Advancing Continuous IDEAs with Mixture Distributions and Factorization Selection Metrics

Peter A.N. Bosman
Peter.Bosman@cs.uu.nl

Dirk Thierens
Dirk.Thierens@cs.uu.nl

Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

Evolutionary optimization based on probabilistic models has so far been limited to the use of factorizations in the case of continuous representations. Furthermore, a maximum complexity parameter κ was required previously to construct factorizations to prevent unnecessary complexity to be introduced in the factorization. In this paper, we advance these techniques by using clustering and the EM algorithm to allow for mixture distributions. Furthermore, we apply a search metric to eliminate the κ parameter. We use these techniques in the IDEA framework to obtain new continuous evolutionary optimization algorithms and investigate their performance.

1 Introduction

Most evolutionary algorithms (EAs) based on probabilistic model building in the continuous case make use of the normal probability density function (pdf) [5, 11, 16]. The normal pdf however is only suited to properly represent *linear* dependencies in the sample set. To overcome this problem, the normal kernels pdf has been investigated by Bosman and Thierens [6] which places a normal pdf over every sample point. Unfortunately, this pdf has been found to be quite hard to handle. The normal mixture pdf represents a trade-off between the normal pdf and the normal kernels pdf. This pdf has been used successfully on two relatively simple test functions by Gallagher, Fream and Downs [9] using an adaptive estimation technique. In this paper, we use two different approaches to constructing a normal mixture. One is by means of clustering whereas the other is by means of the expectation maximization (EM) algorithm. In a pilot study by Pelikan and Goldberg [13], clustering has previously been successfully applied to binary representations.

In previous optimization algorithms that use continuous probabilistic models, only little attention has been paid to search metrics that guide the search for a good probabilistic model. In this paper, we propose to use a metric that effectively prefers simpler models. This metric has previously been successfully applied to the case of binary variables [10, 12, 15].

Our goal in this paper is to use the proposed techniques to construct new continuous evolutionary optimization algorithms and to get an indication of their performance using a small set of test functions.

The remainder of this paper is organized as follows. In section 2, we define the notion of probabilistic models and point out how these can be used in an EA. In sections 3 and 4, we elaborate on the selection of factorizations and factorization mixtures as the structure of the probabilistic model. Subsequently, we take a look at what pdfs we can use in section 5 and thereby regard the normal mixture distribution from a different perspective. Our experiments are presented in section 6. Topics for further research are discussed in section 7 and our final conclusions are drawn in section 8.

2 Probabilistic models and IDEAs

The IDEA is a framework for *Iterated Density Estimation Evolutionary Algorithms* that use probabilistic models in evolutionary optimization and has mostly been used to focus on continuous representations and techniques [5, 6, 7, 8]. We take the elementary building block of probabilistic models to be the *probability density function* (pdf). We define a probabilistic model $\mathcal{M}$ to consist of some structure ς that describes a composition of pdfs, and a vector of parameters $\boldsymbol{\theta}$ for the pdfs implied by ς. We write $\mathcal{M} = (\varsigma, \boldsymbol{\theta})$. We write the probability distribution that is described by $\mathcal{M}$ as $P_{\mathcal{M}}$. The pdf to fit over every factor implied by ς is chosen on beforehand. The way in which the

parameters $\boldsymbol{\theta}$ are fit, is also predefined on beforehand. We denote the parameter vector that is obtained in this manner by $\boldsymbol{\theta} \xleftarrow{fit} \varsigma$. With these assumptions, we denote the resulting probability distribution by P_ς.

We assume that the dimensionality of our problem is l and write $\boldsymbol{\mathcal{L}} = (0, 1, \ldots, l-1)$. We furthermore assume that we have a continuous optimization problem $C(y_0, y_1, \ldots, y_{l-1}) = C(y\langle\boldsymbol{\mathcal{L}}\rangle)$. In the framework, we select $\lfloor \tau n \rfloor$ samples ($\tau \in [\frac{1}{n}, 1]$) in each iteration t and let θ_t be the worst selected sample cost. We then estimate the distribution of the selected samples and thereby find $\hat{P}_\varsigma^{\theta_t}(\boldsymbol{\mathcal{Y}}) = \hat{P}_\varsigma^{\theta_t}(Y_0, Y_1, \ldots, Y_{l-1})$ as an approximation to the true uniform distribution $P^{\theta_t}(\boldsymbol{\mathcal{Y}})$ over all points $y\langle\boldsymbol{\mathcal{L}}\rangle$ with $C(y\langle\boldsymbol{\mathcal{L}}\rangle) \leq \theta_t$ (assuming *minimization*). New samples can then be drawn from $\hat{P}_\varsigma^{\theta_t}(\boldsymbol{\mathcal{Y}})$ and be used to replace some of the current samples. If we draw $n - \lfloor \tau n \rfloor$ new samples, select by taking the best $\lfloor \tau n \rfloor$ vectors and finally replace the worst $n - \lfloor \tau n \rfloor$ samples by the new samples, we have that $\theta_{t+1} = \theta_t - \varepsilon$ with $\varepsilon \geq 0$. This assures that the search for θ^* is conveyed through a monotonically decreasing series $\theta_0 \geq \theta_1 \geq \ldots \geq \theta_{t_{\text{end}}}$. We call an IDEA so constructed *monotonic*.

3 Factorization selection

To estimate the probability distribution of the selected samples in the IDEA, we first search for a model structure. In this section, we only focus on the specific structure of a *conditional factorization*. We can state that variable Y_i is either *conditionally dependent* on variables $Y\langle\boldsymbol{a}\rangle$ with $i \notin \boldsymbol{a}$, or it is not. By identifying a vertex with each variable Y_i and an arc (Y_i, Y_j) if and only if Y_j is conditionally dependent on Y_i, we get the conditional factorization *graph*. A conditional factorization is valid if and only if its factorization graph is *acyclic*. The resulting probability distribution is the product of l conditional pdfs in which each variable is conditioned on at most $l-1$ parents.

We have to learn such a conditional factorization from the vector of selected samples. To this end, a variety of approaches can be taken [14]. We use an incremental algorithm that starts from the empty graph with no arcs. Each iteration, the arc to add is selected to be the arc that increases some metric the most. If no addition of any arc further increases the metric, the final factorization graph has been found. In earlier work on continuous models [5, 11], the entropy was used as a metric. The entropy is however equal to the average negative log–likelihood of the sample set if the pdfs were fitted to be of a maximum likelihood [7]. As a result, this metric monotonically increases as the

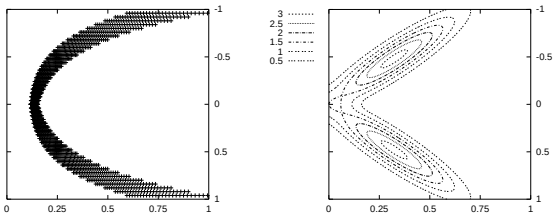

Figure 1: A non–linear dependency in the sample set (left) and the contour lines of the density estimation using two normal pdfs after clustering (right).

complexity of the factorization increases. Therefore, a maximum order of interaction κ had to be used that defines how many parents any variable may be dependent on in a conditional factorization. To remove this non–transparent parameter κ, the negative log–likelihood can be penalized as the complexity of the factorization increases. The metric that we use in this paper, is commonly known as the *Bayesian Information Criterion* (BIC). For a derivation of this metric in the IDEA context, we refer the reader to a more detailed report [8]. Let $\boldsymbol{\mathcal{S}} = (\boldsymbol{y^0}, \boldsymbol{y^1}, \ldots, \boldsymbol{y^{|\mathcal{S}|-1}})$ be the selected vector of l–dimensional samples. The BIC metric is parameterized by a regularization parameter λ that determines the amount of penalization:

$$\underbrace{-\ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})))}_{\text{Error}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}})} + \underbrace{\lambda \ln(|\boldsymbol{\mathcal{S}}|)|\boldsymbol{\theta}|}_{\text{Complexity}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}})} \quad (1)$$

In equation 1, we have used the the *likelihood* of $\mathcal{M}$:

$$\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})) = \prod_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y^i}) \quad (2)$$

4 Factorization mixture selection

The structure of the sample vector may be highly non–linear. This non–linearity can force us to use probabilistic models of a high complexity to retain some of this non–linearity. However, especially using relatively simple pdfs such as the normal pdf, the non–linear interactions cannot be captured even with higher order models. The use of clusters allows us to efficiently break up non–linear interactions so that we can use simple models to get an adequate representation of the sample vector. An example of this is depicted in figure 1. Furthermore, computationally efficient clustering algorithms exist that provide useful results.

Each cluster is processed separately in order to have a factorized probability distribution fit over it. By doing so, we obtain a *mixture distribution*. We let k be the amount of clusters and let $\boldsymbol{\mathcal{K}} = (0, 1, \ldots, k-1)$. In general, we write $\mathfrak{f}$ for a factorization. For a mixture of factorizations, we write $\mathfrak{f}\langle\boldsymbol{\mathcal{K}}\rangle$. The resulting probability distribution is a weighted sum of the individual probability distributions over each cluster:

$$\hat{P}_{\mathfrak{f}\langle\boldsymbol{\mathcal{K}}\rangle}(\boldsymbol{\mathcal{Y}}) = \sum_{i=0}^{k-1} \beta_i \hat{P}^i_{\mathfrak{f}_i}(\boldsymbol{\mathcal{Y}}) \tag{3}$$

One way to set the mixture coefficients β_i, is to proportionally assign larger coefficients to the clusters with a better average cost. A simpler way is to set β_i proportionally to the size of cluster i. In order to perform clustering, different approaches can be taken. One of the most well known algorithms is the k–means clustering algorithm that constructs exactly k clusters. First, k cluster centroids are picked at random. One way to do this, is by randomly selecting k points from the points that have to be clustered. Subsequently, the algorithm iterates until the means do not change to within a significance of ε anymore. One iteration consists of assigning each point to the nearest cluster based on the distance to the cluster centroid. Once all points have been assigned, the means of the clusters are recomputed. The distance measure that we have used, is the Euclidic distance scaled by the variance of the data points in each dimension.

The randomized leader algorithm is one of the fastest clustering algorithms. The first sample to make a new cluster is appointed to be its leader. The leader algorithm goes over the sample vector exactly once. For each sample, it finds the cluster with the closest leader. If this leader is closer than a given threshold $\mathfrak{T}_d$, the sample is added to that cluster. Otherwise, a new cluster is created containing only this single sample. To prevent the first clusters from becoming quite a lot larger than the later ones, we randomize the order in which the clusters are inspected. To avoid problems with two clusters to which some samples are equally close, the order in which the clusters are scanned is randomized as well. The largest difference with the k–means clustering algorithm is that because of the threshold $\mathfrak{T}_d$, we do not know in advance exactly how many clusters will be created. In a previous study [8], the variance scaled Euclidic distance was used. The main problem with this distance measure is that the maximum distance in an l dimensional unit hypercube is $\sqrt{l}$, so the range of $\mathfrak{T}_d$ is not constant with respect to l. To ensure this, we propose to first compute the bounding box of the data points by computing the minimum coordinate y_i^{MIN} as well as the maximum coordinate y_i^{MAX} in each dimension i and by scaling the distance in each dimension by the range in that dimension. This ensures that all points are treated as if inside the unit hypercube. To subsequently ensure that $\mathfrak{T}_d \in [0, 1]$, we could divide the distance by $\sqrt{l}$ as an approximation of the maximum distance in the data set. However, depending on the form of the data set and the dimensionality l, this can be a bad estimate. To compute the exact maximum distance, we require $\mathcal{O}(|\boldsymbol{\mathcal{S}}|^2)$ computations, which is an order of magnitude more than the actual clustering algorithm itself. Therefore, we propose to approximate the maximum distance by taking the maximum distance over all points to the minimal and maximal points in the bounding box respectively. This distance measure, which we call the *Bounding Box Euclidic Normalized Distance* ($\mathbb{B}$END), can be formalized as follows:

$$d_{\mathbb{B}\text{END}}(\boldsymbol{y^i}, \boldsymbol{y^j}) = \frac{\sqrt{\sum_{k=0}^{l-1} \frac{(y_k^i - y_k^j)^2}{(y_k^{\text{MAX}} - y_k^{\text{MIN}})^2}}}{\text{Approx. Max. Dist.}} \tag{4}$$

Since $d_{\mathbb{B}\text{END}} \in [0, 1]$, we also have $\mathfrak{T}_d \in [0, 1]$, regardless of l. This makes the selection of an initial threshold simpler. By limiting the maximum amount of clusters to some value close to the desired average amount of clusters, the $\mathbb{B}$END leader algorithm becomes a fast and flexible adaptive clustering algorithm.

5 Probability Density Functions

A widely used parametric pdf is the normal pdf. The sample average in dimension j is $\overline{Y_j} = \frac{1}{|\boldsymbol{\mathcal{S}}|} \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} (\boldsymbol{y^i})_j$. The sample covariance matrix over variables $Y\langle\boldsymbol{a}\rangle$ is $\boldsymbol{\Sigma}\langle\boldsymbol{a}\rangle = \frac{1}{|\boldsymbol{\mathcal{S}}|} \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} (y^i\langle\boldsymbol{a}\rangle - \overline{Y}\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle - \overline{Y}\langle\boldsymbol{a}\rangle)^T$. To compute the BIC measure, we compute the entropy of the normal pdf, since this is significantly faster than evaluating the negative log–likelihood. The required conditional pdf and the entropy are [5]:

$$f_{\mathcal{N}}(y_{\boldsymbol{a}_0} | y\langle\boldsymbol{a} - \boldsymbol{a}_0\rangle) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_{\boldsymbol{a}_0} - \mu)^2}{2\sigma^2}} \tag{5}$$

$$\text{where} \begin{cases} \sigma = \frac{1}{\sqrt{\boldsymbol{\Sigma}\langle\boldsymbol{a}\rangle^{-1}(0,0)}} \\ \mu = \frac{\overline{Y_{\boldsymbol{a}_0}} \boldsymbol{\Sigma}\langle\boldsymbol{a}\rangle^{-1}(0,0) - \sum_{i=1}^{|\boldsymbol{a}|-1} (y_{\boldsymbol{a}_i} - \overline{Y_{\boldsymbol{a}_i}}) \boldsymbol{\Sigma}\langle\boldsymbol{a}\rangle^{-1}(i,0)}{\boldsymbol{\Sigma}\langle\boldsymbol{a}\rangle^{-1}(0,0)} \end{cases}$$

$$h(Y\langle\boldsymbol{a}\rangle) = \frac{1}{2}(|\boldsymbol{a}| + \ln((2\pi)^{|\boldsymbol{a}|} \det(\boldsymbol{\Sigma}\langle\boldsymbol{a}\rangle))) \tag{6}$$

So far, all continuous factorization selection approaches in probabilistic model building EAs have used

the normal pdf. The only exception is the use of the normal kernels pdf [6]. However, this pdf has found to be difficult to handle. By clustering the samples and fitting a normal pdf in each cluster, we obtain a normal mixture pdf. However, this is only an approximation of the maximum likelihood fit since the clustering algorithm only focuses on neatly partitioning the data set. If we would see the normal mixture pdf as an elementary pdf, we look at fitting this pdf from a maximum likelihood perspective. This leads to the *expectation maximization* (EM) algorithm [3] which is a general approach to finding a maximum likelihood fit. Even though the algorithm is theoretically rigorous, it can easily get stuck in local minima, resulting in a sub optimal fit. This is especially true in higher dimensions. By using this approach, we can no longer speak of clusters that comprise the mixture model, but only of a normal mixture pdf as a whole. Although formulas exist to infer factorizations based on normal mixture pdfs [8], such inference is far too time consuming to be of practical use. Furthermore, since the EM algorithm is itself computationally intensive, its only practical use has been observed [8] to be when the univariate factorization is used. In this case, each normal pdf in the mixture has a diagonal covariance matrix.

6 Experiments

The continuous function optimization problems we used for testing are the following:

C_0	$\frac{1}{4000}\sum_{i=0}^{l-1}(y_i-100)^2 - \prod_{i=0}^{l-1}\cos\left(\frac{y_i-100}{\sqrt{i+1}}\right)+1$	$[-600,600]^l$
C_1	$-\sum_{i=0}^{l-1}\sin(y_i)\sin^{20}\left(\frac{(i+1)y_i^2}{\pi}\right)$	$[0,\pi]^l$
C_2	$\sum_{i=0}^{l-2}100(y_{i+1}-y_i^2)^2+(1-y_i)^2$	$[-5.12,5.12]^l$

All of the test functions should be *minimized*. Function C_0 is Griewank's function, C_1 is Michalewicz's function and C_2 is Rosenbrock's function. In all our testing, we used a *monotonic* IDEA. We used the rule of thumb by Mühlenbein and Mahnig [12] for FDA and set τ to 0.3. We ran tests so as to find the best optimization results for $n \leq 5000$. We allowed each run a maximum of $1 \cdot 10^7$ evaluations. If all of the solutions differed by less than $5 \cdot 10^{-7}$, termination was enforced also. Note that this implies a maximum precision of 6 decimal digits. All results were averaged over 10 runs. We tested both the normal pdf as well as the normal mixture pdf. For the normal pdf, we searched for conditional factorizations using the BIC metric with $\lambda = \frac{1}{2}$. For the normal mixture pdf fitted by the EM algorithm, we have fixed the factorization structure to the univariate one. Furthermore, we have used 10 normal pdfs in each mixture. We applied clustering using both the BEND leader algorithm as well as the k–means clustering algorithm. In the case of the BEND leader algorithm, we set $\mathfrak{T}_d$ to get approximately 10 clusters. For the k–means algorithm we used exactly 10 clusters. The mixture coefficients β_i were set to the proportional cluster sizes.

C_0				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
Normal	150	0.000000	1403463	3.19
EM	1100	0.000000	51832	6.61
Leader	750	0.006705	7268071	19.47
k–Means	2500	0.016360	2054672	23.03

Figure 2: Results on Griewank's function.

C_1				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
Leader	4000	-4.687658	180813	0.42
EM	1800	-4.687658	39756	1.61
Normal	1700	-4.654513	4088736	8.26
k–Means	4250	-4.637136	551834	2.35

Figure 3: Results on Michalewicz's function.

We present the average best results over the 10 independent runs and sort the IDEA instances by this index primarily. For all problems, the best performing algorithm reached the optimum. We also show the minimum population size $n_{\min}$ for the best obtained result, the average amount of evaluations and the *Relative Run Time* RT. Let $\text{FT}(x)$ be the time to perform x random function evaluations and let TRT be the *Total Run Time* on the same processing system. Then, $\text{RT}(x) = \text{TRT}/\text{FT}(x)$. We determined RT as $\text{RT}(10^6)$. The RT index is a processing system independent fair comparison metric. We sort the results by this index secondarily instead of the amount of evaluations, as it *truly* reflects the required amount of time. The results are shown in figures 2, 3 and 4.

C_2				
pdf	$n_{\min}$	$\overline{C}$	$\overline{\text{evals}}$	RT
k–Means	3000	0.000000	82575	3.65
Leader	5000	0.050435	646558	7.36
EM	4000	0.065469	341870	72.56
Normal	2750	1.859228	2370574	19.30

Figure 4: Results on Rosenbrock's function.

The first thing to note is that by using a single normal pdf, only the C_0 function can be optimized with $n \leq 5000$. Moreover, it has been empirically observed that even for larger values of n, both C_1 and C_2 are not optimized. This means that solving these problems is not a matter of finetuning the parameters of the IDEA using normal pdfs, but that it requires more involved techniques. From figure 3 we see that by using a normal mixture model, the desired results *can* be obtained. Using the EM algorithm even requires less than $40 \cdot 10^3$ function evaluations. However, since the computational requirements are quite high, the actual running time is much larger than that of the efficient leader clustering approach. The same can be observed for C_0. Note that the clustering approaches are not very successful in optimizing C_0, whereas they are the most successful on C_2. Function C_2 is a very challenging one because of its non–linearity. It has a curved valley along which the quality of the solutions is much better than in its neighborhood. Furthermore, this valley has a unique minimum of 0 itself. Finally, the gradient along the bottom of the valley is only very slight. Any gradient approach is therefore doomed to follow the long road along the bottom of the valley. For a density estimation algorithm, capturing the valley in a probabilistic model is difficult, even if all of the points within the valley are known. The reason for this is that the valley is non–linear in the coding space. Therefore, it is to be expected that in order to get any reasonable results, we require clustering.

The obtained results indicate that the single normal pdf is no longer adequate when the epistasis or the non–linearity of the landscapes increases. Epistasis as encountered in C_0 and even more in C_1 can be efficiently tackled using the EM algorithm. Clustering on the other hand seems to only partly be able to resolve this problem. Non–linearity as encountered in C_2 can be efficiently tackled using clustering. The EM algorithm seems less well suited to this end. We note at this point that it has been empirically verified that for a lower threshold, the leader algorithm is able to optimize C_2 within 161017 evaluations on average. We can therefore state that mixture distributions are useful in continuous optimization by IDEAs especially for non–linear and highly epistatic problems.

Note that the amount of dimensions is only very small. If the amount of dimensions goes up, the amount of clusters will have to increase accordingly. However, by doing so, the amount of samples will have to increase as well to ensure a large enough cluster size. Therefore, the sole use of density estimation on smooth but strongly non–linear search spaces such as C_2 is not effective enough to compete with approaches that use gradient information. However, gradient based algorithms are known to have difficulties with epistatic search spaces such as C_0 and C_1. Therefore, a hybrid combination of both approaches will most likely result in very effective continuous optimization techniques.

7 Discussion

Next to being a useful tool for improving density estimation in IDEAs, clustering is also useful for other purposes. One of these is multimodal optimization where we want to find multiple global optima or even some additional good local optima [13]. By setting the mixture coefficients β_i to the relative average solution quality, the solutions can be distributed among the different peaks. Clustering also has an application in multi–objective optimization where an optimization problem consists of multiple objectives that are equally important [17]. By clustering in the objective space, a good distribution of the points along the optimal front can be found. An important question is whether and how the added effort of clustering is affected by the increase of the dimensionality. A study in which the scaling behavior is investigated, would therefore be most interesting.

Approaches such as the IDEA that build and use probabilistic models in evolutionary optimization were first proposed as an improvement over binary GAs [2, 4]. The IDEA framework itself has been used to focus on continuous models for problems with real valued variables. This has resulted in a new line of continuous evolutionary algorithms. However, for continuous optimization, evolutionary algorithms have been proposed almost as long ago as the original simple binary GA [1]. Currently, we can identify two variants of this continuous evolutionary algorithm, namely Evolutionary Programming (EP) and Evolution Strategies (ES). Both make use of normal pdfs. For this reason, it is important to question the relevance of a new approach such as the IDEA instances presented so far. In ES, mutation is the most important operator. It is based on normal pdfs. Each solution has a normal pdf associated with it. The parameters of this normal pdf are subject to mutation themselves. The main idea is that a solution is mutated by adding a value drawn from the normal pdf that is associated with the solution. The IDEA approach is a global procedure that attempts to use the structure of the problem landscape to explore the most promising regions. On the other hand, mutation based ES approaches are local procedures that use evolution to explore the inside of promising regions. The two approaches are therefore fundamentally different. For example on Rosenbrock's function,

an IDEA based approach will quite easily find the valley itself, but will by no means be able to traverse the valley to find the global minimum unless points were sampled near the global minimum. The reason for this is that density estimation converges on a certain part of the valley since samples are only available in that part of the search space. On the other hand, once the ES is inside the valley, it can adapt its mutation direction and stepsize to follow the valley to its minimum in a gradient descent fashion. Even though this is a time consuming process, the ES is not as likely to prematurely converge on such a problem as is the IDEA approach. Hybridizing the IDEA to exploit gradient information for pure continuous optimization therefore appears to be a good idea.

8 Conclusions

In this paper we have extended the set of available tools for IDEAs to perform continuous optimization with, by advancing to mixture modeling. Furthermore, we have done away with the maximum interaction parameter κ for conditional factorizations by using the BIC metric. These extensions provide new and more complex tools that allow for a better performance of IDEAs on epistatic and non–linear continuous optimization problems.

References

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

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

[3] J. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. ICSI Technical Report TR-97-021. http://www.icsi.berkeley.edu/~bilmes/papers/em.ps.gz, 1997

[4] J.S. De Bonet, C. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing*, 9, 1996

[5] P.A.N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 767–776. Springer, 2000

[6] P.A.N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In M. Pelikan, H. Mühlenbein, and A.O. Rodriquez, editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference GECCO–2000*. Morgan Kaufmann Publishers, 2000

[7] P.A.N. Bosman and D. Thierens. Negative log–likelihood and statistical hypothesis testing as the basis of model selection in IDEAs. In A. Feelders, editor, *Proceedings of the Tenth Dutch–Netherlands Conference on Machine Learning*. Tilburg University, 2000

[8] P.A.N. Bosman and D. Thierens. Mixed IDEAs. Utr. Univ. Tech. R. UU–CS–2000–45. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2000/2000-45.ps.gz, 2000

[9] M. Gallagher, M. Fream, and T. Downs. Real–valued evolutionary optimization using a flexible probability density estimator. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 840–846. Morgan Kaufmann Publishers, 1999

[10] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Tech. R. 99010. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99010.ps.Z, 1999

[11] P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña. Optimization by learning and simulation of bayesian and gaussian networks. Univ. of the Basque Country Tech. R. EHU-KZAA-IK-4/99. http://www.sc.ehu.es/ccwbayes/postscript/kzaa-ik-04-99.ps, 1999

[12] H. Mühlenbein and T. Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evol. Comp.*, 7:353–376, 1999

[13] M. Pelikan and D.E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 385–394. Springer, 2000

[14] M. Pelikan, D.E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. IlliGAL Tech. Rep. 99018. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99018.ps.Z, 1999

[15] M. Pelikan, D.E. Goldberg, and K. Sastry. Bayesian optimization algorithm, decision graphs and occam's razor. IlliGAL Tech. R. 2000020. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/2000020.ps.Z, 2000

[16] M. Sebag and A. Ducoulombier. Extending population–based incremental learning to continuous search spaces. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 418–427. Springer, 1998

[17] D. Thierens and P.A.N. Bosman. Multi-objective optimization with iterated density estimation evolutionary algorithms using mixture models. In A. Ochoa, H. Mühlenbein, T. English, and P. Larrañaga, editors, *Proc. of the International Symposium on Adaptive Systems 2001 – Evolutionary Computation and Probabilistic Graphical Models (TO APPEAR)*, 2001