Real-valued Evolutionary Optimization using a Flexible Probability Density Estimator*

Marcus Gallagher, Marcus Frean and Tom Downs
Department of Computer Science and Electrical Engineering
University of Queensland, St Lucia Qld 4072, Australia
{*marcusg, marcusf, td*}*@csee.uq.edu.au.*

Abstract

Population-Based Incremental Learning (PBIL) is an abstraction of a genetic algorithm, which solves optimization problems by explicitly constructing a probabilistic model of the promising regions of the search space. At each iteration the model is used to generate a population of candidate solutions and is itself modified in response to these solutions. Through the extension of PBIL to Real-valued search spaces, a more powerful and general algorithmic framework arises which enables the use of arbitrary probability density estimation techniques in evolutionary optimization. To illustrate the usefulness of the framework, we propose and implement an evolutionary algorithm which uses a finite Adaptive Gaussian mixture model density estimator. This method offers considerable power and flexibility in the forms of the density which can be effectively modeled. We discuss the general applicability of the framework, and suggest that future work should lead to the development of better evolutionary optimization algorithms.

1 INTRODUCTION

Shumeet Baluja's Population-Based Incremental Learning (PBIL) algorithm was proposed as an optimization method which "removes the genetics from the genetic algorithm" [Baluja, 1994],[Baluja and Caruana, 1995].

Step 1:	Initialize probability vector $\mathbf{z} = (0.5, \ldots, 0.5)$
Step 2:	Generate pop. using prob. vector $D^{(k)} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$
Step 3:	Eval. cost fn. at each sample point $E^{(k)} = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)\}$
Step 4:	Find best sample point in population $\mathbf{x}^* : f(\mathbf{x}^*) = min(D)$
Step 5:	Update probability vector $z_i = (1-\eta)\mathbf{z}_i + \eta\mathbf{x}^*$
Goto 2:	(until termination condition)

Table 1: The basic PBIL Algorithm.

Rather than *maintaining a population of solutions* in the search space, PBIL *maintains a probabilistic model* of the promising regions of the search space, and uses this model to generate populations at each time step. Originally formulated for optimizing bitstrings, PBIL implements evolution through the construction of a probability vector

$$\mathbf{z} = (z_1, \ldots, z_n),\ z_i = P(x_i = 1)$$

where n is the length of the bitstring to be optimized, x_i is the i^{th} bit of a candidate solution bitstring, and z_i is the probability of the i^{th} bit in a bitstring being equal to 1. At each iteration of the algorithm, a population of bitstrings is generated by sampling from this distribution. Then, PBIL updates the probability model in the direction of the best member of the current population, governed by a learning rate parameter η. Pseudo-code for a simple PBIL variant is given in Table 1. More sophisticated versions update towards the mean of the k best and away from the k worst members. A further variation is to mutate the probability vector after each update step.

Experimental evidence has shown that PBIL can

*In W. Banzhaf et al., editors, *Proc. Genetic and Evolutionary Computation Conference (GECCO'99)*, pp. 840-846, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

significantly outperform traditional genetic algorithms on a variety of test problems [Baluja, 1995]. In [De Bonet et al., 1997], the Mutual-Information-Maximizing Input Clustering (MIMIC) algorithm is presented which improves on PBIL by modeling pairwise conditional probabilities between bits. Because PBIL models the probability distribution of each bit independently, MIMIC is able to offer improved performance on problems where the structure of the solution bitstring is of major importance, by capturing some of the relationships between bits in the bitstring. Baluja has also considered more sophisticated representations of pairwise conditional probabilities, where a tree of pairwise conditional probabilities is constructed (Optimal Dependency-Tree). Allowing only pairwise dependencies, the maximum spanning tree minimizes the (Kullback-Leibler) divergence between the true and estimated distributions [Baluja and Davies, 1997].

The extension of PBIL to the optimization of functions of real-valued variables has only recently been considered. Rudlof et. al. [Rudlof and Köppen, 1996] use independent Gaussian distributions to model each variable of solution vectors. Each univariate Gaussian is specified by its mean, μ_i and variance, σ_i parameters. A bounded feasible region is defined as the search space, and the probability model is initially centered in this region. The mean value of each Gaussian is updated according to the PBIL update rule (in this implementation using the average of the k best samples). The variance values are initialized to some large value, in an attempt to ensure that the feasible region is well-covered (in probability), and are then annealed towards some small value on a geometrically decaying schedule

$$\begin{aligned} \sigma_i &= \sigma_i \times \gamma \ ; \\ \gamma &\leq 1. \end{aligned} \tag{1}$$

Servet et. al. [Servet et al., 1997] propose and implement a different Real-coded version of PBIL. Assuming that each variable is constrained to some interval $[i_{Low}, i_{Up}]$, each component z_i of the probability vector represents the probability that the i^{th} variable is greater than $(i_{Low} + i_{Up})/2$. This probability vector is updated using the PBIL update rule, until $z_i > 0.9$ or $z_i < 0.1$, when the interval for variable i is then halved in the appropriate direction, z_i is reset to 0.5 and the algorithm continues.

Finally, Sebag et. al. [Sebag and Ducoulombier, 1998] also consider using Gaussian distributions as a probability model for Real-valued PBIL. The μ_i are updated by the PBIL rule and several different methods for updating the σ_i are discussed. The simplest choice is to simply fix all σ_i at some constant value, equivalent to setting the γ in (1) to unity.

Whilst Real-valued PBIL algorithms (as for the binary case) have shown good results on a number of test problems, several difficulties remain in applying the algorithms. In particular, the variety of methods used for constructing and evolving a probabilistic model make it difficult to determine which is suitable *a priori* on a given problem, as well as introducing a number of user parameters that must be selected with appropriate values to ensure good results.

The work on PBIL suggests the principle of using probabilistic sampling methods based on density estimation for optimization problems. In the following section we consider a general framework for these algorithms. From this formulation a number of insights can be gained, such as the fact that one form of Real-valued PBIL is equivalent to a $(1, \lambda)$-Evolutionary Strategy. In Section 3 we propose the use of a finite adaptive Gaussian mixture model density estimator in this framework to improve the speed and representational power of the algorithm. Section 4 describes empirical results which demonstrate the advantages of this approach, and Section 5 discusses the results and concludes.

2 EVOLUTIONARY OPTIMIZATION AND PROBABILISTIC MODELING

The conceptual distinction between PBIL and traditional Evolutionary Algorithms is the notion that what is maintained and evolved during the execution of the algorithm is a probabilistic model, rather than a finite population of solutions. The notion of a lasting population present in natural evolution, together with the traditional genetic operators (e.g, crossover) is removed in an attempt to provide faster, simpler and more robust optimization algorithms. What remains are the principles of

(i) Parallel search using a population of solutions at each iteration of the algorithm;

(ii) Coupling of the current search results such that the information gained through the parallelism is used to guide future search.

We agree with the idea that adopting this approach to evolutionary algorithms provides a useful perspective on their workings. From a more practical viewpoint, a general and common framework which follows this approach allows the development of new kinds of evolutionary algorithms. This paper presents first steps in this direction.

2.1 A GENERAL PROBABILISTIC SAMPLING FRAMEWORK

The task of optimization over a set of n Real-valued parameters collected into a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is given as finding

$$\mathbf{x}^* = \arg\min f(\mathbf{x}); \;\; \mathbf{x} \in I\!R^n$$

where $f(\mathbf{x})$ is the error or cost function.

Assume the existence of a probability density function $p(\mathbf{x})$, so

$$\begin{aligned} p(\mathbf{x}) &\geq 0 \\ \int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} &= 1. \end{aligned}$$

At each iteration (generation) t of the algorithm, we generate k candidate solutions for the optimization problem by sampling k times ($D_{(k),t} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$) from the probability distribution $p_t(\mathbf{x})$. For each sample solution (individual) we evaluate the error function $f(\mathbf{x}_k)$, giving

$$E_{(k),t} = \{f(\mathbf{x}_{1,t}), \ldots, f(\mathbf{x}_{k,t})\}.$$

The probability model is then modified according to some procedure Φ, that may depend, for example, on the information gained from the sampling and error function evaluation process of the current and/or previous generations. This produces the probability model for the next generation, $p_{t+1}(\mathbf{x})$, and the process repeats.

This general algorithm may be arrived at by abstracting Steps 4 & 5 of PBIL to a procedure that in some way updates the current probability model (e.g, incorporating information obtained from the current population). For PBIL variants incorporating operations such as mutating the probability vector, or using a number of the best and worst population members to update the probability vector, these operations are simply considered to be part of the update process, Φ. Although this means that considerable complexity is "hidden" in the Update procedure Φ, it describes the evolutionary process of the algorithm concisely and allows flexibility in this "Meta-step".

Step 1:	Initialize probability model $p_0(\mathbf{x}), \; t = 0$
Step 2:	Sample from $p_t(\mathbf{x})$ $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$
Step 3:	Evaluate $E = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)\}$
Step 4:	Update probability model $\Phi : p_t(\mathbf{x}) \rightarrow p_{t+1}(\mathbf{x})$
Step 5:	Increment time $t = t + 1$
Goto 2:	(until termination condition)

Table 2: General Probabilistic Population-Based Algorithm.

In order to use this framework to specify a particular algorithm, it is only necessary to give a form for Φ, an initial probability model $p_0(\mathbf{x})$ and a population size, k. Real-valued PBIL, using Gaussian distributions with constant variance [Sebag and Ducoulombier, 1998] can be stated as

$$\begin{aligned} p_0(\mathbf{x}) &= \mathcal{N}(\mu_0, \sigma) \\ \Phi : p_{t+1}(\mathbf{x}) &= \mathcal{N}(\mu_{i,t+1}, \sigma); \\ \text{with } \mu_{i,t+1} &= (1 - \eta)\mu_{i,t} + \eta \mathbf{x}_t^* \end{aligned}$$

where $\mathbf{x}_t^*$ is the best member of the population at time t, i.e

$$\mathbf{x}_t^* : f(\mathbf{x}_t^*) = \min(E_{(k),t}).$$

If we set $\eta = 1$, the Gaussians are re-centered at each time step to the position of the best sample found in the previous generation. This is in fact equivalent to the simplest $(1, \lambda)$-ES [Schwefel, 1995] and is given by

$$\begin{gathered} p_0(\mathbf{x}) = \mathcal{N}(\mu_0, \sigma) \\ \Phi : p_{t+1}(\mathbf{x}) = \mathcal{N}(\mathbf{x}_t^*, \sigma). \end{gathered}$$

The other Real-valued PBIL algorithms discussed above can also be represented within this framework in a similar way.

Previous Real-valued PBIL algorithms have used a very simple underlying probability model, such as a univariate Gaussian, to model the regions of the search space which are promising for search. The above framework suggests the use of more powerful probabilistic models. To be applicable in a PBIL-like algorithm, we require methods which "learn" to model the search space with a biased view towards locating good extrema. This means that an iterative procedure for updating the model is needed, so that after each generation of points is generated, the model can adjust itself in response to some of the information gained from that generation (e.g, the best/worst point(s) found).

In addition to existing familiar optimization techniques, there are many other methods which allow the iterative (sequential) construction of a representation of (unlabeled) data. In the following section we provide an example of one such technique from multivariate probability density estimation.

3 AN APPLICATION: THE ADAPTIVE MIXTURE MODEL ESTIMATOR

An important possibility in the development of better evolutionary optimization algorithms is the ability to model more sophisticated and powerful probability distributions over the parameter search space [De Bonet et al., 1997], [Baluja and Davies, 1997]. The field of Multivariate Probability Density Estimation (see, e.g, [Scott, 1992]) provides a number of methods which aim to address this issue. These include Kernel, Mixture Model and Nearest Neighbour methods.

3.1 ADAPTIVE MIXTURE MODEL

In order to illustrate the process of applying an arbitrary density estimation technique, we consider the Adaptive Mixture Model of Priebe [Priebe, 1994]. Here the probability density $p(\mathbf{x})$ is represented by the sum of a number of component distributions

$$p(\mathbf{x}) = \sum_{j=1}^{M} \pi_j p(\mathbf{x}|j)$$

where M is the number of mixture components and π_j is the mixing coefficient for the j^{th} component. The coefficients allow the contribution that each component makes to the sum to be varied and satisfy the conditions

$$\sum_{j=1}^{M} \pi_j = 1$$
$$0 \leq \pi_j \leq 1.$$

Gaussians are the most popular choice of component distribution and we use them in the implementation discussed below. Gaussian mixture models offer considerable flexibility in the forms of distribution they can adequately approximate. In particular, multimodal distributions can be effectively modeled (see, e.g, [Bishop, 1995] for an introduction and further references). Mixture models are the subject of ongoing development and a number of issues remain regarding their implementation, but the Adaptive Mixtures algorithm undoubtedly has some attractive properties. Firstly, it uses a recursive update rule (meaning that the model assumes that data points arrive sequentially, as they do here) to update the model after observing each new point (this is the Φ procedure in the above framework). These update equations are:

$$\begin{aligned}
\rho_{j,t+1} &= \pi_{j,t} \frac{p_t(\mathbf{x}_t^*|\pi_{j,t})}{p_t(\mathbf{x}_t^*)}, \\
\pi_{j,t+1} &= \pi_{j,t} + t^{-1}(\rho_{j,t+1} - \pi_{j,t}), \\
\mu_{j,t+1} &= \mu_{j,t+1} + \pi_{j,t}^{-1} t^{-1} \rho_{j,t+1}(x_{j,t}^* - \mu), \\
\sigma_{j,t+1}^2 &= \sigma_{j,t}^2 + \pi_{j,t}^{-1} t^{-1} \rho_{j,t+1} \\
&\quad \times ((x_{j,t}^* - \mu_{j,t})(x_{j,t}^* - \mu_{j,t}) - \sigma_{j,t}^2).
\end{aligned}$$

Secondly, M is allowed to vary during execution of the algorithm, beginning with a single component and adding further components if data is observed which is not adequately accounted for by the existing model. A simple approach that we use is to calculate the Mahalanobis distance from the new data point, $\mathbf{x}_t^*$, adding a new component if this distance is greater than some prespecified threshold value, T_C. In this case, the model is updated using

$$\begin{aligned}
\mu_{M+1,t+1} &= x_{j,t}^*, \\
\sigma_{M+1,t+1} &= \sigma_{0,t}, \\
\pi_{i,t+1} &= \pi_{i,t}(1 - t^{-1}), \\
\pi_{M+1,t+1} &= t^{-1}, \\
M &= M + 1.
\end{aligned}$$

A new component is assigned some small mixing coefficient, and the remaining coefficients are adjusted accordingly.

4 EMPIRICAL RESULTS

We use two problems to demonstrate the application of the adaptive mixture estimator in an evolutionary optimization algorithm. These are the standard quadratic or "sphere" model

$$f_1(\mathbf{x}) = x_1^2 + x_2^2$$

and the well-known six-hump camel-back/Branin function [Törn and Žilinskas, 1989]

$$f_2(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$$

f_1 is useful to demonstrate the convergence behaviour of the algorithm. f_2 is a multimodal cost function with local and global minima and so is able to show how the mixture model constructs a probabilistic search model which is able to capture the complex structure of the underlying search space.

For all algorithms used, the size of the population was set to 5 ($\lambda = 5$ for the $(1, \lambda) - ES$), and the search was initialized randomly in the region $[-5, 5]$. Each experiment was run for 1000 iterations/generations.

For problem f_1, we ran four different algorithms which are realized in the above framework:

- The Adaptive Mixture Model Algorithm ($AMix$) - maximum number of components in the mixture model = 10, initial variance of components new to the mixture = 2.0, Mahalanobis threshold for adding new components = 0.5.
- Real-valued PBIL using Gaussians ($PBIL_{CV}$) - fixed mean value of Gaussians = 1.0, fixed variance value of Gaussians = 1.0, $\eta = 1.0$.
- Real-valued PBIL using Gaussians ($PBIL_{Anneal}$) - fixed mean value of Gaussians = 1.0, initial variance value = 1.0, $\gamma = 0.82$, $\eta = 1.0$
- $(1, \lambda) - ES$ (ES) - standard Gaussian perturbations for each variable (initial variance of perturbations = 1.0), lognormal perturbations of step sizes using standard constants to control the overall ($\tau = \sqrt{2\sqrt{n}}^{-1}$) and individual ($\tau' = \sqrt{2n}^{-1}$) step sizes. For more details see, e.g, [Schwefel, 1995].

The results over 10 runs of each algorithm are shown in Table 3. A typical run for each algorithm is shown in Fig. 1. Although $AMix$ gives the best results here (in terms of lowest error value reached), this is mainly artificial - we have chosen parameters for each algorithm to illustrate convergence behaviour. In addition, f_1 and f_2 are quite well-suited to modelling by Gaussian Mixtures. We see that $AMix$ approaches the minimum at a rate $\sim 1/t$. In contrast, the constant variance value means that $PBIL_{CV}$ initially converges slowly

ALGORITHM	f($\mathbf{x}$)
$AMix$	3.5e-7(2.9e-7)
$PBIL_{Anneal}$	0.9(1.4)
$PBIL_{CV}$	4.5e-6(3.6e-6)
ES	0.07(0.19)

Table 3: Results for the f_1 problem: mean(std. dev.).

due to the steps it takes being too small, increasing as the variance value becomes better suited to the local properties of the cost function, and then slows again as the variance becomes too large to generate the step sizes required to continue fast convergence. Annealing the variance ($PBIL_{Anneal}$) can result in convergence $\gg 1/t$; however, if γ is chosen incorrectly, it can also result in the convergence being "frozen", as shown here. The ES can also provide very fast convergence, but produces poor results with high variability in this instance. Population size and the limited number of iterations are also important considerations.

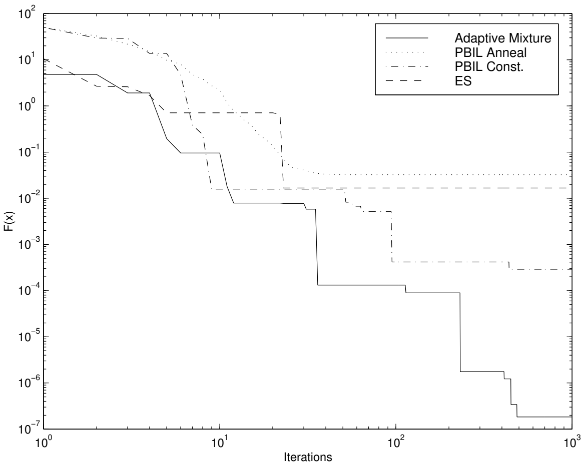

Figure 1: An illustrative run of each algorithm on f_1.

We use problem f_2 to demonstrate the typical dynamics of the $AMix$ algorithm. In the example run shown, 4 components are created in the mixture. Fig. 2 shows the evolution of the mean values of the components (means for x_1 are shown as solid lines and for x_2 as dashed lines). We see different components approaching different mean values asymptotically. This problem has a pair of conjugate global minima at (0.08983,-0.7126) and (-0.08983,0.7126), the latter of which is approached by one of the components. This shows how the Gaussian components in each dimension evolve to form a flexible multimodal density.

Fig. 3 shows the evolution of the standard deviations of the components. We see each component levelling out to a range of different values (though the values for x_1 and x_2 are similar in each case). Some of the components can be seen to "converge" as their variances approach zero, while others remain at larger values. Finally in Fig. 4 the mixing coefficients are shown. The component which approaches the global minimum is the topmost curve, becoming more heavily weighted in the mixture at the cost of the others. Two of the other components have significant mixing coefficients - these correspond to the four highest variance curves in Fig. 2. These components, whilst not converging towards either global minimum, remain part of the search by having corresponding variance values which allow them to produce a reasonable amount of high-quality sample solutions. Other components are effectively removed from the model as their coefficient values become very small.

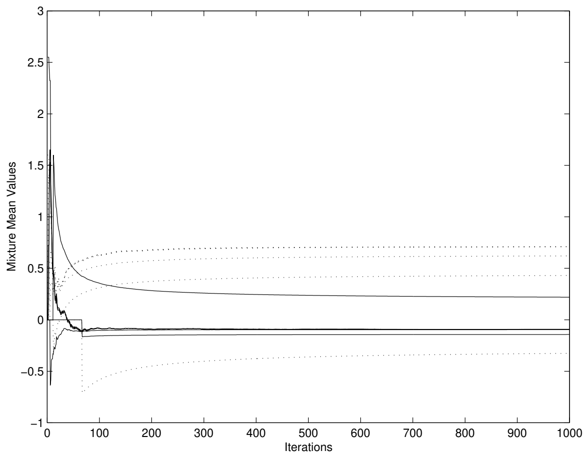

Figure 2: Evolution of mean values for a run of *AMix*.

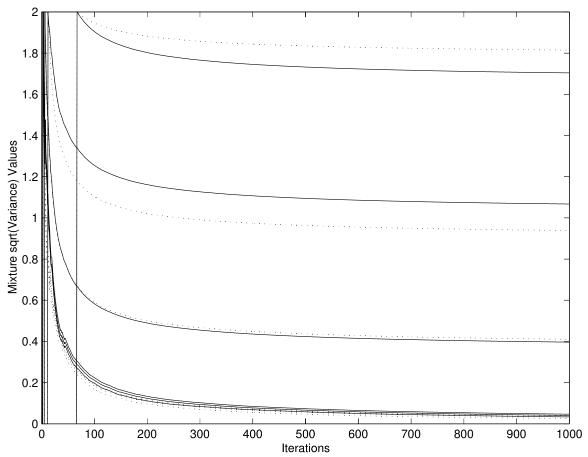

Figure 3: Evolution of the variance values for a run of *AMix*.

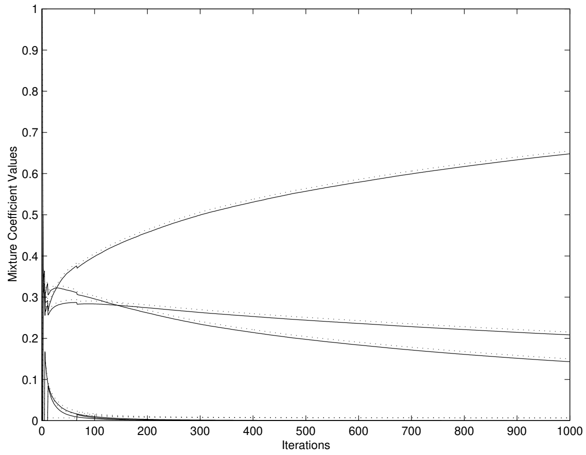

Figure 4: Evolution of the mixture coefficients for a run of *AMix*.

5 DISCUSSION

We have discussed a general framework for the specification of evolutionary optimization algorithms which construct a probabilistic model of the search space. This framework is general enough to include as special cases a large number of familiar optimization techniques, as well as opening up the possibility of using techniques such as unsupervised learning, cluster analysis and probability density estimation. This was motivated by recent research on PBIL algorithms, which aims to "abstract away" the details of the population operators of traditional genetic and evolutionary algorithms and focus on how to best use the information gained through coupled parallel search to guide further search. To demonstrate the applicability of the framework, we apply a sophisticated method from probability density estimation: the Adaptive Mixture Model. Using this density estimator in an optimization context allows considerably more powerful models to be explicitly constructed as part of the search process.

It is not our purpose to suggest that the proposed Adaptive Mixture optimization method is superior to any existing method in any general sense. Typically it is possible to find problems which are most suitable to any given algorithm at hand (and *vice versa*). The goal of this research is to support the development of evolutionary algorithms which have the potential to adapt to and exploit the structure of the search space, through flexible modeling techniques. This is to be traded-off with the significant increase in computational and storage requirements that these methods often demand.

Acknowledgements

This work was supported by a University of Queensland Postgraduate Research Scholarship for the first author.

References

[Baluja, 1994] Baluja, S. (1994). Population-Based Incremental Learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University.

[Baluja, 1995] Baluja, S. (1995). An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University.

[Baluja and Caruana, 1995] Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. Technical Report CMU-CS-95-141, Carnegie Mellon University.

[Baluja and Davies, 1997] Baluja, S. and Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Technical Report CMU-CS-97-107, Carnegie Mellon University.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford.

[De Bonet et al., 1997] De Bonet, J. S., Isbell, Jr., C. L., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, volume 9, pages 424–430.

[Priebe, 1994] Priebe, C. E. (1994). Adaptive mixtures. *Journal of the American Statistical Association*, 89(427):796–806.

[Rudlof and Köppen, 1996] Rudlof, S. and Köppen, M. (1996). Stochastic hill climbing with learning by vectors of normal distributions. In *1st Online Workshop on Soft Computing*, Retrieved from http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/ (8/12/99).

[Schwefel, 1995] Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Wiley, New York.

[Scott, 1992] Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley, New York.

[Sebag and Ducoulombier, 1998] Sebag, M. and Ducoulombier, A. (1998). Extending Population-Based Incremental Learning to continuous search spaces. In et. al., A. E., editor, *Parallel Problem Solving from Nature - PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 418–427, Berlin, New York. Springer.

[Servet et al., 1997] Servet, I., Travé-Massuyés, L., and Stern, D. (1997). Telephone network traffic overloading diagnosis and evolutionary computation techniques. In et. al., J.-K. H., editor, *3rd European Conference on Artificial Evolution (AE'97)*, volume 1363 of *Lecture Notes in Computer Science*, pages 137–144, Berlin, New York. Springer.

[Törn and Žilinskas, 1989] Törn, A. and Žilinskas, A. (1989). *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin.