# The Sensitivity of PBIL to the Learning Rate, and How Detailed Balance Can Remove It

**J. L. Shapiro**
Department of Computer Science
University of Manchester
Manchester, M13 9PL U.K.
jls@cs.man.ac.uk

## Abstract

The size of the learning rate in Population-based Incremental Learning (PBIL) is shown to be critical to its performance. As the system size increases, the learning rate must vanish in a problem-dependent way. For the needle-in-a-haystack problem, the learning rate must be exponentially small in the string length. For one-max, the learning rate must be smaller than a constant times the square-root of the string length. These results are shown theoretically and in simulations. This non-robust feature of PBIL can be removed in a principled way by modifying the dynamics so that it obeys detailed balance. When this is done, it is found in simulations that the performance of PBIL is largely independent of the learning rate, and the algorithm finds optima much faster than standard PBIL because a higher learning rate can be used.

## 1 Introduction

Estimation of distribution algorithms (EDAs) are a class of search algorithms similar to genetic algorithms. Instead of evolving a population, they evolve the parameters of a parameterized probability distribution from which a population can be sampled. The basic steps are: generate a population by sampling from the probability distribution; perform selection on that population to generate a new, better population; then use the new population to update the parameters of the probability distribution to make the selected population more likely to be sampled in the next iteration. By repeating these steps, it is hoped that the parameters of the probability distribution evolve to values which generate good solutions with high probability.

The simplest example of an EDA is Population-based Incremental Learning (PBIL) which was introduced by Baluja (1994). (Mühlenbein (1997) developed a similar approach called UMDA.) PBIL uses a probability model which is a product of independent probabilities for each component in the string. Baluja (1996) found PBIL to be more effective than GAs and various hill-climbing algorithms on a range of problems, and it has been used since in a variety of applications. More recently, there has been a large amount of work extending and applying EDAs. Of particular interest has been the development of EDAs based on increasingly complex probabilistic models, mostly in the form of graphical models. Examples of graphical models used include: chains of pairwise dependencies (MIMIC), trees of pairwise dependencies (COMIT), and other more complex relationships. For reviews on EDAs, see Larrañaga and Lozano (2001) and Pelikan et al. (1999).

During search, an EDA algorithm tries to learn the appropriate parameters and properties of the probability model. This learning acts somewhat like selection in genetic algorithms. If the learning rate is very low, the probability model will be changed only slightly between updates. If the learning rate is large, there will be large changes in the probability model between updates. Thus, the learning rate plays a similar role as the selection strength in genetic algorithms.

In this paper, the effect of the learning rate for PBIL is considered. It is well-known that PBIL requires a very low learning rate, or a slow selection method, and although researchers have not explicitly commented on this, most applications of the method either use a very small value of the learning parameter, or a modified selection method. In fact, in a recent paper (Johnson and Shapiro, 2001) it was argued that the reported differences between the performance of PBIL and MIMIC could be explained solely in terms of the differences in the selection methods for the problems studied. When PBIL was modified to use a selection method which had a comparably slow learning rate as that used in MIMIC, much of the differences in performance vanished.

Herein it is shown why a slow learning rate is required. In PBIL, the learning rate is controlled by a single parameter, $\alpha$. It is shown that this parameter must be vanishingly small as the system size increases for PBIL to find good solutions. The rate at which it must vanish is a problem-dependent function of the number of variables, and the required rate can exponentially small. Thus, the algorithm is not robust; it only works if the learning parameter is very carefully chosen. It also makes it very slow, as the rate of the algorithm depends on the learning rate.

It is also shown how this non-robust behavior can be removed in a principled way, by insuring that the dynamics obey a condition called “detailed balance”. This is a reversibility condition widely used in the Markov Chain Monte Carlo method to insure that a Markov chain converges to a given probability distribution. The condition is used here to insure that the PBIL learning equations generate a uniform distribution in the parameters when the search space is flat (all candidate solutions are equally good). This makes the algorithm largely insensitive to the learning rate, and can improve the run time.

## 2 The Standard PBIL Algorithm

PBIL works on binary strings. Let $\vec{x}$ denote the $L$-component binary strings which are the states of the search space, $x_i$ denotes the $i$th component of $\vec{x}$. PBIL uses a probability model defined by the parameters $\gamma_i(t)$ which are the probabilities that $x_i = 1$ at time $t$. Collectively, the parameters will be described by the $L$-component vector $\vec{\gamma}(t)$.

The algorithm works as follows,

- Initialize $\gamma_i(0) = 1/2$ for all $i$;
- Repeat
  - Generate a population of $N$ strings by sampling from the binomial distribution defined by $\vec{\gamma}(t)$.
  - Find the best string in the population $\vec{x^*}$.
  - Update the parameters $\gamma_i(t+1) = \gamma_i(t) + \alpha[x_i^* - \gamma_i(t)]$ for all $i$.
- until (stopping criterion met)

The algorithm has only two parameters, the size of the population $N$ and the learning parameter $\alpha$. The update equation moves the parameters towards the best member of the population $\vec{x^*}$ to make strings similar to it more likely to be sampled in the next iteration. Possible stopping criteria include: the number of updates exceeds a given threshold, a string of a sufficiently high quality has been produced, or the parameters have become sufficiently close to 0 or 1.

A theoretical study of PBIL was carried out by Höhfeld and Rudolph (1997). They showed that for ones-counting problems and similar problems, the mean of the parameters of PBIL converges to the global optimum. From this they concluded that PBIL converges to the global optimum with probability 1 on these problems, and is therefore an effective algorithm when compared to crossover dominated genetic algorithms for which such convergent results are not known. This result is misleading, however, as will be shown in section 3.3. We shall see that for many values of the learning rate, the algorithm will almost never converge to a good solution, although the mean may converge. PBIL is a problem in which the convergence of the mean reveals little, it is the variance that holds the key.

The problem was anticipated by Gonzales et al. (2001). They showed that if some of the initial parameters are sufficiently close to 0 or 1, the global optimum may never be found. The method is to find the probability that a particular component of the optimal value is never sampled. It is easy to show that this probability is finite, and if the initial parameter is far enough from the optimal value, the likelihood of that happening can be arbitrarily close to 1. Their result was in the restricted case of $L = 2$ and $N = 2$, but it is easy to generalize to more general cases.

In another paper (Gonzalez et al., 2002), these authors also anticipated the role of the size of $\alpha$. They show for one-max that as $\alpha$ goes to 0, the trajectory of PBIL converges to the expected mean trajectory. From this they conclude that PBIL converges to the global optimum for sufficiently small $\alpha$.

## 3 How the Learning Rate $\alpha$ Controls PBIL Performance

### 3.1 PBIL on a flat landscape

The problem with PBIL can be easily understood by considering how it would perform on a flat landscape. In this case, all vectors are equally fit, so the "best" vector $\vec{x^*}$ is a random vector and its expected value is

$$\langle x_i^* \rangle = \gamma_i. \tag{1}$$

(The angled brackets $\langle \cdot \rangle$ denote the expectation operator) From this it follows that

$$\langle \gamma_i(t+1) \rangle = \gamma_i(t), \tag{2}$$

and the parameters remain unchanged on average.

One should not conclude, however, that the system actually goes nowhere. It converges rapidly to one of the corners of the hypercube, it is simply unbiased with regard to which one. The reason for this is simple. As the parameters deviate from 1/2 they will move towards a corner of the hypercube. Then the population generated will be biased towards that corner, which will move the parameters closer yet to that corner. For example, if $\gamma_i$ is slightly bigger than 1/2, it is more likely that $x_i^*$ will be 1, moving it closer to 1, making it even more likely that $x_i^*$ will be 1, ad infinitum. All of the corners of the hypercube are attractors which, although never reached, are increasingly attractive with increasing proximity.

As another way to see this, consider the average distance between the parameters and 1/2,

$$D(t) \equiv \frac{1}{L} \sum_i \left( \frac{1}{2} - \gamma_i(t) \right)^2. \tag{3}$$

When all of the parameters are 1/2, $D(t)$ takes the value 0. At the other extreme, when all of the parameters are 0 or 1, $D(t)$ takes the value 1/4.

Under the dynamics on a flat landscape

$$\begin{aligned} \langle D(t+1) \rangle &= \frac{1}{4} - \frac{1}{L} \sum_i \langle [\gamma_i(1-\alpha) + \alpha x_i^*] \, [1 - \gamma_i(1-\alpha) - \alpha x_i^*] \rangle && (4) \\ &= \langle D(t) \rangle (1-\alpha^2) + \frac{\alpha^2}{4}. && (5) \end{aligned}$$

Thus,

$$\langle D(t) \rangle = \frac{1 - (1-\alpha^2)^t}{4}, \tag{6}$$

which converges to 1/4 with a characteristic time

$$\tau = -1/\log(1-\alpha^2) \approx 1/\alpha^2 \text{ for } \alpha \approx 0. \tag{7}$$

This shows why $\alpha$ must be chosen very small to prevent premature convergence. It also shows why the behavior cannot be understood by considering the expected $\vec{\gamma}$ dynamics. The mean dynamics is static; the mean value of all $\gamma$'s remains 1/2 throughout the run. Convergence is only seen in 2nd order statistics like $D(t)$ which is like a variance (and is directly related to the distribution variance).

### 3.2 PBIL and the needle-in-the haystack problem

As an application of search on a flat space, consider the so-called needle-in-a-haystack problem. Here the fitness of all strings is 0 except for one special string (the "needle") which has a fitness of 1. Assume it is the string of all 1's. It is shown here that PBIL will only find the needle if $\alpha$ is exponentially small, and is inefficient at finding the needle when compared to random search.

Clearly, the algorithm cannot find the needle with an expected time less than $2^{L-1}$, since that is the expected time for exhaustive search and exhaustive search is optimal for this problem. (Actually, the expected lower bound for the search time of PBIL $2^L$, since there is nothing to stop it from revisiting states.) Thus, the parameters must not converge in a time less than $2^{L-1}$ for the needle to be found. From equation (7), a necessary condition for the algorithm finding the needle is $\alpha^2 \leq 2^{-L+1}$.

However, this gives $2^{-L/2}$ as an upper bound for the critical value of $\alpha$ required get convergence to the global optimum. The actual value is closer to $2^{-L}$. To show this, we consider the probability of finding the needle at time $t$, denoted

$$\Omega(t) = \prod_{i=1}^{L} \gamma_i(t). \tag{8}$$

Consider times shorter than $T$ where $T$ is long enough that the needle may be found multiple times, but $\alpha^2 T \to 0$ as $L \to \infty$. It will be shown for small $\alpha$ that when the needle is not found, $\Omega$ decreases by an amount $\alpha^2 L\Omega/2$, whereas when the needle is found, $\Omega$ increases by the amount $\alpha L\Omega$. Since initially, the former happens with probability $1-2^{-L}$ whereas the latter happens with probability $2^{-L}$, $\alpha$ must be less than $2^{-(L-1)}$ for the system to move towards the hypercube corner near the optimum, rather than towards a random corner.

When the needle is not found, the mean of $\Omega(t)$ is invariant, $\langle\Omega(t+1)\rangle = \Omega(t)$. However, earlier discussions show that this is misleading; as the parameters converge towards one of the corners of the hypercube, the probability of finding the needle must change. This is another consequence of the mean not being a revealing statistic. It is because $\Omega$ is not a self-averaging quantity; its mean is affected by exponentially unlikely events which have an exponentially big effect. (For example, the probability of finding the needle in the first step is exponentially small, but were it found, $\Omega$ would increase by the exponentially large factor $(1+\alpha)^L$.)

It is expected that early in the dynamics, $\Omega(t)$ will be an approximately log-normal deviate, since the log of $\Omega$ is the sum of approximately independent quantities. The appropriate measure of the central tendency of a distribution which is log-normal (Limpert et al., 2001), is the exponentiated mean of the log. This will be denoted by $[\![\Omega]\!] \equiv \exp\langle\log\Omega\rangle$. This average is much less sensitive to exponentially rare outliers which have an exponentially large effect. For sharply-peaked distributions, this average will be similar to the mean. The dynamics of $[\![\Omega(t)]\!]$ will be considered for $\alpha^2 t$ small.

The recursion for $\Omega$ expanded to second order in $\alpha$ obeys

$$[\![\Omega(t+1)]\!] = \begin{cases} [\![\Omega(t)]\!]\left[1 - \frac{1}{2}\alpha^2 L\right]; & \text{needle not found} \\ [\![\Omega(t)]\!]\left[1 + \alpha L + \frac{1}{2}\alpha^2 L(L-1)\right]; & \text{needle found.} \end{cases} \tag{9}$$

In these equations, $\gamma_i(t)$ has also been expanded around $1/2$ using the dynamics for the average deviation, equation (6).

When the needle is not found, PBIL is exploring a flat space. Although each $\gamma$ is as likely to increase as decrease, the probability of finding the needle decreases with an extremely high probability, since it is very unlikely for all $\gamma$'s to increase. Equation 9 correctly captures the fact, whereas the mean behaviour does not.

Since the needle will be found with probability $\Omega(t)$ and not found with probability $1-\Omega(t)$, the recursion averages to,

$$[\![\Omega(t+1)]\!] = [\![\Omega(t)]\!] \,(1 - \frac{1}{2}\alpha^2 L) + [\![\Omega(t)]\!]^2 \left[\alpha L - \frac{1}{2}\alpha^2 L(L+1)\right]. \tag{10}$$

The second term actually averages to $[\![\Omega(t)]\!] \langle\Omega(t)\rangle$, but the difference between $\langle\Omega\rangle$ and $[\![\Omega]\!]$ is of order $\alpha$ and so introduces a change to order $\alpha^2$. However, the values of interest for $[\![\Omega]\!]$ are also order $\alpha$, so this correction is unimportant.

Equation (10) has a stable fixed point at 0 and an unstable fixed point at $\alpha/2+O(\alpha^2 L)$. If the initial value of $\Omega(0)$ is less than the unstable fixed point, $\Omega$ will decay to zero. If $\Omega(0)$ is greater than the unstable fixed point, $\Omega$ will grow. Therefor, in order for the probability of finding the needle to increase rather than decay to zero, the initial value of $\Omega$ must be greater than $\alpha/2$. The initial value is $\Omega(0) = 2^{-L}$, so the condition for the likelihood of finding the needle to increase rather than decrease is $\alpha < 2^{-(L-1)}$.

Figures 1 and 2 show simulations of PBIL on the needle-in-a-haystack problem. These confirm the predictions made above. The sensitivity of the performance of the algorithm on $\alpha$ is clear to see. Only if $\alpha$ is smaller than $2^{-L}$ does the algorithm converge to the optimum. Figure 3 shows the time to convergence against $\alpha$ in a log-log plot. Convergence is defined by the condition that all parameters $\gamma_i$ are closer to 0 or 1 than 0.05, at which point the algorithm is stopped. The crossover between the two types of convergence are clearly seen in figure 3. For $\alpha$ large compared to $2^{-L}$, convergence is like that on a flat space with rate $\alpha^2$. For smaller $\alpha$, convergence is proportional to $\alpha$. The algorithm is inefficient because it requires such small $\alpha$; convergence to the optimum scales like $4^L$. This is because the rate of convergence to the optimum goes like $\Omega\alpha$, both of which are $O(2^{-L})$.

### 3.3 PBIL and one-max

Next we consider the so-called one-max problem (any continuous monotonic function of unitation will behave the same). The fitness is

$$u = \frac{1}{L}\sum_i^L x_i, \tag{11}$$

the number of 1's in the string.

The parameters $\vec{\gamma}$ can be decomposed into a component parallel to the optimum and a component perpendicular to that direction. Movement along the perpendicular direction is neutral, it does not change the expected fitness of the sampled strings. Only movement

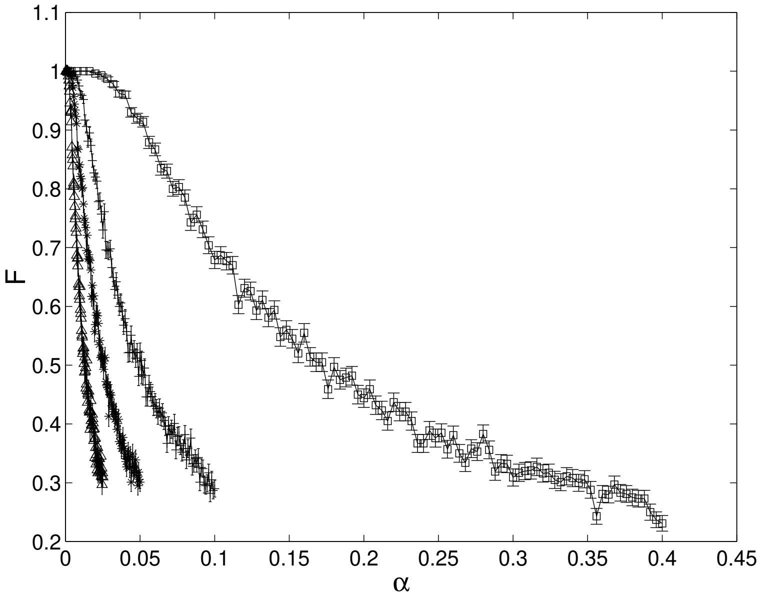


**Figure 1** Simulations on PBIL on needle-in-a-haystack problem for $L = 8, 10, 11, 12$ (respectively $\Box, +, *, \triangle$). The algorithm is run until no parameters are between 0.05 and 0.95, and averaged over 1000 runs. Figure shows the fitness of the best population member at convergence versus $\alpha$. The non-robustness of the algorithm is clear; as $L$ increases, the fitness falls off more quickly with $\alpha$ and $\alpha$ must be very finely set to a very small value to find the optimum.

towards or away from the optimum changes the fitness. Now, random binary strings, like those generated at the start of the algorithm, are almost entirely perpendicular to the global optimum. A random string has expected fitness $1/2$ with fluctuations of size $1/\sqrt{L}$. Thus, initially, movement is in a direction which projects into the neutral subspace an amount $1 - 1/\sqrt{L}$, and in the direction of the optimum an amount $1/\sqrt{L}$.

Thus, the situation is like that of the needle-in-a-haystack problem. The perpendicular direction is flat, so there is convergence towards an arbitrary hypercube corner with a rate,

$$\tau_{\perp} \sim \alpha^2. \tag{12}$$

from equation (7). Movement towards the global optimum occurs at a rate,

$$\tau_{||} \sim \frac{\alpha}{\sqrt{L}}. \tag{13}$$

Thus, $\alpha$ must be small compared to $1/\sqrt{L}$ for movement towards the global optimum to win.

Here is rough argument to show how the fitness in the final population depends on $\alpha$. We make use of the following (see for example Shapiro and Prügel-Bennett (1995)).

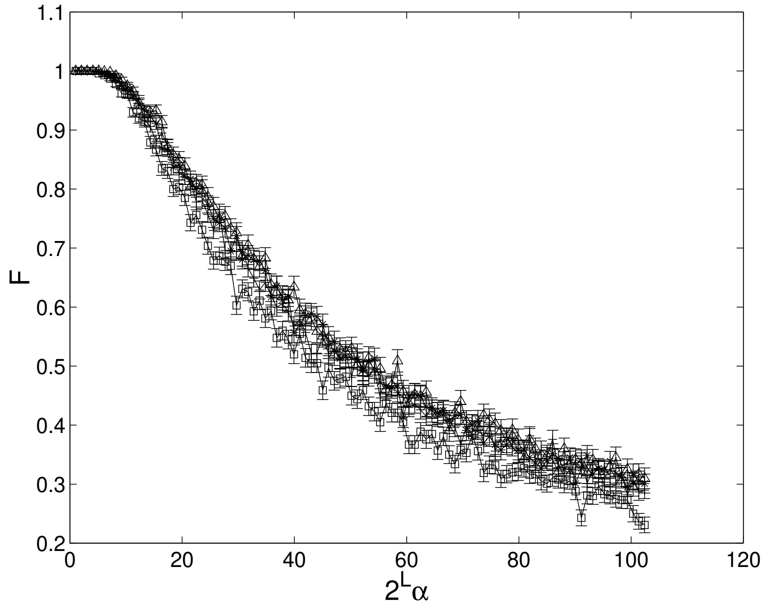


**Figure 2** As previous, but with $\alpha$ scaled by $2^L$. The data approximately collapses, which shows that as $L$ increases, $\alpha$ must decrease like $2^{-L}$ to get the same performance

> If $N$ random variables are drawn from a Gaussian distribution with mean $m$ and variance $\sigma^2$, the expected largest value drawn is $m + \sqrt{2\sigma^2 \log(N)}$ for large $N$.

From this, using the Gaussian approximation to the binomial distribution, the expected best string has fitness value,

$$\langle u^*(t) \rangle = u(t) + \sqrt{2v(t)\log(N)}, \tag{14}$$

where $v(t)$ is the variance in probability distribution,

$$v(t) = \frac{1}{L^2} \sum_i \gamma_i(t)[1 - \gamma_i(t)],$$

$\langle u^*(t) \rangle$ is the expected best fitness, and $u(t)$ is the mean fitness in the currently sampled population. Approximating the expectation of the square root as the square root of the expectation yields,

$$\langle u(t+1) \rangle = \langle u(t) \rangle + \alpha \sqrt{2 \langle v(t) \rangle \log(N)}, \tag{15}$$

the solution to which for infinite time gives the asymptotic value,

$$\langle u(\infty) \rangle = \langle u(0) \rangle + \alpha \sum_{t=0}^{\infty} \sqrt{2 \langle v(t) \rangle \log(N)}. \tag{16}$$

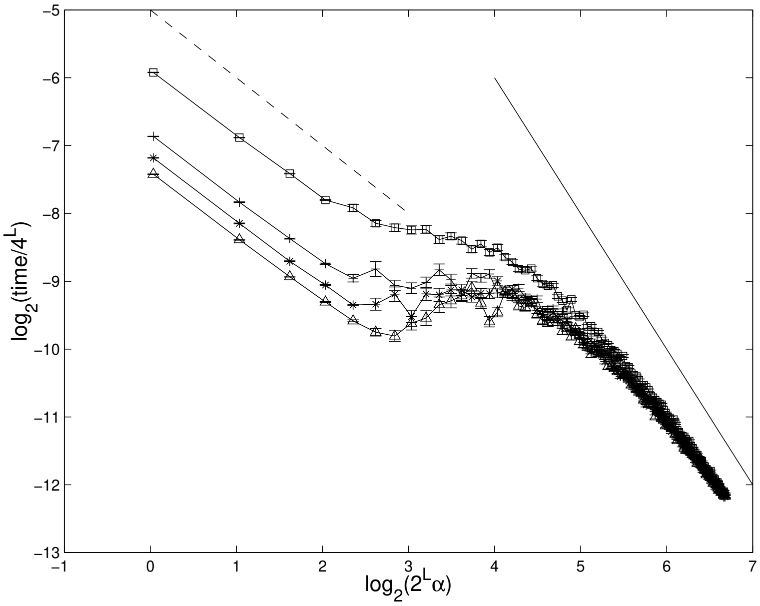


**Figure 3** The convergence time for the needle-in-a-haystack problem, where convergence is when all parameters are greater than 0.95 or less than 0.05. The simulations use the same conditions as the previous figure. The solid line has a slope of $-2$; the dashed line has a slope of $-1$. One clearly sees the crossover in the dynamics from convergence without finding the needle which occurs with rate $\alpha^2$ (slope -2) and convergence to the needle which occurs with rate $\alpha 2^{-L}$ (slope -1). As predicted, the crossover point occurs where $\alpha \propto 2^{-L}$. One also sees the inefficiency of PBIL for this problem; the convergence-time scales as $4^L$.

Assuming that the convergence of the variance is primarily due to the convergence on the flat subspace, similar arguments as those leading to equation (6) show that $\langle v(t)\rangle = \langle v(0)\rangle\,(1-\alpha^2)^t$. Thus,

$$\langle u(\infty)\rangle = \langle u(0)\rangle + \frac{\alpha\sqrt{2\log(N)v(0)}}{1-\sqrt{1-\alpha^2}} \tag{17}$$

$$\approx \frac{1}{2} + \frac{\sqrt{\log(N)}}{\alpha\sqrt{2L}}. \tag{18}$$

This predicts that $\alpha$ must scale like $1/\sqrt{L}$ for large $L$ to get consistent behavior. This equation also predicts that the population size is very unimportant, unless it is increased by an exponentially large (in $L$) factor. This equation must break down when the fitness approaches one, which is where the Gaussian approximation to the binomial breaks down. It also uses an asymptotic value for large $N$, although $N = 20$ has been used throughout.

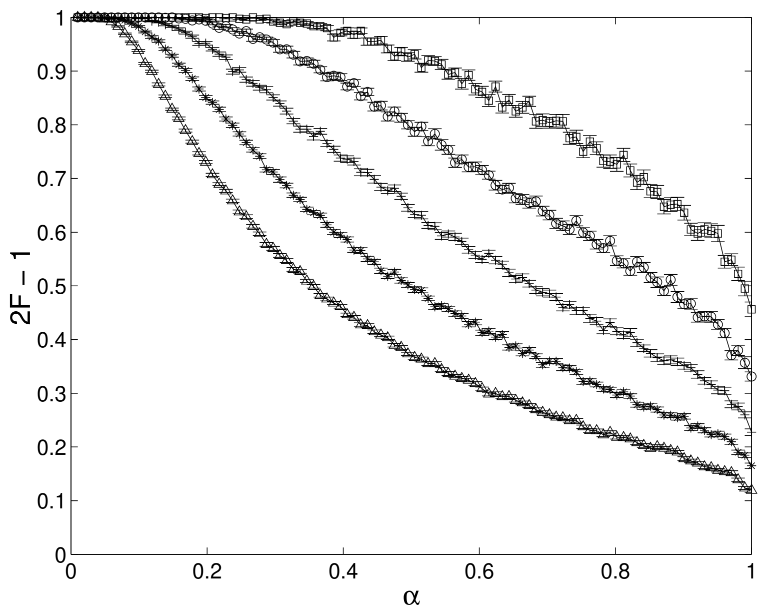


**Figure 4** Simulations on PBIL on the one-max problem for $L = 16, 32, 64, 128, 256$ (respectively $\square, \circ, +, *, \triangle$). The algorithm is run until all parameters are closer to 1 or 0 than 0.05, and averaged over 100 runs. Figure shows the fitness of the best population member at convergence versus $\alpha$. The fitness is scaled so that the global optimum has fitness 1 and the expected fitness of a random string is 0. As $L$ increases, $\alpha$ must be set to a decreasing value to find the optimum.

Simulations of PBIL on one-max confirm these predictions. PBIL fails to converge to the global optimum unless $\alpha$ is small compared to $1/\sqrt{L}$. Figures 4 and 5 shows the scaling of fitness at convergence with $\alpha 2^L$, and compares simulations with equation (18). Figure 6 shows the run times.

For this problem, Höhfeld and Rudolph argued that PBIL converges to the global optimum in the mean. Nonetheless, one can see from figure 4 that for most values of $\alpha$ and on most runs, the optimum is not found. It is plain to see why convergence in the mean does not predict the actual dynamics. Unless $\alpha$ is sufficiently small, most of the dynamics occurs in the subspace orthogonal to the direction towards the global optimum. This subspace is flat and shows no *mean* convergence, because there is no preferred direction to the convergence. The only preferred direction is towards the global optimum. Nonetheless, there is convergence in the orthogonal subspace; to observe this, expected deviations or variances must be studied.

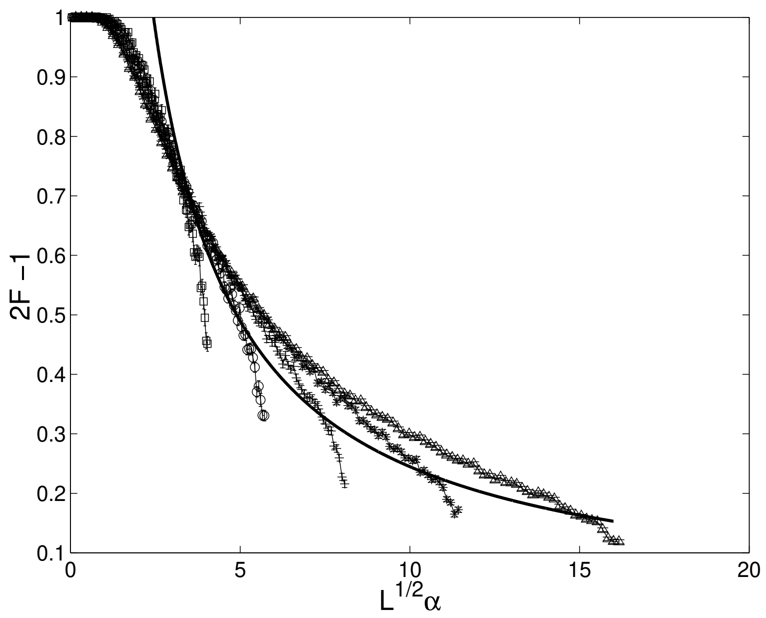


**Figure 5** As previous, but with $\alpha$ scaled by $\sqrt{L}$. The data approximately collapses, which shows that as $L$ increases, $\alpha$ must decrease like $\sqrt{L}$ to get the same performance. The curve shows the approximate theory of equation (18).

# 4 Detailed Balance PBIL

We have seen that PBIL is not robust as conventionally defined. It gets attracted to corners of the hypercube even when there is no evidence from the fitness function to drive it there. To prevent this, $\alpha$ must be made very small. How small $\alpha$ has to be depends on the problem, because the attraction to arbitrary hypercube corners has to be slower than convergence to the corners where good solutions are found. The latter rate is problem dependent. PBIL is also very easy to fix. The remainder of the paper describes a principled way to fix it.

## 4.1 Search algorithms should search flat spaces uniformly

A principle which all search algorithms should satisfy is the following,

> If the search space is flat, the fixed-point distribution of the algorithm should be uniform and independent of the starting conditions.

This might be called, "the principle of leaving no stone unturned". It basically says that the algorithm should search everywhere in an unbiased fashion when the fitness function gives no reason to look at any particular part of search space. Simple hill-climbing algorithms obey this principle, simulated annealing does as well. PBIL does not as we have seen, and

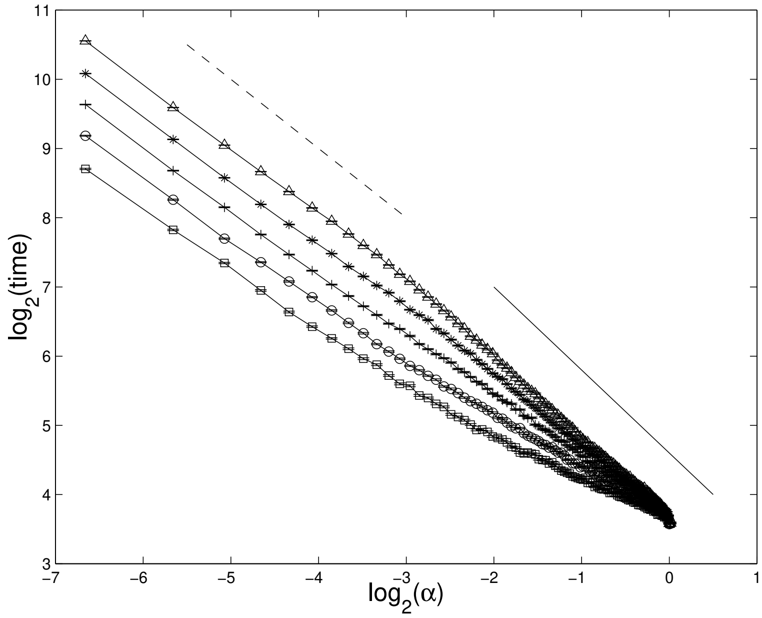


**Figure 6** The convergence time for one-max, where convergence is defined as previously. The dashed (solid) line has a slope of $-1$ $(-2)$. Again the crossover in the dynamics is apparent and occurs where $\alpha \propto 1/\sqrt{L}$ as predicted.

that is the essential problem with it. The region of search space it searches is very much dependent upon the initial steps it took, even on a flat space. Making this principle hold for PBIL points the way for the solution to the problem.

To get any Markov process on variables $x$ to converge to a desired equilibrium distribution $\pi(x)$, it is sufficient to insure that the transition probabilities obey the following condition,

$$w(x'|x)\pi(x) = w(x|x')\pi(x'), \tag{19}$$

where $w(x'|x)$ is the probability of generating $x'$ from $x$. This condition is called *detailed balance*, it is a reversibility condition. (This condition is essential to the Markov chain Monte Carlo method of random sampling. For a review of Markov process and Markov chain Monte Carlo methods, see Neal (1993), especially sections 3.3 and 4.2.) Thus, any search algorithm searching on a flat space should have dynamics which obeys,

$$w(x'|x) = w(x|x'), \tag{20}$$

since on a flat space the probability of being in any state should be the same.

There is a difficulty in modifying the dynamics of PBIL to satisfy detailed balance, however. PBIL visits a set of points which varies from run to run, and (almost) never revisits points. Thus, equation (20) can never be satisfied because if $\vec{\gamma}$ goes to $\vec{\gamma}'$, the dynamics does not take $\vec{\gamma}'$ back to $\vec{\gamma}$. The first modification to PBIL will be to put it on a lattice, which is described in section 4.2. Then the dynamics can be altered to enforce detailed balance, which is described in section 4.3.

### 4.2 PBIL on a lattice

The dynamics of PBIL can be modified to insure that the algorithm always visits a known countably infinite subset of points (a lattice). To do that, we modify the dynamics so that the system can always take a step back to where it was in the previous time-step. One way to do this is to use the standard PBIL update when the parameters move away from 1/2, and its inverse when the parameters move towards 1/2. The dynamics becomes

$$\gamma_i(t+1) = \begin{cases} \gamma_i(t) + \alpha\left[x_i^* - \gamma_i(t)\right], & \gamma_i(t) \text{ and } x_i^* \text{ on the same side of } 1/2, \\ \frac{\gamma_i(t) - \alpha x_i^*}{1-\alpha}, & 1/2 \text{ is between } \gamma_i \text{ and } x_i^*. \end{cases} \tag{21}$$

This modified dynamics will generate parameters that are in slightly different places than those generated by standard PBIL. This is unimportant, since the populations generated will be very similar.

Under this dynamics and assuming that all parameters start at 1/2, the locations of the points on the lattice can also be described by a set of integers $n_i$. The integer $n_i$ is the number of steps taken from 1/2 towards 1 minus the number taken towards 0 for the $i$th component. The relationship between $\gamma_i$ and $n_i$ is.

$$\gamma_i = \begin{cases} 1 - \frac{1}{2}(1-\alpha)^{n_i}, & n_i > 0; \\ \frac{1}{2}(1-\alpha)^{|n_i|}, & n_i < 0; \\ \frac{1}{2}, & n_i = 0. \end{cases} \tag{22}$$

The update equations (21) can be implemented more simply by incrementing and decrementing the $n_i$'s by 1. The change in $\gamma_i$ corresponds in all cases to an increment or decrement of $n_i$. To utilize this simplicity, we will derive the update equations in terms of the $n$'s rather than the $\gamma$'s.

### 4.3 The uniform distribution on the lattice

The goal is to produce a dynamics which produces a uniform distribution when the fitness landscape is flat. The meaning of uniform in this context needs some discussion. We want it to be uniform in $\gamma$ space, not in $n$ space. If the distribution were made uniform in $n$, by making transitions between $n$ and $n+1$ equally likely as transitions from $n+1$ to $n$, the $\gamma$ values would be mostly found close to the corners of the hypercube, since that is where most of the $n$ values are.

We recall how distributions are transformed. If $x$ is transformed to $y = f(x)$, to transform the probability distribution on $x$, we must also transform the measure $dx$. I.e.

$$P(y)dy = P\left[f^{-1}(y)\right]\frac{dx}{dy}dy. \tag{23}$$

From equation (22), we see that

$$d\gamma_i \propto (1-\alpha)^{|n|} dn. \tag{24}$$

Normalizing this yields the distribution in $n$ spacing corresponding to a uniform distribution in $\gamma$ space,

$$P(n) = \frac{\alpha}{2-\alpha}(1-\alpha)^{|n|}. \tag{25}$$

#### 4.3.1 Detailed balance by rejection sampling

One of the easiest methods for sampling from a distribution is to use the rejection method. It is very widely used in Markov chain Monte Carlo method of sampling from a distribution. In this, one has $g(x'|x)$ as a proposal distribution; it is the probability of proposing the value $x'$ from $x$. Then, $A(x'|x)$ is the probability of accepting this change. I.e., having proposed $x'$ from $x$, at time $t$, the value at time $t+1$ is $x'$ with probability $A(x'|x)$ and $x$ with probability $1 - A(x'|x)$. Detailed balance condition becomes

$$g(x'|x)A(x'|x)\pi(x) = g(x|x')A(x|x')\pi(x'). \tag{26}$$

For example, the well-known Metropolis-Hasting algorithm has

$$A(x'|x) = \min\left(1, \frac{\pi(x')g(x|x')}{\pi(x)g(x'|x)}\right). \tag{27}$$

To use rejection sampling to get PBIL to search a flat space uniformly, set $\pi$ to be the uniform distribution defined in equation (25). The proposal probability $g(n'|n)$ is the probability of generating $n'$ from the parameter $n$. On a flat space, the probability of $x^*$ is $\gamma$. Transforming to $n$ space and solving in equation (27), results in these equations for the rejection probability.

**For** $n > 0$ use the following as the acceptance $A(n'|n)$ probability of accepting a change from $n$ to $n'$,

$$A(n+1|n) = \frac{\frac{1}{2}(1-\alpha)^{n+2}}{1-\frac{1}{2}(1-\alpha)^{n}} \tag{28}$$

$$A(n-1|n) = 1. \tag{29}$$

**Likewise, if** $n < 0$ the acceptance probability is

$$A(n-1|n) = \frac{\frac{1}{2}(1-\alpha)^{|n-2|}}{1-\frac{1}{2}(1-\alpha)^{|n|}} \tag{30}$$

$$A(n+1|n) = 1. \tag{31}$$

**For** $n = 0$ accept either move with equal probability.

In applying the acceptance formula, each component is treated independently. Thus, moves can be accepted on some components and not on others.

### 4.3.2 Discussion

The detailed balance algorithm behaves similarly to standard PBIL although it is on a lattice which makes the step-sizes different. It will always accept moves away from hypercube corners. It will sometimes reject moves towards corners of the hypercube and as it gets closer to the corner, the rejection probability increases. Of course, if that corner does contain very fit strings, it will eventually move in that direction, because it does not always reject moves towards a corner. If it is repeatedly generating moves towards a particular corner, it will eventually move there. It is important to point out that the rejection of many moves dues not imply that the algorithm is inefficient. Even if the parameters do not change, the populations generated will be different; the algorithm will still be searching.

### 4.3.3 Results

To test whether Detailed Balance PBIL works well, it is applied to the two problems of the opening sections. For the needle-in-a-haystack, simulations were performed for 100 values of $\alpha$ between 0 and 0.4 equally spaced for $L = 8, 9, 10, 11, 12$; 1000 trials of each, population size 20, with the same convergence criterion as before, simulation halts when all $\gamma_i$'s are less than 0.05 or greater than 0.95. On none of those simulations did the algorithm fail to contain the global optimum in the final population. In other words, the analogous graph to figure 1 is a horizontal line at 1 with zero-sized error bars (which is why it is not plotted).

Figure 7 shows the convergence time of the Detailed Balance PBIL algorithm. This algorithm can be slower than ordinary PBIL for a given value of $\alpha$, but the algorithm is much faster in general because it allows use of a much larger value of $\alpha$. For example, in the figure it can be seen that the convergence time goes like $2^L$ except for very small values of $\alpha$. (The convergence time is measured in number of times a population is generated, not the number of fitness evaluations. The latter is 20 times greater than the former, because population size of 20 has been used throughout. ) The convergence criterion used (all $\gamma_i$'s close to 0 or 1 by a fixed amount) is the same as used to study ordinary PBIL in section 3. It is not an appropriate stopping criterion for Detailed Balance PBIL, since this algorithm is designed to make convergence difficult (even when the optimum has been found many times). It used to make a fair comparison between the two algorithms. It does make the search time longer than would be the case with a stopping criterion more suited to the algorithm.

For the one-max problem, Detailed Balance PBIL does not always find the global optimum, but does find good solutions over a wide range of values of $\alpha$. Figures 8 and 9 show this. The algorithm does always converge to the hypercube corner containing the global optimum, but because the stopping criterion does not require that it get closer than a certain amount, as $L$ increases it is more likely that there are few erroneous bits in every string.

## 5 Conclusions

The learning rate of PBIL has to be very small for the algorithm to work, and unpredictably so, as it depends upon the problem size in a problem dependent way. I have

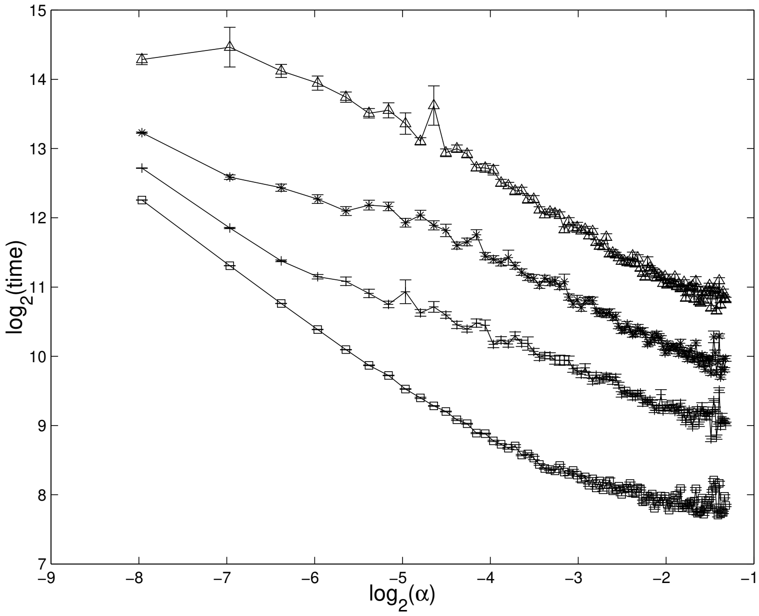


**Figure 7** The convergence time of Detailed Balance PBIL for needle-in-a-haystack with $L = 8, 10, 11, 12$ (respectively $\Box, +, *, \triangle$). All cases converge to the global optimum. Detailed Balance PBIL is much faster than ordinary PBIL because it can use a much larger value of $\alpha$. For large value of $\alpha$, the convergence time is of the order of $2^L$, as can been seen from the figure.

argued why that is by considering flat landscapes, and landscapes with flat subspaces, like one-max and the needle-in-a-haystack. In fact, it is a consequence of a lack of detailed balance. The probability of moving towards the corner of hypercube can be much greater than the probability of moving back.

Detailed balance fixed the problem dramatically in the two cases studied. It is important to see whether a similar picture will hold on more realistic problems, and that work is ongoing.

The phenomenon studied here would hold in other EDAs, because for any type of model, the probability is high of generating a population which reinforces the move just made. On the other hand, more complex models have many more parameters, and also have more sources of variability, so the problems may be less important. It would be interesting to learn if more complex models are more effective because this problem is less acute, or if the complexity is really important to solve the problem. It would also be interesting to see whether detailed balance dynamics can be derived for other EDA models.

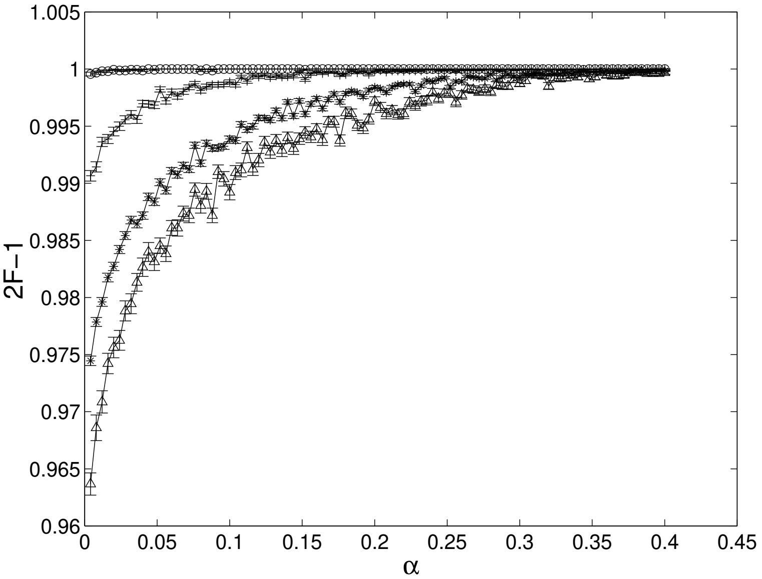


**Figure 8** Final fitness of one-max as a function of $\alpha$. The solutions are all very good, but are not perfect.

## References


Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competive learning. Technical Report CMU-CS-94-163, Computer Science Department, Carnegie Mellon University.

Baluja, S. (1996). Genetic algorithms and explicit search statistics. In *Advances in Neural Information Processing Systems 6*.

Gonzales, C., J. Lozano, and P. Larrañaga (2001). The convergence behaviour of the PBIL algorithm: a preliminary approach. In *Proceedings of ICANNGA 2001*.

Gonzalez, C., J. Lozano, and P. Larrañarraga (2002). Analyzing the PBIL algorithm by means of discrete dynamical system. *Complex Systems*. In press.

Höhfeld, M. and G. Rudolph (1997). Towards a theory of population-based incremental learning. In *Proceedings of the Fourth IEEE Conference on Evolutionary Computation*, Volume 1, pp. 1–5.

Johnson, A. and J. L. Shapiro (2001). The importance of selection mechanisms in distribution estimation algorithms. In *Proceedings of the* $5^{th}$ *International Conference on Artificial Evolution AE01*. `http://www.cs.man.ac.uk/~jls/papers/eapaper.ps.gz`.

Larrañaga, P. and J. A. Lozano (2001). *Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.

Limpert, E., W. A. Stahel, and M. Abbt (2001). Log-normal distributions across the sciences: Keys and clues. *BioScience 51*(5), 341–352.

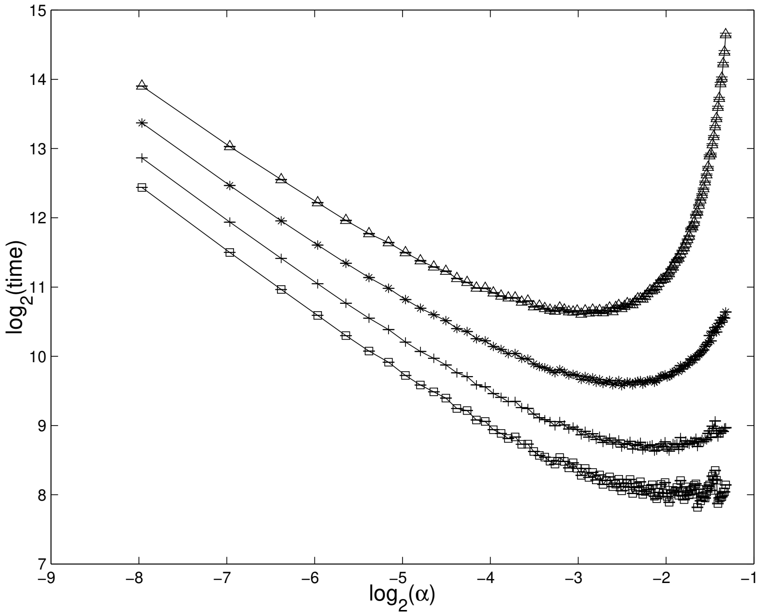


**Figure 9** Convergence time on one-max using Detailed Balance PBIL for $L = 32, 64, 128, 256$ (respectively $\square, +, *, \triangle$). Since a doubling of $L$ corresponds to an approximately unit sized shift in the log scale of the convergence time, the convergence time scales approximately as $L$, which is efficient. There is an optimal value of $\alpha$, but it no longer depends on the size of the system.


Mühlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation 5*(3), 303–346.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto. `http://www.cs.toronto.edu/~radford/review.abstract.html`.

Pelikan, M., D. E. Goldberg, and F. Lobo (1999). A survey of optimization by building and using probabilistic models. Technical report, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Shapiro, J. L. and A. Prügel-Bennett (1995). Maximum entropy analysis of genetic algorithm operators. *Lecture Notes in Computer Science 993*, 14–24.