Continuous Optimization Based-on Greedy Estimation of GMM

Bin Li, Run-tian Zhong, Xian-ji Wang, Zhen-quan Zhuang
Nature Inspired Computation Laboratory, Department of Electronic Science and Technology,
University of Science and Technology of China, Hefei, China, 230026
E-mail: binli@ustc.edu.cn; zrtzky@ustc.edu; xjw@mail.ustc.edu.cn; zqzhuang@ustc.edu.cn

Abstract--**A new Estimation of Distribution Algorithms (EDAs) for continuous optimization based on greedy estimation of Gaussian Mixture Mode is proposed. By incrementally adding new components one by one, the new estimation method can approximate almost any complex probability density function efficiently, so it has the ability to learn the model structure and parameters automatically without any requirement for prior knowledge. Since for each estimation iteration the task is simplified to be a two-component mixture model learning problem, and a greedy strategy is adopted to guarantee the monotonous increasing of likelihood, the new EDA is very fast and efficient. A set of experiments has been implemented to evaluate, and to compare with other EDAs, the efficiency and performance of the new algorithm. The results show that, with a relative small number of generations, the new algorithm can perform very well on both uni-modal and multi-modal function optimization problems.**

Key words: Estimation of distribution algorithm, Continuous optimization, Gaussian mixture model, Greedy EM

I. Introduction

Estimation of Distribution Algorithms (EDAs) are population-based search algorithms that use a probabilistic model of promising solutions to guide further exploration of the search space[1]. The crucial difference between the EDAs and the other evolutionary algorithms (e.g., GA) is that EDAs replace genetic crossover and mutation operators by the following two steps[2]: (1) The true probability density of the selected promising solutions is estimated. (2) New solutions are generated from the estimated distribution. The EDAs have been proved to be very efficient in combinatorial optimization[3]-[7] and continuous optimization[8]-[14].

Most EDAs for continuous optimization use the Gaussian probability density function(pdf) to model the promising area of solution space. [8][9] and [10] model each variable with one Gaussian pdf and assume that there is no interaction among variables. These univariate models are not suitable for complex problems with inter-dependences among variables. While Gaussian Mixture Model(GMM) can model both distribution of variables and the interdependences among them, thus is adopted by later continuous EDAs[15][16][17][18]. Two different approaches are usually adopted to construct a Gaussian Mixture Model: One is by means of clustering; the other is by means of the expectation maximization (EM) algorithm. The clustering algorithms adopted by previous EDAs usually require prior knowledge, either a predefined cluster number or an minimal distance between different clusters, such as that in[15][17]. In [16], authors used another clustering method, Rival Penalized Competitive Learning (RPCL) [19], which can detect the real number of clusters without any prior knowledge. This clustering method usually assigns a larger number of clusters than the actual number in the data set, at the end of clustering, some clusters are discarded according some criterion. In fact, all the EDAs incorporating clustering techniques divide the probability density function estimation step into two steps: cluster the selected individuals and estimate the distributions. So these EDAs usually need larger population and the convergence speed is slow. For EDAs, the EM algorithm has some limitations: It assumes that the number of components is known, which is usually not the case in the estimation task of EDAs; and it is sensitive to the initial parameters, which makes it easily get trapped in local maxima of the likelihood function.

In this paper, a new continuous optimization algorithm based on Gaussian mixture model is proposed. Different from previous algorithms, the new algorithm adopts a greedy EM algorithm[20] to estimate GMM. The most important advantage is that it can implement the model structure learning and model parameter learning during the same procedure automatically. Our motivation is to construct an EDA that can implement the estimation of the probability density function in a more simple and efficient way, so that the convergence speed could be faster. By using greedy EM, the proposed EDA needn't do clustering and requires no prior knowledge. By incrementally adding Gaussian components one by one, the mixture got in this way can approximate almost any complex probability density function efficiently. In addition, due to its greedy nature, the new EDA is very fast and efficient. The algorithm is evaluated and compared with other EDAs on a set of selected function for efficiency and performance assessment. The results show that it can perform very well on all the selected functions and that the convergence speed is faster than the previous EDAs.

The rest of this paper is organized as follows. Section 2 introduces the greedy EM for GMM estimation. Section 3

0-7803-9422-4/05/$20.00 ©2005 IEEE

presents our new continuous optimization algorithm based on greedy estimation of GMM. Section 4 presents the experiment results on a set of functions to assess the performance of the proposed EDA. Section 5 gives the conclusion and discussions.

II. Greedy EM algorithm for Gaussian Mixture Learning[20]

Finite mixture distribution is a popular and important framework for modeling population heterogeneity and is widely used in computational intelligence community. The mixture usually takes the formula as follow:

$$f_k(x) = \sum_{j=1}^{k} w_j \phi(x;\theta_j) \tag{1}$$

where $\phi(x;\theta_j)$ is the jth component model parameterized on θ_j, w_j is the mixing weight of jth component, k is the number of components in the mixture model. For Gaussian mixture model, $\phi(x;\theta_j)$ usually defined as follow:

$$\begin{aligned}\phi(x;\theta_j) &= \phi(x;\mu_j,\sigma_j) \\ &= (2\pi)^{-\frac{1}{2}}\left|\sigma_j\right|^{-\frac{1}{2}} e^{\left[-\frac{1}{2}(x-\mu_j)^T\sigma_j^{-1}(x-\mu_j)\right]}\end{aligned} \tag{2}$$

where μ_j and σ_j are the mean and covariance matrix of the jth component model, they can be collectively represented by θ_j.

In EM algorithm for estimating Gaussian Mixture, given the number of components k, initial parameters (w_0, μ_0 and σ_0, usually guessed), and a set of training data, the parameters μ, σ and w are updated iteratively to increase monotonously the likelihood of training set. [21] proves that the mixture density can be estimated by maximum likelihood via incrementally adding components to the mixture up to a desired number of components k, and if the new component is added in an optimal way, the mixture learned in this way can be as good as any other mixture in the form (1). Based on this, [20] proposes a greedy EM algorithm for learning Gaussian mixture models, which starts with a single component and adds components sequentially until some criteria is met. In more detail, the greedy EM algorithm has two primary steps that are carried out iteratively, one is to find a proper set of initial parameters of the new added component via a global search, the other one is to refine the parameters of the new component to maximum likelihood via a local search procedure.

Assume, at some point of procedure, a new component $\phi(x;\theta)$ is added to a k-component mixture $f_k(x)$, giving the new mixture

$$f_{k+1}(x) = (1-\alpha) f_k(x) + \alpha\phi(x;\theta) \tag{3}$$

where $0 < \alpha < 1$. The task is to optimize the weight α and the parameter vector θ of $\phi(x;\theta)$ to maximize the new log-likelihood

$$\begin{aligned}L_{k+1} &= \sum_{i=1}^{n} \log f_{k+1}(x_i) \\ &= \sum_{i=1}^{n} \log[(1-\alpha) f_k(x_i) + \alpha\phi(x_i;\theta)]\end{aligned} \tag{4}$$

the above procedure is implemented iteratively until some criteria is met. As a result, for a proper k, the resulting mixture has almost as high a log-likelihood as is achieved by any mixture in the form (1). Then the Maximum Likelihood learning of a general Gaussian mixture can be simplified to be a successive learning of the two-component mixtures $f_{k+1}(x)$, where the first component is the old mixture $f_k(x)$ and the second one is the Gaussian component $\phi(x;\theta)$ with $\theta = [\mu, \sigma]$. Appropriate search techniques need to be developed to optimally specify the parameters α, μ, and σ that maximize L_{k+1}.

First, global search technique is adopted to find the proper initial values of new component's parameters μ σ and α. To achieve that, we expand L_{k+1} by second order Taylor about $\alpha_0 = 0.5$ and then maximize the resulting quadratic function with respect to α.

$$\hat{L}_{k+1} = L_{k+1}(\alpha_0) - \frac{\left[L'_{k+1}(\alpha_0)\right]^2}{2L''_{k+1}(\alpha_0)} \tag{5}$$

where L'_{k+1} and L''_{k+1} are the first and second derivatives of L_{k+1} with respect to α. We can define

$$\delta(x,\theta) = \frac{f_k(x) - \phi(x;\theta)}{f_k(x) + \phi(x;\theta)}, \tag{6}$$

then a local maximum of L_{k+1} near $\alpha_0 = 0.5$ (equation (5)) can be rewritten as

$$\hat{L}_{k+1} = \sum_{i=1}^{n} \log\frac{f_k(x_i) + \phi(x_i;\theta)}{2} + \frac{1}{2}\frac{\left[\sum_{i=1}^{n}\delta(x_i,\theta)\right]^2}{\sum_{i=1}^{n}\delta^2(x_i,\theta)} \tag{7}$$

and is obtained for α equal to

$$\hat{\alpha} = \frac{1}{2} - \frac{1}{2}\frac{\sum_{i=1}^{n}\delta(x_i,\theta)}{\sqrt{\sum_{i=1}^{n}\delta^2(x_i,\theta)}} \quad (8)$$

The above procedure makes the likelihood function in (3) independent of the mixing weight α, and the next step is to find a good initialization of the center μ and covariance matrix σ of the new component. From (6) we can see that $\hat{L}_{k+1}$ depends only on $\phi(x_i;\mu,\sigma)$ which, for constant covariance matrix $\sigma = \sigma_d^2 I$, is just a function of the Euclidean distance of μ to the input point x_i. Thus, if we restrict our global search for the new μ only over the input points x_j, then evaluation of $\hat{L}_{k+1}$ for all μ requires the computation of all pair wise Euclidean distances $||x_i - x_j||$ between inputs which can be carried out only once at the beginning of the algorithm.

Second, a local search procedure is adopted to refine the parameters of the new initialized component, which is implemented by EM algorithm. Since the parameters of old mixture $f_k(x)$ remain fixed during component allocation, partial EM steps can be used to update only the α, μ, and σ of the newly inserted component, so it's simple and fast.

Consequently, the algorithm for learning a multivariate Gaussian mixture can be summarized as follow.

Algorithm1: *Greedy EM*:

1. Initialize using one component with $\mu = E[x]$ and $\sigma = \mathrm{Cov}(x)$.
2. Search over all x_j for candidate locations for the new component. Set μ to the x_j that maximizes (7).
3. Initialize the partial EM with the estimated value of μ, σ, and $\hat{a}$.
4. Apply partial EM steps until convergence: $\left|L_k^t / L_k^{t-1} - 1\right| < 10^{-6}$.
5. If $L_{k+1} \leq L_k$ then terminate, otherwise go to 2.

III. Continuous Optimization based-on Greedy Estimation of GMM

In EDA framework, by using the Greedy EM algorithm to estimate the Gaussian mixture model, we get the Greedy EM based EDA (GEMEDA), shown as follow:

Algorithm2: *GEMEDA*

1. Generate N individuals randomly $\rightarrow D_l$
2. For $l = 1,2,\ldots$ until some termination criteria is reached

(a) Select $S_e \leq N$ individuals from $D_{l-1} \rightarrow D_{l-1}^{S_e}$

(b) Using Greedy EM for GMM to estimate the probability density function based on $D_{l-1}^{S_e} \rightarrow p_l(x)$

(c) Sample N individuals from $p_l(x) \rightarrow D_l$

In GEMEDA, Gaussian mixture model is used to break the single Gaussian distribution assumption, and by adopting Greedy EM algorithm, the model structure is learned automatically and efficiently along with the learning of the model parameters, no prior knowledge about the number of components and no clustering procedure are needed, that make GEMEDA different from the previous EDAs that use Gaussian mixture model.

IV. Experimental Analysis

To compare with other previous literatures, we select five test functions used in [16] to evaluate the performance of BMMEDA. Two of them are uni-modal, and the others are multi-modal. The settings of the parameters are the same as [16].

A. Experiment Design

Func. 1: Sphere $\quad F(\vec{x}) = \sum_{i=1}^{n} x_i^2 \quad (9)$

Func. 2: SumCan

$$F(\vec{x}) = 1/(10^{-5} + \sum_{i=1}^{n} |\, y_i \,|)$$

$$y_1 = x_1 \quad (10)$$

$$y_i = x_i + y_{i-1}, i \geq 2.$$

Func. 3: TwoPeaks

$$F(\vec{x}) = \sum_{i=1}^{2} \alpha_i N(\vec{x}, \mu_i, \Sigma_i) \quad (11)$$

where $N(\vec{x}, \mu_i, \Sigma_i)$ is a multivariate normal distribution, which has n-dimensional mean vector μ and $n \times n$ covariance matrix Σ. $\alpha_1 = 1000$, $\alpha_2 = 900$, $\mu_1 = (-10,\ldots,-10)$, $\mu_2 = (10,\ldots,10)$, $\Sigma_i (i = 1,2)$ are diagonal matrices with all the diagonal elements equaling 1.

Func. 4: TreePeaks

$$F(\vec{x}) = \sum_{i=1}^{3} \alpha_i N(\vec{x}, \mu_i, \Sigma_i), \quad (12)$$

where the settings are the same with TwoPeaks plus $\alpha_3 = 500$ and $\mu_3 = (0,\ldots,0)$.

Func. 5: Shekel

$$F(\vec{x}) = \sum_{i=1}^{n} [(\vec{x} - \vec{a}_i)(\vec{x} - \vec{a}_i)^T + \vec{c}_i]^{-1} \quad (13)$$

where

i	$a_{ij}, j=1,\ldots,4$	c_i
1	2 2 2 2	0.1
2	4 4 4 4	0.2
3	8 8 8 8	0.2
4	6 6 6 6	0.4
5	3 7 3 7	0.4

The settings of the five functions are shown in Table I.

TABLE I
SETINGS FOR THE TEST FUNCTIONS

Functions	Dim	Domain	Type	Optimum
Sphere	30	[-100,100]	Min	0
SumCan	10	[-0.16,0.16]	Max	100000
TwoPeaks	5	[-100,100]	Max	10.1053
ThreePeaks	5	[-100,100]	Max	10.1053
Shekel(n=5)	4	[0,10]	Max	10.1033

We compare GEMEDA with other three EDAs: UMDAc[11], EGNA[11][13], CEGDA[16] on the test functions. We denote the number of individuals in each generation *Population*, the number of promising solutions selected from individuals in the population *Selection*. The *Population* and *Selection* of each algorithm are shown in Table II. For CEGDA, $\alpha_c = 0.05$, $\alpha_r = 0.002$.

TABLE II
EXPERIMENTAL SETTINGS FOR THE TEST ALGORITHMS

Algorithms	Population	Selecion
UMDAc	1000	500
EGNA	1000	500
CEGDA	2000	500
GEMEDA	1000	500

B. Experiment Results and Analysis

We run our algorithm 30 times independently on each function. The final results are the averages, which are shown in Table III-VII. The results of the other EDAs are all from [16]. From Table III-VII, we can see that with fewer generations, our algorithm can perform well compared with other EDAs.

Function Sphere is easy for all the four EDAs and the results are all good. The reason is that it has only one global optimum and no local optima. Moreover, the variables are independent of each other. Among the EDAs, UMDAc, which assumes single Gaussian distribution and considers no interdependencies, has the best result. That is because for uni-modal functions, single Gaussian distribution is more suitable than Gaussian mixture model; moreover, this function accords with the no interdependencies assumption very well.

Function SumCan is also a uni-modal function, but the variables are interdependent. Among the four EDAs, UMDAc performs very badly because it doesn't take any interdependencies into account. EGNA performs best; the reason might be it takes the network structure to model the interdependencies among variables that is the most important in optimizing function SumCan. GEMEDA is a little better than CEGDA.

TwoPeaks, TreePeaks and Shekel are all multi-modal functions. They all have one global optimum with one or several local optima. From the results we can see that, with only 100 iterations, GEMEDA outperforms UMDAc and EGNA, and perform almost as well as CEGDA. GEMEDA can reach the global optimum almost in every run, especially on the TwoPeaks functions.

TABLE III
EXPERIMENTAL RESULTS FOR THE SPHERE FUNCTION

Algorithms	Gen	Best	Mean	Std
UMDAc	200	1.88e-016	3.24e-016	5.59e-017
EGNA	200	5.86e-010	1.20e-009	3.40e-009
CEGDA	100	3.38e-008	3.41e-006	8.40e-007
GEMEDA	200	1.01e-013	1.31e-013	2.15e-014

TABLE IV
EXPERIMENTAL RESULTS FOR THE SUMCAN FUNCTION

Algorithms	Gen	Best	Mean	Std
UMDAc	200	698.72	221.771	116.101
EGNA	200	100000	100000	0
CEGDA	100	99834.5	99748.1	63.2197
GEMEDA	100	99996.827	99995.308	0.765

TABLE V
EXPERIMENTAL RESULTS FOR THE TWOPEAKS FUNCTION

Algorithms	Gen	Best	Mean	Std
UMDAc	400	10.1053	9.6327	0.1073
EGNA	400	10.1053	9.8324	0.0828
CEGDA	200	10.1053	10.0999	5.92e-003
GEMEDA	100	10.1053	10.1002	0.0166

TABLE VI
EXPERIMENTAL RESULTS FOR THE THREEPEAKS FUNCTION

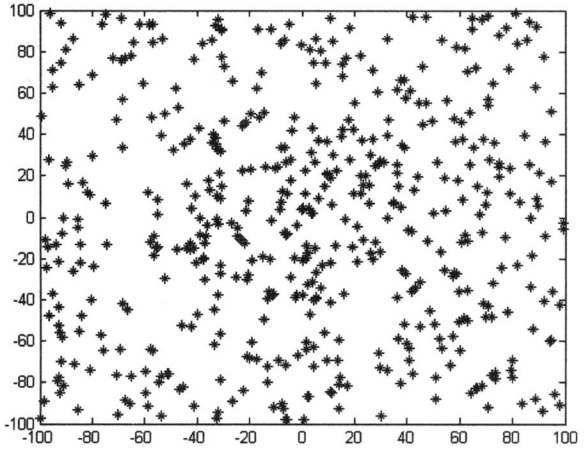
Algorithms	Gen	Best	Mean	Std
UMDAc	400	5.05266	5.05266	8.88e-016
EGNA	400	5.05266	5.05266	8.88e-016
CEGDA	200	10.1053	10.1048	7.99e-004
GEMEDA	100	10.1053	10.0988	0.01854

TABLE VII
EXPERIMENTAL RESULTS FOR THE SHEKEL FUNCTION

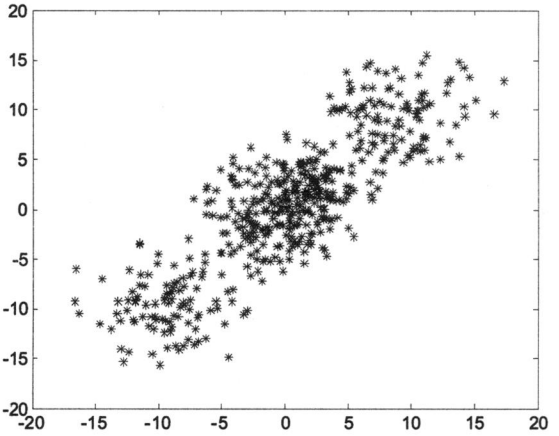
Algorithms	Gen	Best	Mean	Std
UMDAc	400	5.1877	4.7331	0.7406
EGNA	400	8.2036	4.9691	0.7786
CEGDA	200	10.1033	10.1033	8.8818e-015
GEMEDA	100	10.1033	10.0940	0.04227

Figure 1 illustrates the convergence process of the

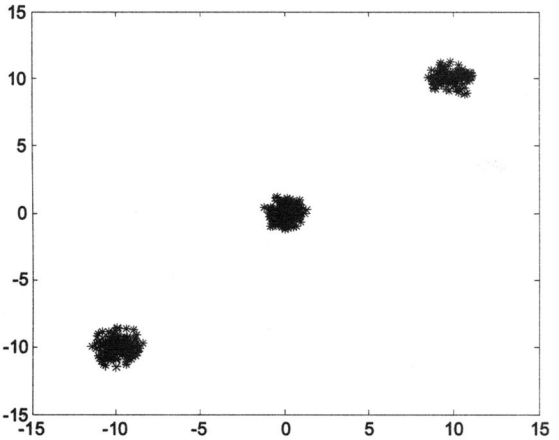
sampling area of the algorithm GEMEDA when it is used to optimize ThreePeaks function. We can see that the searching area shrinks quickly from a large space to smaller and smaller areas along with the increase of generation, where the global and/or local optimums lie in.



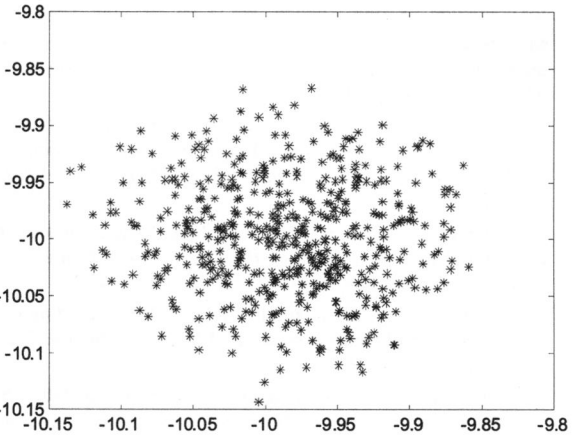
(a) samples taken at generation 1



(b) samples taken at generation 15



(c) samples taken at generation 20



(d) samples taken at generation 30

Fig. 1. the variation of sampling(searching) area along with the increase of generation, which converges from a large space to a much smaller area

VI. Conclusion

The learning of probabilistic model is the key step in EDAs. For complex problems, complex models that can describe the interdependences among variables are necessary. For these complex models, the learning task includes two aspects: parameter learning and model structure learning. In previous EDAs, the two aspects are usually implemented separately, which makes the model learning an inefficient process under the constraint of limited number of samples. In this paper, a new EDA for continuous optimization—GEMEDA is proposed. The novel contribution is that it implements the model structure learning along with the parameter learning automatically in a incremental way, which makes the new algorithm faster and more efficient.

Acknowledgement

The work is partially supported by the Nature Science Foundation of China(NSFC) under grand No.60401015, Joint Research Fund for Overseas Chinese Young Scholars of NSFC under grand No.60428202 and the Nature Science Foundation of Anhui province under grand No. 050420201.

Referances

[1] H. Mühlenbein and G. Paaß. "From recombination of genes to the estimation of distributions I. Binary parameters," *Parallel Problem Solving from Nature* – PPSN V, pages 178-187. Springer, 1998

[2] M. Pelikan, D.E. Goldberg, and F. Lobo. "A survey of optimization by building and using probabilistic models," *IlliGAL* Tech. Rep. 99018., 1999

[3] H. Mühlenbein, "The equation for response to selection and its use

for prediction," *Evolut. Comput.*, vol. 5, pp. 303–346, 1998.

[4] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm,"*Adv. Soft Comput.—Eng. Design and Manuf.*, pp. 521–535,1999.

[5] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. Genetic and Evolutionary Computation Conf. (GECCO-99)*, vol. I, 1999, pp. 525–532.

[6] G.R. Harik, F.G. Lobo, and D.E. Goldberg, "The compact genetic algorithm," in *Proceedings of the International Conference on Evolutionary Computation (ICEC)* 1998, pp. 523-528

[7] J. S. D. Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Adv. Neural Inform. Process. Syst.*, vol. 9, p. 424, 1997.

[8] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Parallel Problem Solving from Nature—PPSN V*, 1998, pp. 418–427.

[9] S. Rudlof and M.Köppen, "Stochastic hill climbing by vectors of normal distributions," in *Proc. 1st OnlineWorkshop on Soft Computing (WSC1)*, Nagoya, Japan, 1996.

[10] I. Servet, L. Trave-Massuyes, D. Stern, Telephone network traffic overloading diagnosis and evolutionary computation techniques. In *Proceedings of the Third European Conference on Artificial Evolution (AE'97)*. 1997, pp. 137-144.

[11] P. Larrañaga, R. Etxeberria, J.A. Lozano and J.M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," in *Proc. 2000 Genetic and Evolutionary Computation Conf. Workshop Program*, pp. 201–204, 2000.

[12] H. Mühlenbein, T. Mahnig, and O. Rodriguez. "Schemata, distributions and graphical models in evolutionary optimization" .in *Journal of Heuristics*, 5:215-247, 1999

[13] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization by Learning and Simulation of Bayesian and Gaussian Networks," Dept. Comput. Sci. Artific. Intell., Univ. Basque Country, Tech. Rep. EHUKZAA- IK-4/99, 1999.

[14] P. A. N. Bosman and D. Thierens, "*Expanding from discrete to continuous estimation of distribution algorithms: The IDEA," in Paralle Problem Solving from Nature—PPSN VI*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. New York: Springer-Verlag, 2000, pp. 767–776.

[15] P. A. N. Bosman and D. Thierens, "Advancing continuous IDEA's with mixture distributions and factorization selection metrics," in *Proc. Optimization by Building and Using Probabilistic Models (OBUPM) Workshop at the Genetic and Ecolutionary Computation Conf. (GECCO- 2001)*, M. Pelikan and K. Sastry, Eds., San Francisco, CA, 2001, pp. 208–212.

[16] Qiang Lu and Xin Yao. "Clustering and Learning Gaussian Distribution for Continuous Optimization", in *IEEE Transactions on Systems,Man and Cybernetics-PART C:Applications and Reviews*,VOL.35,NO.2,May 2005.

[17] M. Pelikan and D. E. Goldberg, "Genetic algorithms, clustering, and the breaking of symmetry," in *Proc. Parallel Problem Solving from Nature— PPSN VI*, Paris, France, 2000, pp. 385–394.

[18] 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

[19] L. Xu, "Rival penalized competitive learning, finite mixture, and multisets clustering," in *Proc. Int. Joint Conf. Neural Networks,* vol.II, pp. 2525-2530, 1997.

[20] NIKOS V. and ARISTIDIS L., "A Greedy EM Algorithm for Gaussian Mixture Learning", in *Neural Processing Letters* 15: 77-87, 2002.

[21] Li, J. Q. and Barron, A. R., "Mixture density estimation", In *Advances in Neural Information Processing Systems* 12, The MIT Press, 2000.