Population-Based Continuous Optimization, Probabilistic Modelling and Mean Shift

Marcus Gallagher marcusg@itee.uq.edu.au
School of Information Technology and Electrical Engineering,
University of Queensland, Brisbane QLD 4072, Australia

Marcus Frean marcus@mcs.vuw.ac.nz
School of Mathematical and Computing Sciences,
Victoria University, PO Box 600, Wellington, New Zealand

Abstract
Evolutionary algorithms perform optimization using a population of sample solution points. An interesting development has been to view population-based optimization as the process of evolving an explicit, probabilistic model of the search space. This paper investigates a formal basis for continuous, population-based optimization in terms of a stochastic gradient descent on the Kullback-Leibler divergence between the model probability density and the objective function, represented as an unknown density of assumed form. This leads to an update rule that is related and compared with previous theoretical work, a continuous version of the population-based incremental learning algorithm, and the generalized mean shift clustering framework. Experimental results are presented that demonstrate the dynamics of the new algorithm on a set of simple test problems.

Keywords
Probabilistic modelling, estimation of distribution algorithms, population-based incremental learning, mean shift, continuous optimization.

1 Introduction

Evolutionary Algorithms (EAs) are a broad class of methods that apply mechanisms inspired by biological evolution to the solution of optimization problems. An optimization problem can be constructed through the representation of a problem solution as an (n-dimensional) parameter vector, $\mathbf{x} = (x_1, \ldots, x_n)$, together with an objective function $f(\mathbf{x}) : \mathcal{S} \rightarrow I\!R$, where $\mathcal{S}$ is the set of feasible solutions. EAs are applicable where no further information is available regarding the optimization problem, for example derivatives of f. Although the ultimate goal is to find a globally optimal solution vector

$$\mathbf{x}^* = \mathrm{argmax} f(\mathbf{x})$$

in practice, the aim is often to find as good a solution as possible given constraints such as available computation time (Törn and Žilinskas, 1989). The objective function may present difficulties such as local maxima, discontinuities or be subject to noise. In such situations, EAs and other heuristic and meta-heuristic methods have gained wide popularity and demonstrated impressive performance.

By convention, an EA produces an iterative search process using a population $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ of sample points (candidate solutions). Beginning with a typically randomly generated population, each iteration (generation) of the EA involves the application of a set of genetic operators (e.g, selection, recombination and mutation) to evolve the population for the following generation. In this way an EA explores the search space, using information about the objective function obtained from individual candidate solutions to direct future search. To take a slightly different view, the population (being the only "information" that persists into the next generation) can be thought of as an implicit model of the potentially promising regions of the search space (Baluja, 1997).

In recent years, a number of researchers have focused specifically on this notion of modelling the search space. That is, the model is made an explicit part of the search process, and the algorithm is viewed as performing optimization via the construction and utilization of an inductive model of the solution space. Although several different modelling techniques have been adopted from statistics and machine learning (Boyan et al., 2000), most of the work within the EA community has employed a probabilistic model of the solution space. These methods have become known as Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2002), Optimization by Building and Using Probabilistic Models (OBUPM) (Pelikan et al., 2002) or Probabilistic Modelling Evolutionary Algorithms (Gallagher, 2000). To avoid confusion, in this paper we will refer to these methods as EDAs. A general outline of an EDA is:

1. Initialize the probability model, $Q(\mathbf{x})$.
2. Create a population of points X by sampling from $Q(\mathbf{x})$.
3. Evaluate the objective function $f(\mathbf{x})$ for each point in the population: $F = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)\}$.
4. Update $Q(\mathbf{x})$ using X and F, via a Selection operator.
5. Goto step 2 (until termination condition).

The majority of previous work in combining explicit modelling and EAs has focused on discrete (binary) optimization problems ($\mathbf{x} \in \mathcal{S} \subseteq \{0,1\}^n$), for reviews see (Larrañaga, 2002; Pelikan et al., 2002). One of the first and more widely-known algorithms of this kind is Population-based incremental learning (PBIL) (Baluja, 1994). The PBIL model is a probability vector that provides a simple realization of this notion, modelling solutions encoded as fixed length binary vectors using independent probabilities for each bit value $x_i \in \{0,1\}$. Subsequent work in binary search spaces has involved using more powerful probabilistic models, which can be considered as Bayesian networks with various structural (dependency) constraints (Baluja and Davies, 1997; De Bonet et al., 1997; Etxebrria and Larrañaga, 1999; Pelikan et al., 1999). The motivation for such work is that discovering and exploiting problem structure (i.e., relationships between variables) is an effective way to develop algorithms that scale favourably to high-dimensional problems. In general, this is a key goal throughout machine learning.

Continuous optimization problems ($\mathbf{x} \in \mathcal{S} \subseteq \mathbb{R}^n$) are the focus of several EAs, notably Evolution Strategies (ES) (Schwefel, 1995) and Evolutionary Programming (Fogel, 1995). At their simplest, these algorithms search by generating a new population of sample points as Gaussian perturbations of the current point(s). It is possible to make a connection between continuous EAs such as these and the general EDA framework above. This is done by focusing on the evolution of a probabilistic model itself (which is used to generate candidate solutions) rather than the direct evolution of a finite population of candidate solutions. In fact, it can be shown that simple evolution strategies and a continuous version of PBIL based on a Gaussian model (Rudlof and Köppen, 1996; Sebag and Ducoulombier, 1998) are equivalent when the PBIL learning rate is set to its maximum value (Gallagher, 2000). These methods are based on greedy heuristics aided by the stochastic and distributed nature of the search. A number of other models based on density estimators have been proposed for continuous problems, such as histogram, kernel and mixture models (Bosman and Thierens, 1999; Bosman and Thierens, 2000; Gallagher et al., 1999; Gallagher, 2000). While potential advantages of these approaches have been demonstrated, few detailed experimental studies and comparisons of the algorithms are currently available. Furthermore, while the models used in these algorithms are based on probabilistic methods, theoretical understandings of the behaviour of such algorithms are in relatively early stages of development.

This paper explores a theoretical foundation for continuous, population-based optimization using probabilistic modelling. In Section 2, optimization is formulated as incrementally adapting a known probability density $Q(\mathbf{x})$ to approximate an unknown probability density $P(\mathbf{x})$, where the latter is of an assumed form and depends on the objective function $f(\mathbf{x})$. The adaptation considered minimizes the Kullback-Leibler divergence between $Q(\mathbf{x})$ and $P(\mathbf{x})$ using a stochastic gradient descent. We compare this result with a similar existing framework formulated for binary search spaces. The relationship between the framework and existing algorithms (continuous-PBIL and the $(1, \lambda)$-ES) is discussed. In Section 3, wider relationships between the developed framework and the generalized mean shift clustering framework are considered. Section 4 presents some experimental results that illustrate the dynamics of the algorithm compared to the continuous PBIL algorithm and Section 5 provides some conclusions and discusses directions for future work.

2 Probabilistic Modelling as Gradient Descent

A simple probabilistic model for a multi-dimensional search space $I\!R^n$ assumes that each variable x_i can be modelled independently; i.e. the joint density $Q(\mathbf{x})$ factorizes as

$$Q(\mathbf{x}) = \prod_{j=1}^{n} Q(x^j); \mathbf{x} = (x^1, \ldots, x^n) \tag{1}$$

Given this restriction, it is sufficient to consider the univariate problem ($n = 1$). The optimization task is assumed to be one of unconstrained maximization.

Suppose we have an objective function $f(x)$ where $x \in I\!R$. Consider a representation[1] of the objective function $f(x)$ using a Boltzmann distribution

$$P(x) \quad = \quad \frac{1}{Z} \exp\left(\frac{f(x)}{T}\right). \tag{2}$$

For small T (akin to the temperature in physical systems) all the probability mass concentrates around the highest points on the surface. One appealing feature of the Boltzmann distribution is that it is invariant to arbitrary "vertical" translations of the objective function f. Z is a normalization factor (the partition function) to constrain $P(x)$ to be a true probability distribution.

[1]Such a representation will exist under general conditions.

Assume that the form of f, and therefore $P(x)$, is unknown. One approach to the optimization of f is to consider approximating $P(x)$ with some model distribution, $Q(x)$, whose analytical form *is* known. For example we could use a Gaussian:

$$Q(x) \;=\; \mathcal{N}(\mu, v) \;=\; \frac{1}{(2\pi v)^{1/2}} \exp\left[-\frac{\parallel x-\mu \parallel^2}{2v}\right]$$

Now consider the Kullback-Leibler divergence between the known distribution, $Q(x)$ and the unknown one, $P(x)$:

$$K_{Q,P} = \int_x Q(x) \log \frac{Q(x)}{P(x)} \; dx$$

Given the assumptions above, a well-justified approach is to minimize $K_{Q,P}$[2]. This is because we search by sampling from our model Q and, with high probability, samples from Q will have high objective function values when $K_{Q,P}$ is minimized. This optimization problem is over the parameter values of Q (for the Gaussian above, the mean μ and variance v). Unlike the original optimization problem, $K_{Q,P}$ is differentiable, and we can consider performing gradient descent on this measure to improve our model by adapting its parameters. It is easy to check that the gradient with respect to a model parameter θ is

$$\frac{\partial K_{Q,P}}{\partial \theta} \;=\int_x \left[1 \,+\, \log \frac{Q(x)}{P(x)}\right] \frac{\partial Q(x)}{\partial \theta} \; dx \tag{3}$$

Now, since $Q(x)$ is Gaussian,

$$\log Q(x) \;=\; -\,\frac{\parallel x-\mu \parallel^2}{2v} \;-\frac{1}{2}\log(2\pi v)$$

and we can substitute for the $\log P$ term in (3). Doing this gives the direction of the gradient of $K_{Q,P}$ with respect to θ as

$$\int_x \left[1 - \frac{\parallel x-\mu \parallel^2}{2v} - \frac{f(x)}{T} - \log\left(\frac{\sqrt{2\pi v}}{Z}\right)\right] \;\frac{\partial Q(x)}{\partial \theta} \; dx \tag{4}$$

In the following, we assume that v is given. For the mean μ of the Gaussian for $Q(x)$, we have

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

Substituting this into (4) gives

$$\frac{\partial}{\partial \mu} K_{Q,P} \;=\; \int_x \left[1 - \frac{\parallel x-\mu \parallel^2}{2v} - \frac{f(x)}{T} - \log\left(\frac{\sqrt{2\pi v}}{Z}\right)\right] \;Q(x)\frac{(x-\mu)}{v} \; dx$$

Notice that the term $(x-\mu)$ outside the square brackets is odd about μ, but that all terms in the square brackets apart from $f(x)$ are even about μ in x, as is $Q(x)$ itself. Thus each term apart from that involving $f(x)$ is an integral over all x of an odd function, which is zero since a Gaussian goes to zero faster than any of the other terms.

[2]Note this is not the same as the divergence $K_{P,Q}$, which is the form more commonly used in machine learning algorithms.

In particular this disposes of the unknown normalization Z that would otherwise be troublesome. The gradient now simplifies to

$$-\frac{1}{vT}\int_x Q(x)\ \ (x-\mu)\ \ f(x)\ \ dx$$

Everything in this equation is weighted by $Q(x)$, so a stochastic approximation to the gradient can be obtained by replacing the integral with a collection of samples $X = \{x_1, \ldots, x_k\}$ from $Q(x)$,

$$\frac{\partial}{\partial\mu}K_{Q,P} \ \approx\ -\frac{1}{kvT}\sum_{x_i\in X}(x_i-\mu)\ f(x_i)$$

This approximation improves as the number of samples increases, and its expected value is unaffected if we shift all f by a constant amount. Here we use $\hat{f}(x) = f(x) - \bar{f}$, where $\bar{f} = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$ is the sample mean of f in X.

The above suggests an iterative procedure for gradient descent of $K_{Q,P}$ (the minus sign disappears from the previous expression, since we aim to move in the directions of negative gradient). At each iteration (i.e., generation) we take some samples from $Q(x)$, evaluate $f(x_i)$ for each of them, and then move the mean μ towards each sample by an amount proportional to its objective function value,

$$\Delta\mu = \alpha\sum_{x_i\in X}(x_i-\mu)\ \hat{f}(x_i) \tag{5}$$

where α plays the role of a learning rate.

This simple algorithm performs stochastic gradient descent (with respect to μ) in the Kullback-Leibler divergence between a multimodal probability distribution which can be made arbitrarily sharply peaked at the maximum of f, and a unimodal (here Gaussian) approximation to that distribution. We will refer to this algorithm below as $\mathrm{P}_{\mathrm{KLD}}$.

Note that, unfortunately the gradient with respect to the model variance v does not simplify in this way: in particular the term involving the normalization Z does not vanish. This means that in the case of the variance, no simple expression for the exact gradient is available. Whilst estimates of the gradient such as those obtained by MCMC or variational techniques may be possible, in this paper we simply take v to be a constant.

2.1 Relationship to Previous Work

There is a close relationship between the framework developed above and work done independently by Berny (2001), although this previous work is developed for binary search spaces, and the derivation is somewhat different (see (Berny, 2000a; Berny, 2000b) for other theoretical insights). In this section we compare and contrast the framework developed above to this previous work. González et al. give a different analysis of PBIL in discrete spaces (González et al., 2000; González et al., 2002).

Berny develops a framework that casts combinatorial optimization as a statistical learning task, to construct a probability distribution model that minimizes one of two criteria. Using our notation from above, the first criteria considered is the KL divergence between a known model distribution and an unknown distribution that is dependent on f, of the form given in (2) above. The KL divergence in discrete form is used, and under the assumptions that: (a) the temperature parameter, T is a constant; (b) $\sum_{x \in \mathcal{S}} Q(x) = 1$ (where $\mathcal{S}$ is the set of all possible bitstrings, x), the problem of minimizing the KL divergence is reduced to minimizing the "free energy" of the system, F, where

$$\begin{aligned} F &= E - TS && (6) \\ &= \sum_{x \in \mathcal{S}} Q(x) f(x) + T \times \sum_{x \in \mathcal{S}} Q(x) \log Q(x), && (7) \\ K_{Q,P} &= \log Z + \frac{1}{T}(E - TS) && (8) \end{aligned}$$

F is then minimized through a stochastic discrete time dynamical system of the form

$$\frac{d\theta}{dt} + \frac{\partial F}{\partial \theta} = 0$$

where θ is a given parameter of the model probability distribution. Under this discrete formulation, the notion of a population disappears from the resulting algorithm; a model parameter θ is updated in an "on-line" fashion based on a single sample (and its cost function value) from the model.

Berny's update rule for free energy minimization performs stochastic gradient descent in the continuous parameter space of $Q(\mathbf{x})$. Two specific forms for $Q(\mathbf{x})$ are then considered: a Bernoulli distribution (suitable for binary search spaces) and the log of a multivariate Gaussian distribution. Because the work was concerned with binary search spaces, samples drawn from $Q(\mathbf{x})$ need to be discretized (i.e. for each bit, $x_i \rightarrow x'_i$ where $x'_i = 1$ if $x_i > 0.5$ and $x'_i = 0$ otherwise). When optimizing in continuous space this is not of concern and samples from $Q(\mathbf{x})$ can be used directly.

In the above we have considered only the derivation of an update rule for the mean vector of a Gaussian distribution. Berny's update rule for the (log) mean vector is

$$\Delta\mu = -\alpha(x - \mu)(f(x) + T(1 + \log Q(\mathbf{x}))). \tag{9}$$

where α is a learning rate parameter. Comparing (9) with (5), it can be seen that our update rule is very similar, except that the $T(1 + \log Q(\mathbf{x}))$ term disappears in our derivation and our update is based on a set of samples from the model.

Berny goes on to derive an update rule for the full covariance matrix of a multivariate Gaussian model $Q(\mathbf{x})$. He also points out a number of practical difficulties associated with the implementation of adapting the covariance matrix in an algorithm.

Despite these details, we agree that the fundamental difference between an algorithm based on stochastic gradient descent of the KL divergence and existing EDAs is that the objective function appears explicitly in the update rule, rather than being made implicit via selection (Berny, 2001). The potential advantages of this approach are discussed below.

2.2 PBIL and the $(1, \lambda)$-Evolution Strategy as an approximation of Gradient Descent on $K_{Q,P}$

When viewed as a probabilistic modelling algorithm (Gallagher, 2000), the $(1, \lambda)$-ES generates a population of λ candidate solutions at time t, and updates the mean, μ of the Gaussian model (with constant variance v) to the single population point with the best (here maximum) corresponding objective function value

$$Q^t(x) = \mathcal{N}(\mu^{t-1}, v), \quad \mu^{t-1} = \text{argmax}(\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)\}) \tag{10}$$

Similarly, continuous-PBIL updates μ_t based on the learning rule

$$\mu^t = (1-\alpha)\mu^{t-1} + \alpha \operatorname{argmax}(\{f(\mathbf{x}_1), \dots, f(\mathbf{x}_k)\}) \tag{11}$$

Comparing (10) and (11) with (5) above, PBIL and the $(1, \lambda)$-ES can be seen as performing a crude approximation of the gradient descent of $K_{Q,P}$ described in the framework above. The distinguishing feature of the $\mathrm{P_{KLD}}$ update rule is the absence of a selection operator[3]. Instead of utilizing all points in the current population (weighted by their objective function values), only the best point in the current population is selected (ignoring the single corresponding f value). This approximation may be satisfactory when the objective function value of the best sample in the current population is significantly greater than those of the rest of the population (an outlier in that sample of f values), but there is no reason to expect that this would be likely in practice. In the case of PBIL (11) the effect of this is offset by updating μ^t using an averaging trace of values at previous time steps. An extension to the PBIL learning rule that has been used is to update the model based on a subset of the population, i.e. to move the mean vector in the direction of the average of the best few samples and in the negative direction of the average of the worst few samples in the current population (Baluja, 1997; Sebag and Ducoulombier, 1998). Utilizing the best few samples is in fact a better approximation to the stochastic gradient descent of $K_{Q,P}$ as described above. It has also been shown that moving away from the worst few samples is also an improvement to this same approximation (**?**). Note that in this case it must be assumed that the mean of the objective function values is zero (as in Equation 5 above) and that negative values are exponentiated before being used in the update. In any case, most samples are still disregarded in this type of scheme. It is worth noting that in implementation, when more than one of the best and/or worst samples are used, the population must be at least partially ordered according to f values, to determine these best/worst subsets. This is an additional computational cost which may be significant for large population sizes[4]. Finally, in this version of PBIL the f values corresponding to the subset of samples used to update the model are not used: the selected samples are unweighted in the learning rule summation. An alternative way to think about this is that the real-valued function $f(\mathbf{x})$ is arbitrarily thresholded to a binary value, with the number of ones being the number of "best" sample points utilized.

3 Comparison with Existing Methods and Mean Shift Clustering

In this section, we relate $\mathrm{P_{KLD}}$ and the framework developed in Section 2 to a framework for clustering algorithms known as Generalized Mean Shift (Fukunaga and Hostetler, 1975; Cheng, 1995). In the simplest case, the mean shift algorithm clusters a set of data by (simultaneously) moving each point towards the sample mean $\bar{\mathbf{x}} = \frac{1}{k}\sum \mathbf{x}_i$ of the data set

$$\mathbf{x}_i \propto \mathbf{x}_i - \bar{\mathbf{x}} \tag{12}$$

Generalizing this idea, the nature of the clustering can be varied through the use of a kernel function over points in the data set. This allows the movement of points to be dependent on a subset of the data, or to cause points to be weighted in the calculation of the sample mean. For example, a "flat" kernel of radius r calculates the mean value of points that are less than or equal to a distance r away from the given

[3]Thereby in some sense removing the final "genetic operator from the genetic algorithm" (Baluja, 1994).

[4]Another alternative is to consider a different selection scheme, e.g. Boltzmann or fitness proportional selection.

sample point to be shifted; points outside this radius do not affect the shift of the sample point. Another example is a Gaussian kernel, that weights points in the sample mean calculation according to a Gaussian with mean at the given sample point to be shifted. Intuitively, these kernels implement clustering which is localized. The mean shift framework also allows for the contribution of data points in the sample mean calculation to be weighted by some function, $w(\mathbf{x})$. Finally, the framework considers the case where the data is partitioned into two subsets, $\mathcal{P}$ and $\mathcal{T}$, where each element of $\mathcal{T}$ is mean-shifted based on the basis of $\mathcal{P}$ which itself remains fixed. If $\mathcal{T}$ is $\mathcal{P}$, the mean shift algorithm is considered a blurring process of the data. The well-known k-means clustering algorithm can be seen as a limiting case of the mean shift algorithm (Cheng, 1995).

Hence, generalized mean shift moves a data point $\mathbf{x}$ to a new location, $\mathbf{x}'$, using

$$\mathbf{x} \leftarrow \mathbf{x}'; \quad \mathbf{x}' = \frac{\sum_{\mathbf{x}_i \in X} K(\mathbf{x}_i - \mathbf{x}) w(\mathbf{x}_i) \mathbf{x}_i}{\sum_{\mathbf{x}_i \in X} K(\mathbf{x}_i - \mathbf{x}) w(\mathbf{x}_i)} \tag{13}$$

where K is the kernel function, the $\mathbf{x}_i$ are elements of the data set $\mathcal{P}$ and w is the weighting function. The algorithm shifts points simultaneously in an iterative procedure.

In the case of a Gaussian kernel function, it can be shown (Cheng, 1995) that the mean shift algorithm of (13) moves in the gradient direction of the density estimate at $\mathbf{x}$ given by

$$\sum_{\mathbf{x}_i \in X} K(\mathbf{x}_i - \mathbf{x}) w(\mathbf{x}_i) \tag{14}$$

This confirms the intuition that clustering a data set, by simultaneously adjusting points towards the sample mean of the Gaussian kernel density estimator defined over that data set, will move the data towards local modes of the density estimate.

It is possible to closely relate generalized mean shift to the framework developed in Section 2. In fact, Cheng briefly considers a global optimization algorithm based on mean shift, which involves using the objective function as the weighting function in into (14), i.e. $w(\mathbf{x}_i) = f(\mathbf{x}_i)$ (Cheng, 1995)[5]. Assuming an approximately uniform distribution of the points in $\mathcal{P}$ (with, for example, $\mathcal{T} = \mathcal{P}$ initially), the algorithm will find multiple local optima based on the given (weighted) kernel density estimator. Now, compare this idea with the framework of Section 2 (Equation 5). Initially, ignore the f term in the update rule. While there is no kernel function in our framework, the population at each iteration is generated from a Gaussian centered at the current model (mean) value. The density of points in the population will reflect this Gaussian distribution, which is approximately (given a finite number of samples) the same as weighting a uniform distribution of points by a Gaussian kernel function.

Next, consider the model mean vector μ as a (singular) data set $\mathcal{T}$, and the population generated from $Q(\mathbf{x})$ at a given iteration of the algorithm as the data set $\mathcal{P}$. It is evident that Equation 14 will, in a single iteration, move $Q(\mathbf{x})$ towards the sample mean of $\mathcal{P}$, which is equivalent to the mean shift procedure. However, because our $\mathrm{P_{KLD}}$ algorithm generates a new population at each iteration (that itself depends on the updated $\mathcal{T}$), the generalized mean shift framework cannot describe the algorithm over a sequence of iterations. This reasoning is not changed by including the objective function weighting term in the update rule for both procedures. In our approach, the model used to approximate the unknown distribution is explicit, and the unknown distribution is explicitly a function of $f(\mathbf{x})$. The basis of mean shift is a kernel density

[5]To our knowledge, no further investigations or evaluation of this algorithm have been carried out.

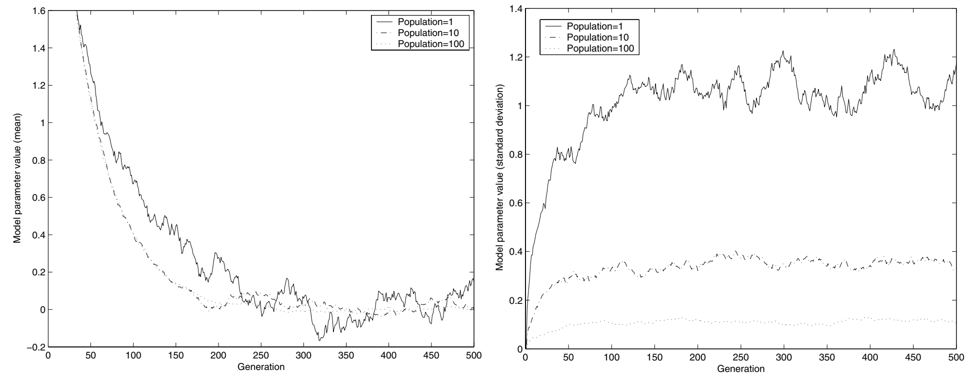

Figure 1: The mean (left) and standard deviation (right) of the evolution of the learning curve for the model parameter, μ on the sphere function. Curves are for the different population sizes indicated.

estimator, and its application to optimization is achieved implicitly via the weighting of component kernels.

4 Experimental Results

In this section we explore the performance of the proposed gradient descent method, $\mathrm{P}_{\mathrm{KLD}}$ and compare it to the standard continuous-PBIL algorithm in some simple problem settings. These experiments are not intended to provide a comprehensive empirical evaluation of the problem-solving performance of $\mathrm{P}_{\mathrm{KLD}}$. Rather, since the contribution of the paper is mainly theoretical, we intend to gain some understanding of the dynamics of the algorithm in practice.

Firstly, consider a "flat" objective function, $f(x) = c$ for some constant value c (e.g. $c = 1$). Ignoring the $\hat{f}(x_i)$ term and learning rate α, and remembering that the samples x_i are generated from a Gaussian with mean μ, Equation 5 will implement a simple random walk with Gaussian distributed stepsizes. This component of the equation will result in a relatively noisy trajectory about the mean value, and may wander an arbitrary distance from the mean value, but will on average be a distance $\sqrt{\sigma}$ from the mean. The $\hat{f}(x_i)$ term in Equation 5 will bias the stochastic process towards higher objective function values, while the constant α will scale down the fluctuations of the update values.

The parabolic "sphere" function, $f(x) = x^2$ is often used in studies of ESs, partly because it is a smooth, unimodal function and as such is useful for studying the basic dynamics of a stochastic algorithm. We used a 1-D sphere function to study the influence of the two parameters of $\mathrm{P}_{\mathrm{KLD}}$, the learning rate, α and the population size, μ on the progress of the algorithm in terms of the updating of the model parameter, μ. Figure 1 summarizes the convergence of μ over 500 generations for population sizes of 1, 10 and 100. The learning rate in each case was fixed at $\alpha = 0.1$. Figure 1 left and right show the mean and standard deviation respectively, over 100 separate runs of the algorithm from a fixed starting point ($\mu = 3.0$). As expected, an increase in population size leads to higher stability and lower fluctuations in these learning curves, as the estimate of the gradient of the Kullback-Liebler divergence improves (see Section 2). Figure 2 shows a similar pair of curves for different values of learning rate ($\alpha = 0.01, 0.1, 1.0$)

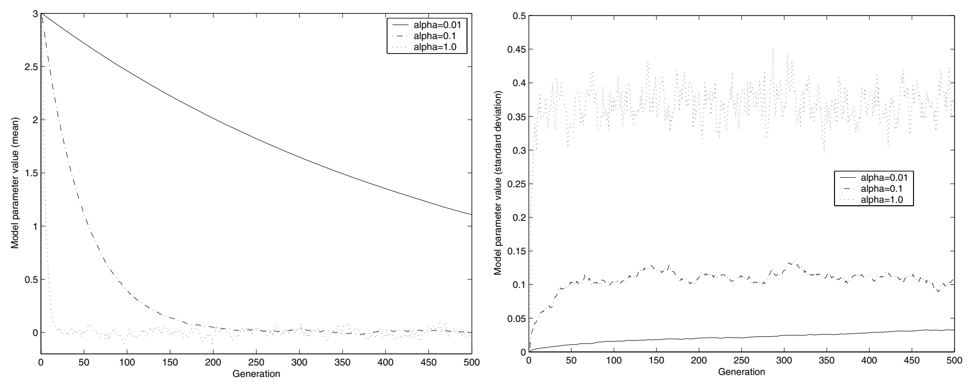

Figure 2: The mean (left) and standard deviation (right) of the evolution of the learning curve for the model parameter, α on the sphere function. Curves are for the different learning rate values indicated.

with a population size of 100. Considering that the Kullback-Liebler divergence for this simple function will be a smooth, unimodal function of μ, the results of these experiments can be understood in terms of a stochastic gradient descent on a unimodal surface with finite step size (where here, population size controls the quality of the stochastic approximation and α controls the step size). Hence, increasing the value of α leads to oscillatory behaviour about the optimum point, and there will be some critical value beyond which the descent will diverge (see, e.g.(Hertz et al., 1991)).

The final set of experiments considered the performance of $\mathrm{P}_{\mathrm{KLD}}$ on a non-smooth function, and compares its dynamics with the standard continuous-PBIL algorithm. Figure 3 (left) illustrates the test functions used, based on an inverted parabolic bowl (f_1) and corrupting this function with different levels of Gaussian (white) noise ($\sigma = 1.0$ for f_2 and $\sigma = 4.0$ for f_3). Note that for f_2 and f_3, the noise is additive, so the actual value returned by the objective function changes with successive evaluations. Intuitively, increasing the variance of the noise corrupting f leads to a more difficult optimization problem. The evolution of the mean of the Gaussian model distribution tends towards the optimum point of the objective function. At the optimum, sampling from the distribution will yield the best possible points (meaning those with the highest objective function values), given its fixed form and variance.

For each algorithm, the model (mean value) was again initialized to 3.0 and the population size was set to 10. Trial and error was used to identify values for the adjustable parameters that gave roughly the same average convergence behaviour on f_1: for both algorithms the variance of the Gaussian model v was set to 1.0, while for PBIL, $\alpha = 0.05$ and for $\mathrm{P}_{\mathrm{KLD}}$, $\alpha = 0.01$.

Figure 3 (right) shows the result of running the algorithms on f_1, in terms of the evolution of the mean vector of the Gaussian model. The mean and standard deviation over 20 trials are shown. It is clear that the convergence of the mean value is very smooth for PBIL compared to $\mathrm{P}_{\mathrm{KLD}}$ ("KLD" in the figures), where the random walk-like behaviour leads to a noisier and more variable convergence. Both algorithms are able to approach the optimal mean value, 0, within 100-200 generations. Once found, the mean value remains close to the optimal value under both algorithms, but $\mathrm{P}_{\mathrm{KLD}}$ is much more irregular. In hindsight, PBIL seems to be a better algorithm to optimize a smooth, unimodal function such as f_1.

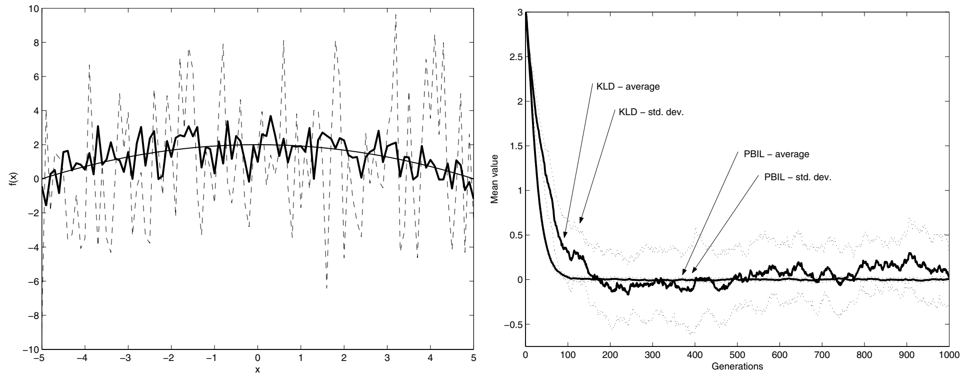

Figure 3: Left: Three test functions used in the simulations. f_1: An inverted parabolic bowl (thin line); f_2 (thick line): f_1 with added Gaussian noise ($\sigma = 1$); f_3 (dashed line): f_1 with added Gaussian noise ($\sigma = 4$). Right: Evolution of the mean parameter for the $\mathrm{P_{KLD}}$ and PBIL algorithms, on the smooth bowl function (f_1). Curves shown are the average (solid lines) and upper and lower standard deviation (dashed lines) of results for each algorithm, over 20 trials.

In the next two experiments, all parameter values are held fixed, but the objective functions f_2 and f_3 are used. Results are shown in Figures 4 (left) and 4 (right) respectively. Comparing these results with Figure 3 (right), two main trends are evident. Firstly, the presence of noise in the objective function has a dramatic impact on the convergence of PBIL. The mean vector takes much longer to approach the optimal value, and its convergence is less stable and more stochastic (mean and standard deviation curves). In contrast, $\mathrm{P_{KLD}}$ is largely unaffected by the presence of noise in the objective function, approaching the optimal mean value, on average, in a similar amount of time for f_1, f_2 and f_3. For the very noisy function f_3, the mean and standard deviation curves for $\mathrm{P_{KLD}}$ show some deterioration, but the algorithm clearly provides faster convergence on f_3 than PBIL.

5 Conclusion

This paper has developed a theoretical framework to interpret the dynamics of continuous evolutionary algorithms based on probabilistic modelling. Under this framework, the basic continuous-PBIL algorithm and the $(1, \lambda)$-ES approximate a stochastic gradient descent of the Kullback-Leibler divergence between the chosen probability model and an assumed target density. The framework also leads to an alternative algorithm assuming a simple Gaussian probability model, which uses all of the information obtained during search (i.e. the entire population and its associated objective function values) to update the probability model. We have shown that the framework complements an existing theoretical framework developed for discrete model-based EAs, and also with the generalised mean shift framework developed in the context of clustering algorithms. The experimental results presented illustrate the stability and fluctuations of the algorithm on a simple problem, and indicate that the proposed algorithm has a robustness to noise in the objective function, and has a more stochastic behaviour than PBIL.

There are several interesting possibilities for future work using the ideas presented in this paper. A comprehensive empirical study of the $\mathrm{P_{KLD}}$ algorithm is needed to

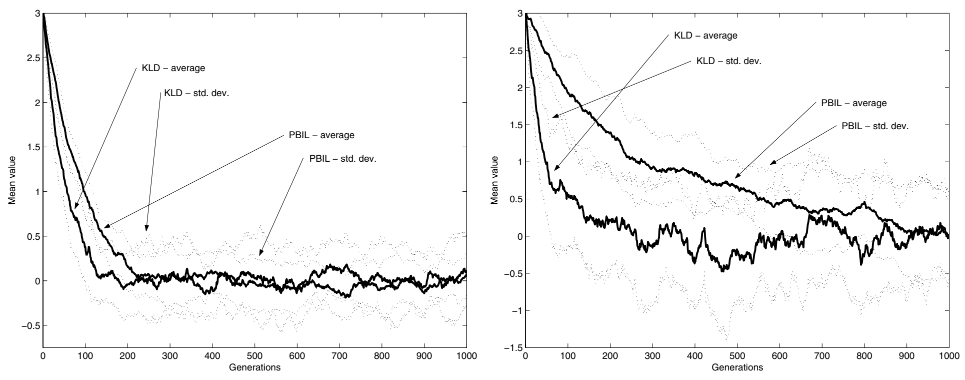

Figure 4: Left: Evolution of the mean parameter for the $\mathrm{P_{KLD}}$ and PBIL algorithms, on the bowl function with moderate noise (f_2). Curves shown are the average (solid lines) and upper and lower standard deviation (dashed lines) of results for each algorithm, over 20 trials. Right: Evolution of the mean parameter for the $\mathrm{P_{KLD}}$ and PBIL algorithms, on the bowl function with high noise (f_3). Curves shown are the average (solid lines) and upper and lower standard deviation (dashed lines) of results for each algorithm, over 20 trials.

investigate its performance at solving more challenging problems (including multi-dimensional and real-world problems), as well as investigating the behaviour of the method across a range of values of its adjustable parameters. From a more theoretical perspective, we are interested in exploring the application of the framework to more flexible probability models. The relationship to mean shift clustering also deserves further investigation, particularly in the context of existing EDAs based on kernel and Gaussian mixture model density estimators (Bosman and Thierens, 1999; Bosman and Thierens, 2000; Gallagher et al., 1999; Gallagher, 2000).

References

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, S. (1997). Genetic algorithms and explicit search statistics. In Mozer, M., Jordan, M., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9, pages 319–325, Cambridge, MA. The MIT Press.

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.

Berny, A. (2000a). An adaptive scheme for real function optimization acting as a selection operator. In Yao, X. and Fogel, D., editors, *First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 140–149. IEEE.

Berny, A. (2000b). Selection and reinforcement learning for combinatorial optimization. In et al., M. S., editor, *Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 601–610. Springer Verlag.

Berny, A. (2001). Statistical machine learning and combinatorial optimization. In Kallel, L., Naudts, B., and Rogers, A., editors, *Theoretical Aspects of Evolutionary Computation*, pages 287–306. Springer Verlag.

Bosman, P. A. N. and Thierens, D. (1999). An algorithmic framework for density estimation based evolutionary algorithms. Technical Report UU-CS-1999-46, Department of Computer Science, Utrecht University.

Bosman, P. A. N. and Thierens, D. (2000). Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In *Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 767–776.

Boyan, J., Buntine, W., and (eds.), A. J. (2000). Statistical machine learning for large-scale optimization. *Neural Computing Surveys*, 3:1–58.

Cheng, Y. (1995). Mean shift, mode seeking and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.

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.

Etxebrria, R. and Larrañaga, P. (1999). Global optimization with Bayesian networks. In *2nd Symposium on Artificial Intelligence/CIMAF99*, pages 332–339.

Fogel, D. B. (1995). *Evolutionary computation : toward a new philosophy of machine intelligence*. IEEE Press, New York.

Fukunaga, K. and Hostetler, L. D. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, IT-21(1):32–40.

Gallagher, M. (2000). *Multi-layer perceptron error surfaces: visualization, structure and modelling*. PhD thesis, Dept. Computer Science and Electrical Engineering, University of Queensland.

Gallagher, M., Frean, M., and Downs, T. (1999). Real-valued evolutionary optimization using a flexible probability density estimator. In Banzhaf, W. and et al., editors, *Proc. Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 840–846, San Francisco, CA. Morgan Kaufmann.

González, C., Lozano, J. A., and Larrañaga, P. (2000). Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479.

González, C., Lozano, J. A., and Larrañaga, P. (2002). Mathematical modelling of discrete estimation of distribution algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter 6, pages 147–163. Kluwer.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA.

Larrañaga, P. (2002). A review on estimation of distribution algorithms. In Larrañaga, P. and Lozano, J. A., editors, *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter 3, pages 57–100. Kluwer.

Larrañaga, P. and Lozano, J. A., editors (2002). *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Kluwer.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In Banzhaf, W. and et al., editors, *Proc. Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 525–532, San Francisco, CA. Morgan Kaufmann.

Pelikan, M., Goldberg, D. E., and Lobo, F. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20.

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

Sebag, M. and Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. In Eiben, A. and et al., editors, *Parallel Problem Solving from Nature - PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 418–427, Springer Verlag.

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