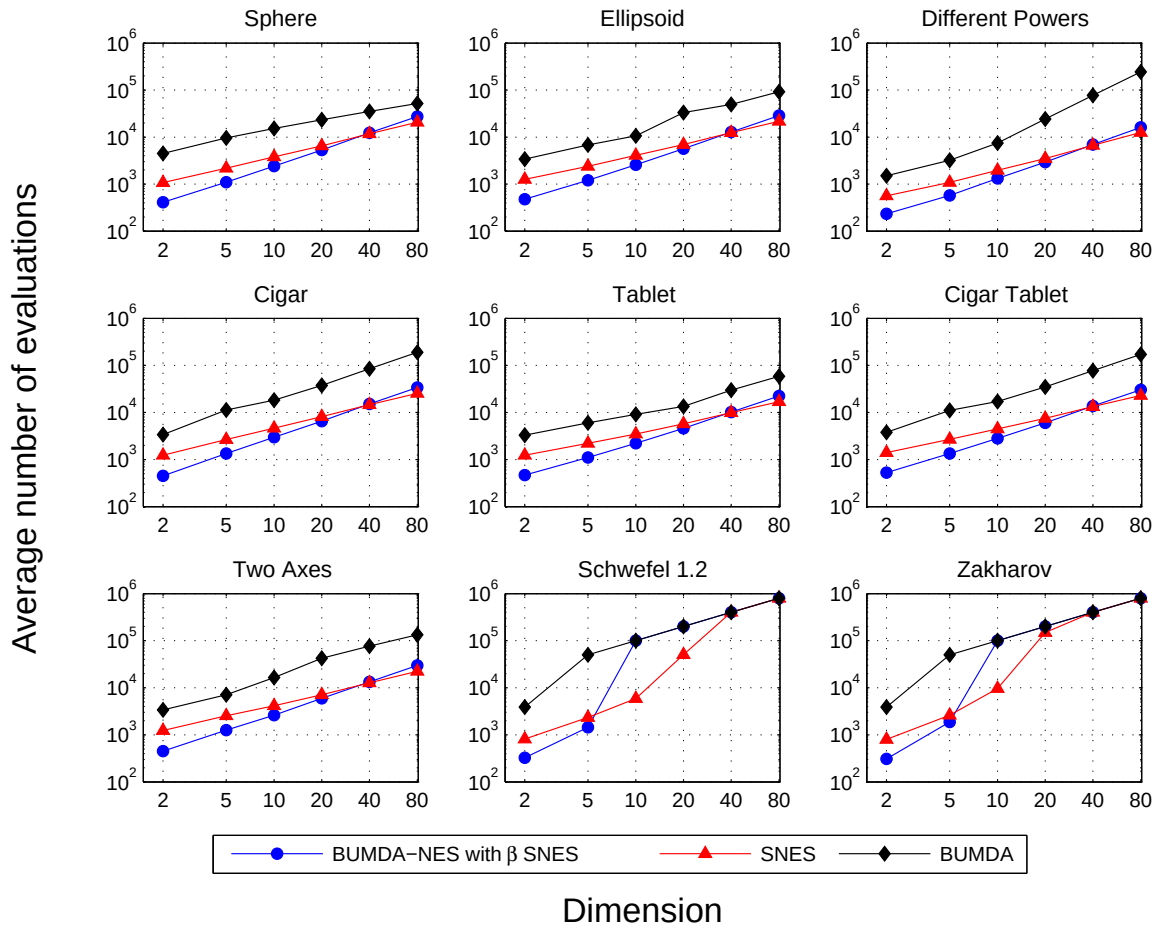


Fig. 6.10 Log-log plot of the average number of fitness evaluations for BUMDA-NES, BUMDA and SNES required to reach the target fitness value of -10^{-6} over the 30 runs using the set of unimodal benchmark functions on dimensions 2 to 80.

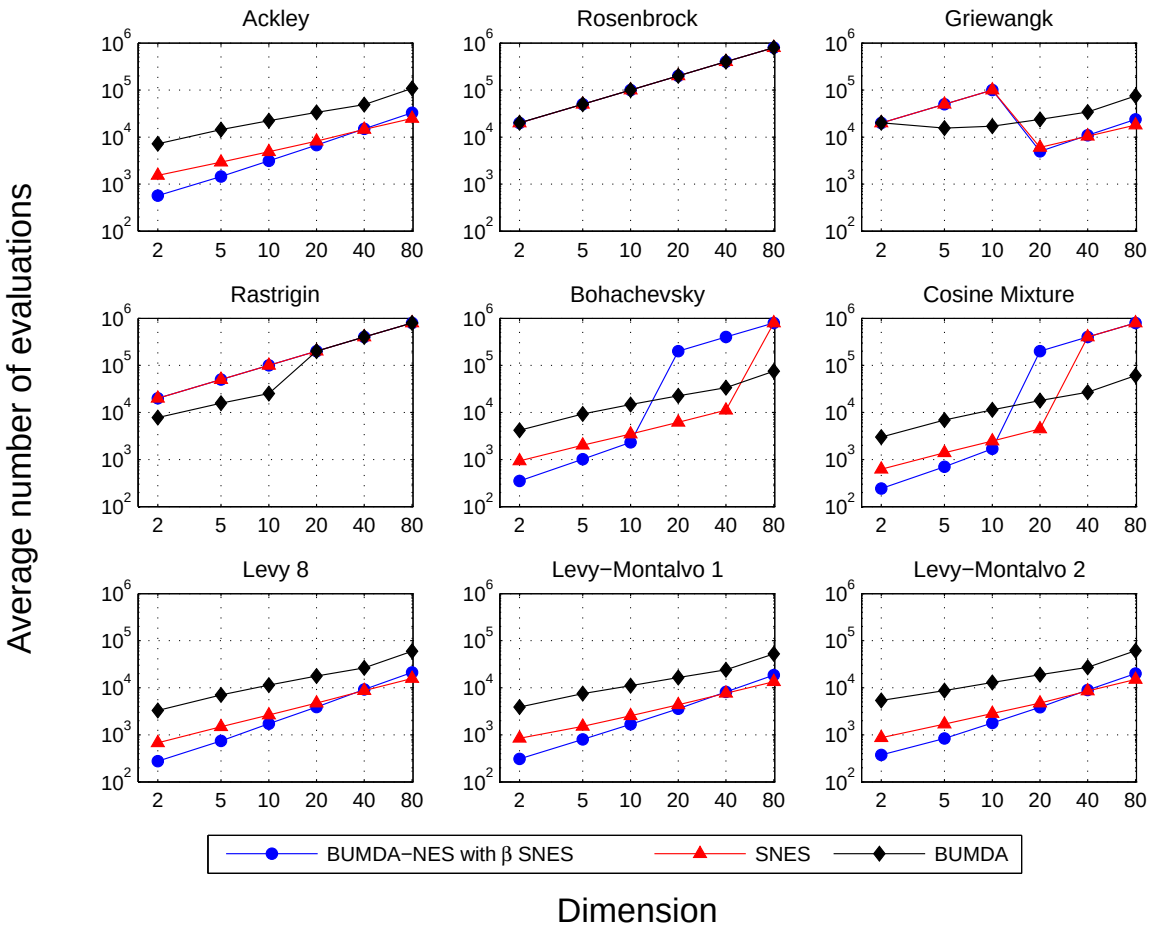


Fig. 6.11 Log-log plot of the average number of fitness evaluations for BUMDA-NES, BUMDA and SNES required to reach the target fitness value of -10^{-6} over the 30 runs using the set of multimodal benchmark functions on dimensions 2 to 80.

6.4 Summary

According to the experiments, BUMDA-NES performs well in unimodal problems, as expected. Furthermore, BUMDA-NES shows better performance in lower dimensions than the other comparing approaches for unimodal functions, however when the dimensionality of the problem increases (up to 40 or 80) its performance decreases.

On the other hand, when the functions are multimodal, BUMDA-NES presents difficulties to find a good solution, as expected. Despite BUMDA-NES shows competitive performance, lacks of consistent results over the 30 runs, that is why presents success rates lower than 100 in many multimodal cases. The Rosenbrock function is not solved for any of the three approaches.

Chapter 7

Conclusions and Future Work

This thesis study the coupling and the unification of two novel concepts presented (in a separate way) in some existing stochastic optimization algorithms. These two concepts are the Boltzmann distribution and the natural gradient direction. Both have already been used in Evolutionary Computation to propose new algorithms based on strong theoretical foundations unlike the traditional algorithms based on mimicking the nature.

First, related to the (parameterized) model in a MBEA, the Boltzmann distribution is commonly introduced in EDAs as a good choice for the search distribution, however it has several properties that complicate using it directly in practice. Second, refering to the model update phase in a MBEA, the natural gradient direction was introduced in NES as the true gradient direction to follow in the parameter space of a certain parameterized search distribution in order to maximize a given cost function.

Additionally, NES makes explicit the idea of improving the solution on an alternative optimization problem to find the search distribution parameters which iteratively locates the distribution closer to global optima regions.

The unification is performed by taking a univariate normal distribution as the search distribution. Then we define a different cost function $J(\theta)$ than the presented in NES. This alternative cost function represents the "distance" between the Boltzmann and the univariate normal distribution measured with the Kullback-Leibler divergence. In this way, the normal distribution parameters $\theta = (\mu, \sigma)$ are updated by maximizing $J(\theta)$ through estimating the natural gradient direction using the current sample. Finally, the resulting EDA is called BUMDA-NES because unifies the main principles of BUMDA and NES algorithms respectively.

In summary, the advantages presented in BUMDA-NES are:

- The natural gradient direction that minimizes the KLD between the Boltzmann and Gaussian densities has a closed (and simple) expression.
- The natural gradient incorporates invariance in the parameterization chosen to represent the Gaussian distribution.
- The Univariate Gaussian distribution makes a more scalable algorithm for dealing with higher dimensions than multivariate algorithms in the sense that just $2d$ parameters have to be adapted (where d refers to the problem dimensionality).
- The Boltzmann distribution incorporates a new free parameter (the Boltzmann temperature β) in the update equations for μ and σ, which can add robustness to the update phase if β is controlled in a proper way.
- It is easy to implement.

Furthermore, according to the experimental results obtained, BUMDA-NES shows competitive performance with respect to its predecesors, even when its tunnable parameters n, η and u have not been studied yet.

As a conclusion it is important to notice that establishing a link between existing ideas in other approaches is relevant to develop new principled EDAs of this type. Furthermore, in order to move forward in this line of research, some open questions have to be addressed, such as: how to choose a 1) cost function $J(\theta)$ and a 2) search distribution $Q(x;\theta)$ in order to improve the guided search of a NES algorithm?

Finally, in order to continue with the research developed in this thesis, future work will contemplate:

- To study the behavior of the remaining parameters in BUMDA-NES, which are: population size n, learning rates η and the weighting function $u(f(x))$ in order to adjust them in a proper way (e.g. deriving general rules to get auto-adaptable parameters).
- To analyze the derivation of a more general annealing schedude for the β parameter by using some properties of the one obtained in this work (e.g. using a different value for each dimension, be depending on the current population variance, be fluctuating over the optimization process).
- To derive a different β annealing schedule following the same strategy with other similar algorithms to BUMDA-NES (e.g BUMDA or Sep-CMA-ES).
- To compare BUMDA-NES with multivariate EDAs in order to measure the benefit of using a univariate model to perform the search.

References

[1] Akimoto, Y., Nagata, Y., Ono, I., and Kobayashi, S. (2010). Bidirectional relation between CMA evolution strategies and natural evolution strategies. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6238 LNCS(PART 1):154–163.

[2] Akimoto, Y., Nagata, Y., Ono, I., and Kobayashi, S. (2012). Theoretical foundation for cma-es from information geometry perspective. *Algorithmica*, 64(4):698–716.

[3] Amari, S. and Douglas, S. (1998). Why natural gradient? *... , 1998. Proceedings of the 1998 IEEE ...*, 9:1213–1216.

[4] Amari, S.-i. (1985). *Differential-Geometrical Methods in Statistics*, volume 28 of *Lecture Notes in Statistics*. Springer New York, New York, NY.

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

[6] Arnold, L., Auger, A., Hansen, N., and Ollivier, Y. (2011). Information-Geometric Optimization Algorithms: A Unifying Picture via Invariance Principles.

[7] Bäck, T., Foussette, C., and Krause, P. (2013). *Contemporary Evolution Strategies*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg.

[8] Beyer, H.-G., Beyer, H.-G., Schwefel, H.-P., and Schwefel, H.-P. (2002). Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3 – 52.

[9] Eiben, a. E. and Rudolph, G. (1999). Theory of evolutionary algorithms : a bird's eye view. *Theoretical Computer Science*, 229:3–9.

[10] Gallagher, M. and Frean, M. (2005). Population-based continuous optimization, probabilistic modelling and mean shift. *Evolutionary computation*, 13(1):29–42.

[11] Gallier, J. (2011). *Geometric Methods and Applications*, volume 38 of *Texts in Applied Mathematics*. Springer New York, New York, NY.

[12] Glasmachers, T., Schaul, T., Yi, S., Wierstra, D., and Schmidhuber, J. (2010). Exponential natural evolution strategies. *Matrix*, pages 393–400.

[13] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):pp. 671–680.

[14] Lozano, J. a., Lozano, J. a., Larrañaga, P., Larrañaga, P., Inza, I., Inza, I., Bengoetxea, E., and Bengoetxea, E. (2002). *Estimation of Distribution Algorithms*, volume 2 of *Genetic Algorithms and Evolutionary Computation*. Springer US, Boston, MA.

[15] Mahnig, T. and Muhlenbein, H. (2001). A new adaptive Boltzmann selection schedule SDS. *Evolutionary Computation, 2001. ...*.

[16] Mühlenbein, H. and Mahnig, T. (2000). Theoretical Aspects of Evolutionary Computing. pages 137–176.

[17] Mühlenbein, H. and Mahnig, T. (2001). Comparing the adaptive Boltzmann selection schedule SDS to truncation selection. In *Proceedings of the Third International Symposium on Adaptive Systems*, number 3, pages 121–128.

[18] Mühlenbein, H., Mahnig, T., and Rodriguez, A. O. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247.

[19] Peters, J. (2007). Machine Learning of Motor Skills for Robotics. *Baseline*, 22(May):41–43.

[20] S. Ivvan Valdez, Arturo Hernández, S. B. (2008). A boltzmann based estimation of distribution algorithm. Technical Report I, CIMAT.

[21] Schaul, T., Glasmachers, T., and Schmidhuber, J. (2011). High dimensions and heavy tails for natural evolution strategies. *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, page 845.

[22] Segovia-Dominguez, I. and Hernandez-Aguirre, A. (2015). An Estimation of Distribution Algorithm based on the Natural Gradient and the Boltzmann Distribution. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*, pages 527–534, New York, New York, USA. ACM Press.

[23] Segovia-Dominguez, I., Hernandez-Aguirre, A., and Valdez, S. I. (2015). Designing the Boltzmann estimation of multivariate normal distribution: Issues, goals and solutions. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, number x, pages 2082–2089. IEEE.

[24] Segovia-Domínguez, I., Valdez, S. I., and Hernández-Aguirre, A. (2014). A Boltzmann Multivariate Estimation of Distribution Algorithm for Continuous Optimization. In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications*, pages 251–258.

[25] Srivastava, A. (2012). CVPR Tutorial on Differential Geometry in ComputerVision and Pattern Recognition, June 21.

[26] Sun, Y., Wierstra, D., Schaul, T., and Schmidhuber, J. (2009). Efficient Natural Evolution Strategies. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 539–546, New York, NY, USA. ACM.

[27] Thierens, D. and a.N. Bosman, P. (2014). (Slides) Model-based evolutionary algorithms. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14*, pages 431–458.

[28] Valdez, S. I., Hernández, A., and Botello, S. (2013). A Boltzmann based estimation of distribution algorithm. *Information Sciences*, 236:126–137.

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

[30] Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., and Schmidhuber, J. (2011). Natural Evolution Strategies. *arXiv*, pages 1–52.

[31] Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. (2008). Natural Evolution Strategies. *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387.

[32] Yunpeng, C., Xiaomin, S., and Peifa, J. (2006). Probabilistic modeling for continuous EDA with Boltzmann selection and Kullback-Leibeler divergence. *... of the 8th annual conference on ...*, pages 389–396.

[33] Zlochin, M., Birattari, M., Meuleau, N., and Dorigo, M. (2004). Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1-4):373–395.