# Playing in Continuous Spaces: Some Analysis and Extension of Population-Based Incremental Learning

**Bo Yuan Marcus Gallagher**
School of Information Technology and Electrical Engineering
The University of Queensland
QLD 4072, Australia
{boyuan, marcusg}@itee.uq.edu.au

**Abstract - As an alternative to traditional Evolutionary Algorithms (EAs), Population-Based Incremental Learning (PBIL) maintains a probabilistic model of the best individual(s). Originally, PBIL was applied in binary search spaces. Recently, some work has been done to extend it to continuous spaces. In this paper, we review two such extensions of PBIL. An improved version of the PBIL based on Gaussian model is proposed that combines two main features: a new updating rule that takes into account all the individuals and their fitness values and a self-adaptive learning rate parameter. Furthermore, a new continuous PBIL employing a histogram probabilistic model is proposed. Some experiment results are presented that highlight the features of the new algorithms.**

## 1 Introduction

Evolutionary Algorithms (EAs) refer to a broad class of optimization algorithms, which take some inspiration from evolutionary systems in the natural world. Usually, each algorithm begins with a population of randomly generated individuals. Then, the fitness of each individual is calculated based on some performance measure, called the fitness function. Next, a new population of individuals will be generated through some direct manipulation of the current population. Different algorithms employ different operators to do this job. In Genetic Algorithms (GAs), crossover is often regarded as the dominant operator while mutation is used simply for keeping the genetic diversity of the population. As a contrast, Evolution Strategies and Evolutionary Programming only use mutation in general (see [1] for an outline and references).

EAs enjoy several advantages compared to other optimization algorithms. For example, they only require a minimal amount of problem-specific knowledge and do not have any special requirements of the fitness function such as smoothness and differentiability. Furthermore, since EAs often maintain a population of individuals, investigating many areas in parallel, they are less likely to get stuck in local optima and can be implemented efficiently through parallel computation.

In recent years, a new class of EAs has emerged based on probabilistic modelling of the search space[2, 3]. Although these methods are often different from each other in terms of the type of model in use, they are distinct from traditional EAs in that each algorithm employs a probabilistic model of the search space that is the only persistent part of the searching process. Initially, this model is often randomly generated or set to represent general distribution. In each generation, a population of individuals is generated by sampling from this model. After the fitness of each individual is calculated, some (typically the best) individuals are chosen to update the old model. In these algorithms, new individuals are not generated by directly manipulating old ones (e.g., via crossover or mutation). So, the most fundamental difference between these algorithm classes is that traditional EAs work on low-level representations (e.g., genes) of the search space, trying to create better individuals by directly changing the gene values, while model-based EAs employ probabilistic models as the high-level abstraction of the search space and use them to drive the searching process. Such probabilistic models are built on the statistical information contained in selected individuals, which is a kind of estimation of the structure of good individuals.

Population-Based Incremental Learning (PBIL) is one of the earliest model-based EAs, which "removes the genetics from the genetic algorithm"[4, 5]. Originally, PBIL was designed for binary search spaces. It employs a Bernoulli random variable as the model for each bit, which is collected into a real-valued vector. During evolution, each element of the vector is updated towards the best individual(s). Recently, PBIL has been extended to continuous spaces by using an interval approach [6] and a Gaussian distribution model[7].

However, there are still some issues related to PBIL. For example, when updating the probability vector, the fitness value of each selected individual is not taken into account. Instead, all of them are given the same strength, which is questionable. Secondly, the learning rate parameter plays an important role in PBIL. There is always a trade-off between reliability and convergence speed. Finally, the continuous implementations of PBIL seem to have some inherent disadvantages, which sometimes may result in serious performance loss.

The next section briefly reviews PBIL and its extensions for continuous search spaces. A new learning rule incorporating fitness values and a strategy for self-adapting the learning rate are also described. A new continuous PBIL based on histograms is proposed in Section 3. Related experiment results are presented in Section 4. The paper is finished in Section 5 with conclusions and potential work direction in the future.

## 2 Population-Based Incremental Learning

### 2.1 Binary PBIL ($PBIL_B$)

PBIL is one of the simplest model-based EAs, which assumes no dependence among variables. The probabilistic model in use is a real-valued vector with each element independently representing the probability of generating a 1 in each corresponding bit.

**Step 1**: Initialize probability vector P to 0.5, t=0
**Step 2**: Sample a population of individuals X from P
**Step 3**: Evaluate individuals
**Step 4**: Update P by the best individual
$P(t+1) = (1-\alpha) \cdot P(t) + \alpha \cdot X^{Best}$
**Step 5**: t=t +1;
**Step 6**: Go to Step 2 until stopping criteria are met

**Parameters**:
Population Size, Learning Rate (α)

Table 1: The Basic Framework of $PBIL_B$

The basic framework of $PBIL_B$ is in Table 1. $PBIL_B$ starts from a probability vector with all elements set to 0.5, which means that each bit in the individual will be set to 0 or 1 with equal probability. During evolution, the value of each element will be updated by the best individual in the population and move away from 0.5, modifying its estimation about the structure of good individuals. Finally, the algorithm will converge to a vector with each element close to 0 or 1. In practice, two decisions have to be made: the value of the learning rate parameter (α) and the number of individuals used to update the vector.

For the learning rate in $PBIL_B$ , a small value is usually recommended[8]. It controls the trade-off between reliability and convergence speed. In traditional EAs, old individuals are replaced by new ones instantly while $PBIL_B$ uses an incremental learning strategy to modify its model cautiously, maintaining a long-term memory. The reason is that the vector in $PBIL_B$ does not represent a single individual in the search space. Instead, it represents the whole search space with certain bias towards some areas. If the learning rate is too large, this vector will quickly move towards 0 or 1 in each element, reducing the exploration ability of $PBIL_B$ dramatically.

The number of individuals used to update the vector is another parameter of the algorithm. In the above framework, only the best individual is utilized while all others are discarded. Certainly, more individuals can be selected. Another variation is to let $PBIL_B$ move towards the best individual(s) and move *away* from the worst individual at the same time[5].

$PBIL_B$ has been successful in practice on several test functions[9]. However, there are still some open questions. Firstly, we have discussed the necessity of a small learning rate but this may result in a low convergence speed. A possible solution is to use a self-adaptive method to modify its value dynamically. Secondly, when using multiple individuals to update the vector, it may be useful to take account of their fitness values explicitly. This allows different individuals to have different strengths in modifying the probability vector.

In $PBIL_B$ each element in the vector is evolved and sampled independently. However there are often many complex relationships among variables in real-world problems. This means that what value a variable should take in order for the whole individual to have good fitness cannot be decided solely. In fact, it is pretty common that good individuals are quite different from each other, which means that different values of a variable may all produce some good individuals provided that the values of other variables are set accordingly. Under this situation, $PBIL_B$ may oscillate for a while and then converge to one optimum due to some random factors.

This problem is due to the simplified model that $PBIL_B$ uses. In order to overcome this disadvantage, some work has been done towards using more complicated models such as dependency chains, dependency trees or Bayesian networks to explicitly capture the interaction among variables in the hope of allowing the algorithm to concentrate its sampling[2, 3]. In a word, we expect these models to tell the algorithm what value a variable should take given the values of other variables. However, there are some new issues with these models[10]. Firstly, building a model itself can be a time-consuming task. The cost of building a model typically needs to be traded-off with the potential benefit we can expect to get from it. Secondly, there is no formal justification about which model should be used. A complex model can capture more information about the relationship among variables but it may also reduce the exploration ability of the algorithm using this model. Finally, these model-based EAs have not been extensively tested on large-scale problems. For example, in a high-dimensional space, the number of individuals needed to build a reasonable model can be extremely large. As a result, a model built on the information provided by a limited number of points in the search space can be incomplete and misleading.

### 2.2 PBIL Based On Interval ($PBIL_I$)

One of the earliest attempts to extend PBIL to continuous spaces is based on the idea of an interval[6]. For each variable $X_i$, three parameters are specified: the lower boundary $L_i$, the upper boundary $U_i$ and a probability value $P_i$ representing the probability of $X_i$ being greater than the middle of the interval $(L_i + U_i)/2$. Initially, each pair of boundaries is set to be equal to the pair of boundaries of the search space in each dimension. $P$ is set to 0.5 in all dimensions so that each variable generated can be greater or less than the middle of the interval with equal probability. This setting gives $PBIL_I$ the freedom to initially search the whole space uniformly. In each generation, each $P_i$ is updated towards 1 if $X_i$ on the best individual is in the upper interval and vice versa. If the value of a $P_i$ is sufficiently close to 1 or 0, which means that the algorithm is confident that the value of $X_i$ should be greater or less than the middle of the current interval,

one of the two boundaries will be moved accordingly to the middle, shrinking the interval by half.

$PBIL_I$ employs a similar mechanism as $PBIL_B$ in that it also starts from a most general model and gradually concentrates on a smaller part of the whole search space. However, it may be limited in that the movement of boundary is not reversible and the sampling is restricted within the interval. This means that if an incorrect or premature shrinking of interval happens, there is no chance for $PBIL_I$ to correct it and the search space outside the current interval has zero probability of being sampled again.

### 2.3 PBIL Based On Gaussian Distributions ($PBIL_G$)

Another approach to continuous PBIL is somewhat different. It employs a model of a Gaussian distribution on the search space, which is the product of a set of 1D Gaussians for each variable[7]. It also starts with a rather general distribution with the mean vector of its Gaussian in the middle of the search space. In each generation, the mean vector X is updated by the combination of the best, the second best and the worst individuals (Eq.1).

$$X^{t+1} = (1-\alpha) \cdot X^{t} + \alpha \cdot (X^{best,1} + X^{best,2} - X^{worst}) \qquad (1)$$

$PBIL_G$ also introduces a new parameter σ, the standard deviation of the univariate Gaussian. Its value determines the diversity of the population. Generally speaking, a small value restricts individuals in a small area around the mean vector while a large area can be searched with a large σ. In both cases, every point in the search space has the chance to be sampled, with probability concentrated around the mean according to the value of σ. So, in theory, $PBIL_G$ has the property of global optimization. In practice, the value of σ should not be too large because the population size is limited and using a large σ means distributing limited number of individuals in a large area. As a result, the information about the structure of the problem collected from these individuals can be inaccurate. As the value of σ increases, $PBIL_G$ approaches random search behaviour. A strategy for dynamically adapting the value of σ based on the distribution of best individuals has also been proposed[7].

This algorithm is quite different from $PBIL_B$ and $PBIL_I$ described above. In fact, it works as a population-based hill-climbing algorithm coupled with Gaussian mutation. This is shown with the Stochastic Hill Climbing with Learning by Vectors of Normal Distributions (SHCLVND) algorithm[11].

$PBIL_G$ is very sensitive to its starting position, especially in multimodal situations because it always focuses on the neighbouring area. In fact, it can quickly move to a local optimum close to its starting position and get stuck there. Since it moves towards/away from the best/worst individuals in the population, no matter their fitness values and positions, the mean vector will not converge towards the local optimum; rather, it will oscillate around it. As mentioned above, in order to make the initial distribution general, $PBIL_G$ is restricted to start from the middle of the search space. Surprisingly, two of the test functions (F1 and F2) used in [7] have the origin as their global optima (i.e., the search space is symmetric about the origin), which means that experiments conducted on these two test functions actually started with the model centred at the global optima! A comprehensive set of experiments investigating the performance of $PBIL_G$ only using $X^{best,1}$ can be found in [12].

Another problem arises from the way those individuals chosen to update the mean vector combine. In Eq.1, each individual has equal weight, making the same contribution to the updating. For the two best individuals, this may not be a problem but if we want to extend it to more samples, simply averaging them may be inappropriate. Instead, each individual's fitness value could be explicitly considered.

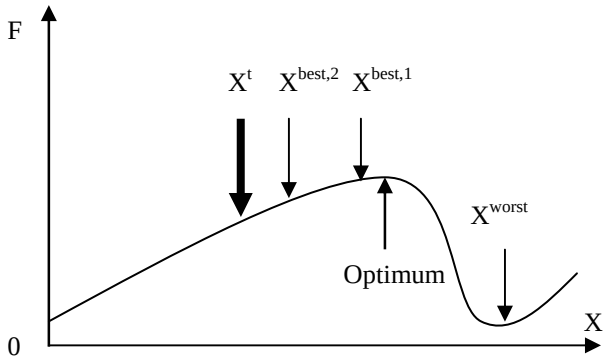


Figure 1: A sample of searching on landscape

A negative sample has also been used in the updating rule. Moving away from the worst sample may seem reasonable but this is not always the case. An example is shown in Figure 1 where the five arrows (from left to right) represent the mean vector, the second best individual, the best individual, the optimum and the worst individual. The optimum that $PBIL_G$ is looking for is on the right of the current mean vector. In fact, the first and second best individuals found are also on the right, which will drag the mean vector towards the optimum. However, the worst individual is on the right of the mean vector too. According to Eq.1, the linear combination of these three individuals is very likely to be in the left of the mean vector, depending on the exact values of these individuals. If so, the mean vector will be updated to the left, which is not what is intended.

Furthermore, there are some other problems with the linear combination in Eq.1. For example, if $-X^{worst}$ is regarded as a new individual, the linear combination will be the summation of three individuals, which may exceed the search space and the mean vector may also be moved outside of the search space, especially with a large learning rate. So, it seems to be more reasonable to use the average value instead of the summation (i.e., $-X^{worst}$ should also be limited within the search space if necessary to make the whole combination within the search space). Another issue is that Eq.1 is not even a correct implementation of using the negative sample. Consider a 1D sample: if $|X^t|>|X^{worst}|$, the component $-X^{worst}$ will actually try to modify $X^t$ towards $X^{worst}$, the worst sample!

### 2.4 Extensions to $PBIL_G$

*2.4.1 A New Updating Rule*

From the discussion above we can see that usually only a small fraction of the population is chosen to update the probability vector of $PBIL_G$ while others are simply discarded. However, each individual in the population tells us something about the structure of the landscape. If we only utilize the best ones and the worst one, we will actually lose a lot of information contained in the population. In order to thoroughly exploit this information, a new updating rule can be formulated that takes into account all individuals. Below is a new mean vector updating rule of $PBIL_G$ for maximization problems.

$$X^{t+1} = X^t + \alpha \cdot \sum_{i=1}^{N} (x_i - X^t) \cdot \frac{f(x_i)}{\sum_{i=1}^{N} f(x_i)} \qquad (2)$$

In Eq. 2, the amount of modification of the mean vector is decided by the product of the learning rate and the sum of the difference between the mean vector and each individual, weighted by the ratio between the fitness value of that individual and the sum of the fitness values of all individuals. It can be also easily modified to deal with minimization problems and/or negative fitness values. Compared to the former updating rule, the new one takes into account all individuals and it does not explicitly distinguish between "good individuals" and "bad individuals". The contribution of each individual is simply decided by its position and the fitness value.

*2.4.2 A Self-Adaptive Learning Rate*

Usually, a very small value is adopted for the learning rate of $PBIL_G$ (e.g., 0.01). The major reason is to keep the algorithm reliable, resistant to some random sampling error. However, this reliability is gained at the cost of low convergence rate. Fortunately, under certain situation, it is possible to employ a large learning rate to accelerate the convergence while keeping the reliability.

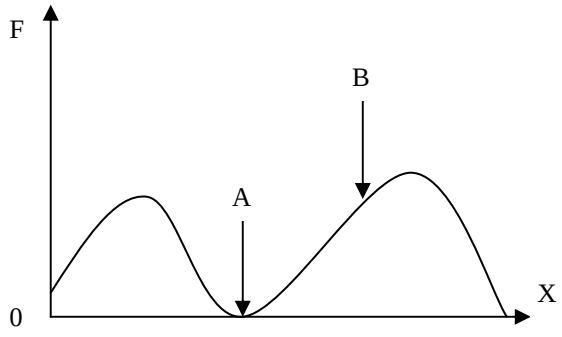


Figure 2: Learning rate vs. Landscape

For example, Figure 2 shows a local area of a multimodal landscape. If the mean vector is at position A, good individuals are often located on both sides. In order to make sure (i.e., with high probability) that the mean vector will eventually move towards the better optimum (right), a very small learning rate is needed. In fact, $PBIL_G$ will oscillate around position A for a while and stochastically drift right. As a contrast, when $PBIL_G$ is at position B, far from the worse optimum and close to the better optimum, most good individuals will be on the right side and it can "confidently" move right.

1. Initialize the learning rate α to a small value p
2. If the direction of movement in current generation is the same as that in the last generation, increase the learning rate (up to 1.0):
   $\alpha(t+1) = \alpha(t) \cdot (1+q)$ $(q>0)$ (Bonus)
   Otherwise reset its value to p:
   $\alpha(t+1) = p$ (Penalty)

Table 2: A Strategy for Self-Adaptive Learning Rate

The general idea in the above situation is to use a small learning rate in position A and use a large learning rate in position B. This strategy is implemented in Table 2. Here, we maintain a separate learning rate for each variable. The reason behind this strategy is that when the mean vector oscillates, it suggests that there may be multiple attractors in the neighbouring area. Hence, the algorithm should be cautious by using a small learning rate. When $PBIL_G$ moves in the same direction within consecutive generations, it may imply that there is only one major attractor and it will be safe to gradually increase the learning rate to speed up the movement.

For the value of the parameter q, which controls how fast we want to increase the learning rate, it can be a predefined constant such as 0.1 or 0.2 or could be decided dynamically by some heuristic. For example, maybe we might choose the value based on the distribution of good individuals. If the majority of good individuals are on the same side that $PBIL_G$ will move towards, we can probably increase the value of q. For simplicity, in this paper, a predefined constant value of q is used.

*2.4.3 Limitations*

Although we have proposed some modifications that hopefully could improve the performance of $PBIL_G$, the algorithm has a seemingly more inherent shortcoming. What we have tried to do is to increase the reliability and the convergence speed of this algorithm but this does not change the fundamental mechanism. That is, even if $PBIL_G$ can quickly, reliably converge to the better optimum instead of the worse optimum (see Figure 2), it may then get stuck there if the global optimum is far from that local optimum and there are no intermediate local optima that can be used as steps.

The key to this problem resides in the Gaussian model it employs. Although all points can be sampled in theory, it focuses on a small area around its mean vector from the beginning of evolution. That is, $PBIL_G$ has strong bias towards the search space.

## 3 PBIL Based On Histograms ($PBIL_H$)

### 3.1 Why Histograms?

In Section 2, we have reviewed $PBIL_B$ and two extensions of $PBIL_B$ in continuous spaces. $PBIL_I$ has some similarity to $PBIL_B$ in that they both start from a very general distribution and then gradually concentrate on a smaller area. However, the movement of boundary in $PBIL_I$ is deterministic and points outside the interval are ignored permanently. In fact, there is another potential problem with this approach. Initially, each interval covers the whole space in each dimension and $PBIL_I$ is simply doing random searching. Only when the algorithm is confident enough that the global optimum is in the upper half interval or the lower half interval, would the interval be reduced by half but how about if the algorithm is not "convinced" for a long time? If so, $PBIL_I$ will keep random searching within both upper and lower intervals. Here, the problem is that the boundary will be either unmoved or moved dramatically by half of the interval. In fact, the searching area should be reduced smoothly and it should always be possible for points outside the current focus to be sampled, even with very low probability. As discussed above, $PBIL_G$ employs in contrast a local searching mechanism, a Gaussian model moving around the search space. This is quite different from the original idea of $PBIL_B$ in that it has strong bias from the beginning to the end. It does not shrink its searching area but simply localizes the search in space.

Can we suggest an alternative model for PBIL in continuous spaces? This model should be general enough to be capable of representing the whole search space without any initial bias for the sake of global optimization and it should also be able to concentrate the searching effort into a smaller area gradually.

In this section, we propose to use histograms as the probabilistic model for PBIL in continuous spaces. The histogram approach is one of the most common models used for non-parametric density estimation in Machine Learning and Pattern Recognition [13] and has a number of advantages. For example, there are no parameters to learn and it can be created very efficiently because it only needs to check each point one by one. Furthermore, it is also extremely flexible because it makes no assumption about the distribution of the data points. One of its disadvantages is that in high-dimensional spaces, the number of bins needed to create a reasonable histogram can be huge, as can be the number of data points needed. Suppose we divide each dimension into B bins, then for a space of N-dimension, totally $B^N$ bins are required, which will make this model intractable for large N. However, since we are talking about PBIL in which no dependence is considered, we can maintain a histogram for each variable independently. So, the number of bins required is reduced dramatically to B•N.

Histograms have already been used in model-based EAs[14, 15]. In previous work, only a few data points are selected and put into the corresponding bin and each histogram is simply a counter of frequency. As a result, the value of each bin is decided by the number of data points that fall into it. In order to utilize all individuals and also incorporate the fitness value of each of them, in our algorithm, some changes have been made. That is, the value of each bin represents the fitness value not the number. We use the fitness value to represent the estimation of the goodness of the range that this bin stands for in continuous spaces. Another thing is that it is likely that several points may belong to a single bin. If so, the value of this bin will not be incremented as in the case of being a counter but the highest fitness will be retained.

### 3.2 Algorithm Framework

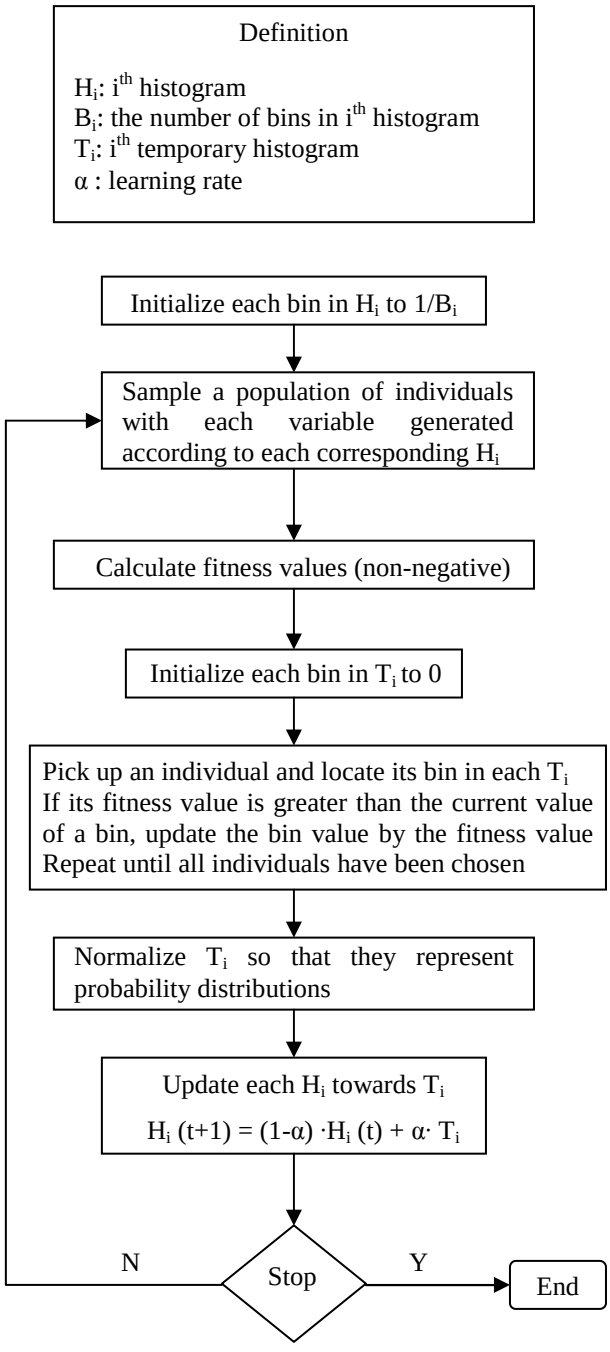


Figure 3: Flow diagram of $PBIL_H$

From the framework in Figure 3 we can see that initially each histogram is simply a uniform distribution, capable of generating all individuals with equal probability. In each generation, a population of individuals will be generated by sampling from those histograms. More specifically, for each variable in each individual, a bin in the corresponding histogram will be chosen in proportion to its value. The actual value assigned to each variable can be then chosen randomly within the range that the bin stands for. After all individuals are generated and evaluated, an empty temporary histogram will be created for each variable. Next, all individuals will be chosen one by one to fill in them and the value of each bin will be set to the fitness value of the fittest individual that falls into this bin instead of the number of individuals belonging to it. After normalization, these temporary histograms will represent the probability distributions estimated from the current population. Finally, each original histogram will be updated towards each corresponding temporary histogram in the same incremental manner as in $PBIL_B$.

In summary, a set of histograms will be evolved based on the fitness values of all individuals. In fact, the real-valued element in the probability vector of $PBIL_B$ is an extreme case of the histogram. If we apply histograms in binary spaces, each histogram will contain only two bins, representing 0 and 1 respectively and only one real-valued element is needed to describe each histogram due to the complementarity. Within each histogram, bins corresponding to optima will be expected to have higher values than others and the searching will gradually be biased towards areas represented by these bins.

## 4 Experiments

### 4.1 Objective

We have discussed the updating rule, the learning rate and a new framework for PBIL in continuous spaces. In this section, we tested our ideas through a set of experiments. Certainly, in order to draw any general conclusion, a large number of experiments on a wide range of test problems are needed to make sure the results are reliable. Here, however, we wanted to focus on some specific problems to obtain an intuitive understanding of our analysis. At the same time, we do not intend to claim that those approaches proposed are perfect and superior to others under all situations but we did want to point out some potential directions towards the improvement of PBIL.

### 4.2 Methodology

We conducted two experiments. The first experiment compared $PBIL_G$ with the new Gaussian-based PBIL ($PBIL_N$) employing the new updating rule (Eq. 2) and the self-adaptive learning rate (Table 2). The purpose was to demonstrate that, facing with multiple attractors, $PBIL_N$ is more likely to converge to the better one and/or within less time in terms of the number of function evaluations. For this purpose, we designed a 2D landscape with 2 peaks (Figure 4). Each peak is a weighted Gaussian probability density function. The fitness value of a point in the landscape is decided by the highest value returned from all density functions. Table 3 shows the parameters of the two optima (i.e., one density function is scaled down to represent the local optimum). We started $PBIL_G$ and $PBIL_N$ at the origin, which is also the middle position between the two optima (Figure 5). Instead of recording the best individual found in terms of fitness value, we examine the mean vector itself as the movement of the model of the landscape maintained by PBIL. The performance of each PBIL was evaluated by the frequency that the mean vector moved to a neighbouring area around the global optimum (over several trials) and how many generations were needed to achieve this.

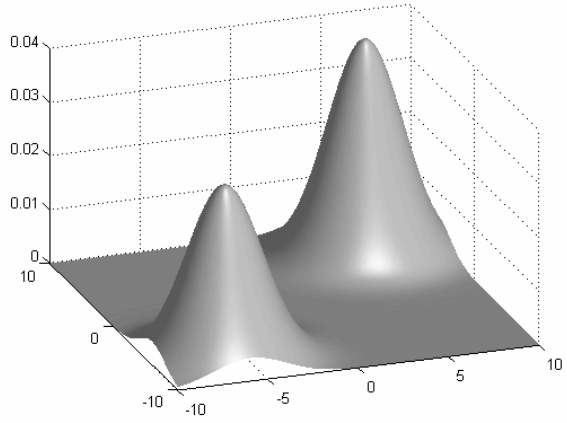


Figure 4: The landscape with two optima

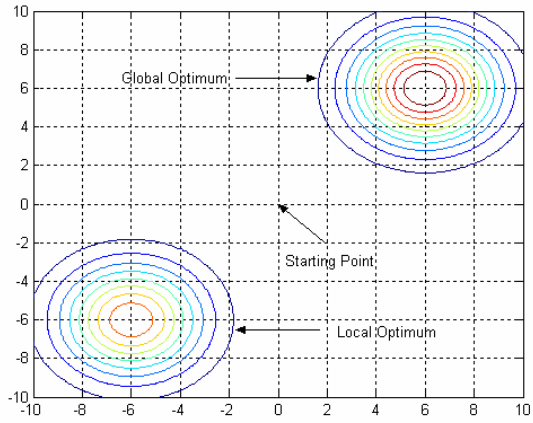


Figure 5: The contour of the landscape

| Peak | Mean | Variances | Scalar | Value |
|---|---|---|---|---|
| Global | (6,6) | (4.0, 4.0) | 1.0 | 0.039789 |
| Local | (-6,-6) | (4.0, 4.0) | 0.8 | 0.031831 |

Table 3: Parameters of the landscape

In the second experiment, we compared $PBIL_H$ with $PBIL_G$ on the same landscape. The purpose was to demonstrate that $PBIL_H$ is robust and has better global optimization ability on this problem (i.e., less likely to get stuck on the local optimum). Since $PBIL_H$ does not employ a mean vector, we used the fitness value of the best individual found so far as the performance criterion.

### 4.3 Experiment Results

For the first experiment, we simply adopted some common parameter values (Table 4). The learning rate of $PBIL_N$ was initially set to 0.01, the same as that of $PBIL_G$ but could be increased during evolution until 1.0. The scalar q was set to 0.2, which means that each time we increased the learning rate by 20%. When the Euclidean distance between the mean vector and the global optimum was less than the predefined distance threshold, we would record it as a successful trial.

Population Size: 50
Maximum Generation: 400
Standard Deviation: 1.0
Distance Threshold: 0.5

For $PBIL_G$:
Learning rate: 0.01

For $PBIL_N$:
Learning rate: [0.01, 1.0]
Scalar q: 0.2

Table 4: Parameters for the first experiment

| | Success Rate | Number of Generations |
|---|---|---|
| $PBIL_G$ | 85% | 161 |
| $PBIL_N$ | 95% | 122 |

Table 5: $PBIL_G$ vs. $PBIL_N$ (over 200 trials)

Table 5 summarizes the results over 200 trials. It is clear than $PBIL_N$ achieved a higher success rate than $PBIL_G$ and its mean vector could move to the global optimum within less time. In order to have deeper understanding of the strategy for self-adapting the learning rate, we tested two variations of $PBIL_G$: $PBIL_{G1}$ and $PBIL_{G2}$. The first one employed the same self-adaptive learning rate as that in $PBIL_N$ while the second one used a large learning rate: 0.05. From Table 6 we can see that a larger learning rate accelerated the convergence rate but at the cost of reliability (i.e., the success rate). Instead, a self-adaptive learning rate could help $PBIL_G$ have a good balance between these two goals.

| | Success Rate | Number of Generations |
|---|---|---|
| $PBIL_{G1}$ | 80% | 32 |
| $PBIL_{G2}$ | 68% | 31 |

Table 6: $PBIL_{G1}$ vs. $PBIL_{G2}$ (over 200 trials)

Having demonstrated the usefulness of the new updating rule and the self-adaptive learning rate, we compared $PBIL_G$ and $PBIL_H$. For $PBIL_H$, there are three parameters: the population size, the learning rate and the number of bins in each histogram. In our experiments, the population size and the learning rate were the same as those in $PBIL_G$ and each histogram was divided into 50 bins. Again, each algorithm was run for 400 generations in each of the 200 trials. For $PBIL_G$, the initial position of its mean vector was randomly chosen in the whole search space. The best individuals found within 400 generations were recorded for comparison.

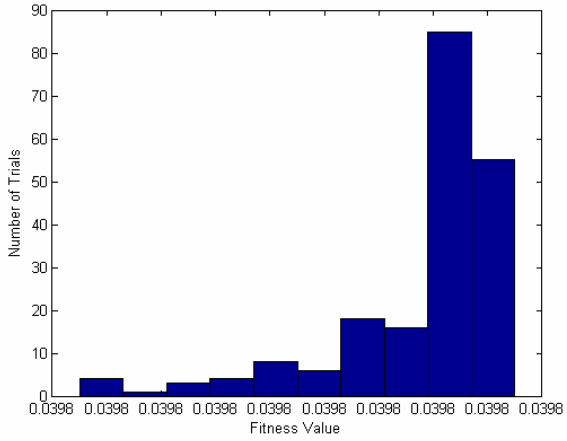


Figure 6: Performance of $PBIL_H$

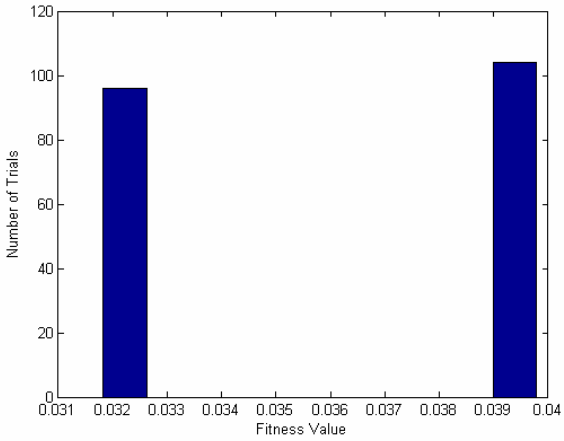


Figure 7: Performance of $PBIL_G$

The distribution of the fittest individuals for each algorithm is shown in Figures 6&7. It is evident that $PBIL_H$ was always successful at finding the global optimum as all best individuals found had fitness values extremely close to the global optimum (0.039789). As a contrast, $PBIL_G$ could only find the global optimum in around half of the runs. In fact, in other runs, it got stuck in the local optimum (0.031831). The explanation is that there were two optima and which optimum $PBIL_G$ finally found was highly dependent of the starting position. Since the two optima had similar basin sizes and the initial position was randomly generated, it is reasonable to see the bimodal distribution in Figure 7.

Finally, we selected one particular trial of $PBIL_H$ and illustrated how the probabilistic model evolved during evolution. Figure 8 shows the evolving process of one histogram (i.e., both two histograms had very similar evolving process because of the symmetry of the landscape). Initially, the probabilistic model was simply a uniform distribution. Gradually, the values of bins corresponding to optima increased and the values of other bins decreased accordingly.

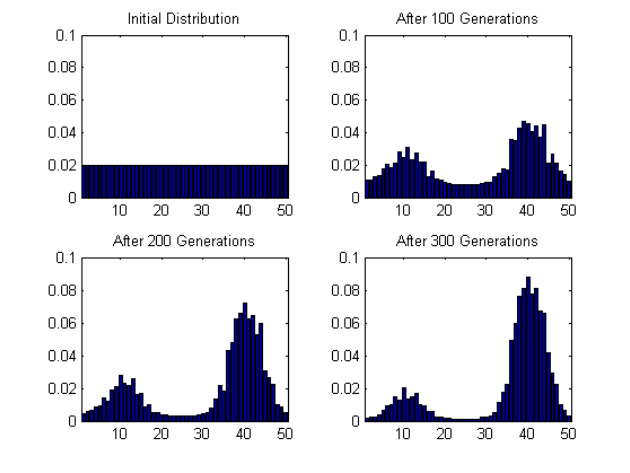


Figure 8: Evolution of histogram

## 5 Conclusion

This paper reviewed the fundamental mechanism and some basic ideas of PBIL including $PBIL_B$ and two extensions of $PBIL_B$ in continuous spaces. We pointed out some potential problems with the updating rule and the small learning rate used before. A new updating rule that takes into account all individuals and their fitness values was proposed. In order to find a better balance between reliability and convergence speed, we also introduced a strategy for self-adapting the learning rate during execution. Experiment results showed that both techniques could improve the performance of $PBIL_G$. In order to overcome some disadvantages of previous approaches to continuous PBIL, a histogram probabilistic model was used. Our approach is quite different from previous work in that we used the fitness value instead of the number of points as a measure of the goodness of each bin. Experiment results showed that $PBIL_H$ greatly outperformed $PBIL_G$ in terms of global optimization ability. Furthermore, each peak in the histogram represents a promising area, which means that $PBIL_H$ could search and maintain several optima at the same time.

Certainly, the work reported here is far from a complete story. Much more work is needed to investigate how to design a better updating rule (e.g., best/worst individuals vs. whole population with fitness values) and a better strategy for self-adapting the learning rate. We intend to use this as a good starting point for future work. Lastly, the histogram method presented is only a preliminary framework and there is still much room for improvement. For example, the number of bins in each histogram may be self-adaptive during evolution and the possibility of using more complex histogram model to capture dependences among variables also needs further investigation.

## Acknowledgment

This work was partially supported by the Australian Postgraduate Award granted to Bo Yuan.

## References

[1] Fogel, D.B. "An Introduction to Simulated Evolutionary Optimization", *IEEE Transactions on Neural Networks*, **5**(1): pp. 3-14, 1994.

[2] Pelikan, M., Goldberg, D.E., and Lobo, F. "A Survey of Optimization by Building and Using Probabilistic Models", Tech. Report No.99018, University of Illinois at Urbana-Champaign, 1999.

[3] Larranaga, P. and Lozano, J.A., Eds. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2001.

[4] Baluja, S. and Caruana, R. "Removing the Genetics from the Standard Genetic Algorithm", Tech. Report CMU-CS-95-141, Carnegie Mellon University, 1995.

[5] Baluja, S. "Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning", Tech. Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[6] Servet, I., Trave-Massuyes, L., and Stern, D. "Telephone Network Traffic Overloading Diagnosis and Evolutionary Computation Techniques". In *Artificial Evolution 97*, J.-K. Hao, et al. Eds., France, Springer, pp. 137-144, 1997.

[7] Sebag, M. and Ducoulombier, A. "Extending Population-Based Incremental Learning to Continuous Search Spaces". In *Parallel Problem Solving from Nature-PPSN V*, A.E. Eiben, et al. Eds., Amsterdam, Springer, pp. 418-427, 1998.

[8] Shapiro, J.L. "Scaling of Probability-based Optimization Algorithms", *To Appear in Advances in Neural Information Processing Systems 15 (NIPS2002)*, 2003

[9] Baluja, S. "An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics", Tech. Report CMU-CS-95-193, Carnegie Mellon University, 1995.

[10] Baluja, S. and Davies, S. "Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space". In *the Fourteenth International Conference on Machine Learning*, D.H. Fisher Ed., pp. 30-38, 1997.

[11] Rudlof, S. and Köppen, M. "Stochastic Hill Climbing with Learning by Vectors of Normal Distributions". In *The First Online Workshop on Soft Computing (WSC1)*, Najoya, Japan, 1996.

[12] Gallagher, M. "An Empirical Investigation of the User-Parameters and Performance of Continuous PBIL Algorithms". In *Neural Networks for Signal Processing X*, B. Widrow, et al. Eds., IEEE, pp. 702-710, 2000.

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

[14] Tsutsui, S., Pelikan, M., and Goldberg, D.E. "Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain", Tech. Report No. 2001019, University of Illinois at Urbana-Champaign, 2001.

[15] Bosman, P.A.N. and Thierens, D. "An Algorithmic Framework For Density Estimation Based Evolutionary Algorithms", Tech. Report UU-CS-1999-46, Utrecht University, 1999.