Probabilistic model building Genetic Algorithm (PMBGA): A survey

Siddhartha K. Shakya

Technical Report.
Computational Intelligence Group,
School of computing, The Robert Gordon University,
Aberdeen, Scotland, UK.
August 2003.

Abstract

Probabilistic model building Genetic Algorithm (PMBGA) is a novel concept in the field of evolutionary computation which is motivated by an idea of building a probabilistic model of the population to preserve important building blocks in subsequent generation. Growing number of research is being carried out in this field and different variant of PMBGA s has been purposed. Aim of this paper is to survey currently existing PMBGAs, categorise them according to their used probability model, describe their workflow and analyse their strengths and weakness.

1. Introduction

Genetic Algorithms (GAs) are a class of optimization algorithm motivated from the theory of natural selection and genetic recombination. It tries to find better solution by selection and recombination of promising solution. It works well in wide verities of problem domains. However, sometimes simple selection and crossover operators are not effective enough to get optimum solution as they might not effectively preserve important patterns (known as building blocks or partial solutions) in chromosome. It often happens in the problem domains were the building blocks are loosely distributed. The search for techniques to preserve Building Blocks lead to the emergence of new class of algorithm called Probabilistic Model Building Genetic Algorithm (PMBGA)[Pelikan, Goldberg & Lobo(1999)] also known as Estimation of Distribution Algorithm (EDA) [Mühlenbein & Paaß (1996)]. The principle concept in this new technique is to prevent disruption of partial solutions contained in a chromosome by giving them high probability of being presented in the child chromosome. It can be achieved by building a probabilistic model to represent correlation between variables in chromosome and using build model to generate next population. PMBGA is a developing area in the field of evolutionary and Genetic algorithms. First purposed by [Baluja (1994)] with the simplest form on this class and so called Population

Based Incremental Learning (PBIL)(early work on PBIL has been published with the name Equilibrium Genetic Algorithm (EGA) together with [Jues, Baluja & Sinclair (1993)]), a dozen or more different variants of PMBGA has been proposed till the date and are subject of active research for evolutionary and Genetic Algorithm community. Most of the early PMBGAs were focused on binary representation of solution vector i.e focused on desecrate problem domain. They has been later modified to work on continues domain and published with different names. For simplicity of this paper, we will consider PMBGAs on desecrate domain however continuous variant of most of the algorithms discussed in this paper has already been purposed.

This paper is a survey of existing PMBGAs. The paper is organised as follows. Section 2 will describe general frame work of PMBGA and categorize existing PMBGAs in three different classes according to their used probability model: Univariate, Bivariate and Multivariate model. Section 3 will describe algorithms using Univariate Model of probability distribution followed by Section 4 and 5 describing algorithms using Bivariate and Multivariate model respectively. Section 6 will briefly discuss the bottle neck problem for PMBGAs so called problem of learning probabilistic model. We conclude paper by summarising the achievements made so far in PMBGAs and giving overview of further work.

2. General PMBGA model

As in traditional GA, all PMBGAs start with generating initial population of size *M*. Then *N* individuals out of *M* are selected according to chosen selection criteria. Estimation of distribution is carried out (The joint probability distribution of individual is calculated) from selected set of individuals and used to sample offspring to replace parent population.

The general PMBGA is as follows:

1. Generate initial population of size *M*
2. Select *N* promising solution where *N*<=*M*
3. Calculate joint probability distribution of selected individuals
4. Generate offspring according to the calculated probability distribution and replace parent.
5. Go to step 2 until termination criteria are meet

Neither crossover nor mutation is involved in the process of generating offspring and completely replaced by calculation and sampling of probability distribution (However in some PMBGAs operators similar to mutation is allowed [Baluja (1994)]). In contrast with implicit processing of building blocks in GA, the process here is explicit and fully depends on used probability model. The goodness of probability model is the decisive factor in PMBGA performance. As accurate the probability model, as effective the algorithm will be in preventing disruption of important building blocks.

Before going any further, let us first introduce the notations used in this paper. We follow the approach taken by [Larranaga, Etxeberria, Lozano and Pena (1999)].

Let X_i be a random variable and x_i be one of its possible value, then $r(X_i = x_i)$ (or simply $r(x_i)$) represents the *generalised probability density function* (gpdf) over the point x_i. Now let $X = \{X_1, X_2, ..., X_n\}$ be the vector (individual) with n variables and $x = \{x_1, x_2, ..., x_n\}$ be the value taken by each variable of vector X then $r(X = x)$ (or simply $r(x)$) represents the *joint generalised probability density function* (jgpdf) of X. Similarly the generalised conditional probability density function of variable X_i taking value x_i given value x_j for the variable X_j will be represented as $r(X_i = x_i \mid X_j = x_j)$ (or simply $r(x_i \mid x_j)$).

If the problem domain is discrete, i.e. If variable X_j is discrete, $r(X_i = x_i) = p(X_i = x_i)$ (or simply $p(x_i)$) is the *univariate marginal distribution* of the variable X_i. If all the variables in X are discrete then $r(X = x) = p(X = x)$ (or simply $p(x)$) will be *complete joint probability distribution*. Similarly *conditional probability* that X_i will take value x_i given value x_j for the variable X_j can be denoted as $r(X_i = x_i \mid X_j = x_j) = p(X_i = x_i \mid X_j = x_j)$ (or simply $p(x_i \mid x_j)$).

In case of continuous X_j i.e. in continuous domain, $r(X_i = x_i) = f(X_i = x_i)$ (or simply $f(x_i)$) is the *density function* of the variable X_i. If all the variables in X are continuous then $r(X = x) = f(X = x)$ (or simply $f(x)$) will be *joint density*. Similarly $r(X_i = x_i \mid X_j = x_j) = p(X_i = x_i \mid X_j = x_j)$ (or simply $p(x_i \mid x_j)$) will be *conditional density* of X_i taking value x_i given value x_j for the variable X_j.

As we will be focusing on discrete domain in this paper, let us describe marginal and conditional probability in discrete domain in more detail.

We can define *univariate marginal distribution* of i^{th} variable $p(X_i = x_i)$ (or simply $p(x_i)$) as a sum of probability of all vectors in population having $X_i = x_i$ which can be written as:

$$p(x_i) = \sum_{X, X_i = x_i} P(X)$$

Let X_S be the sub vector of X and x_S be the possible set of values taken by X_S then *marginal probability* $p(X_S = x_S)$ (or simply $p(x_S)$) is defined as sum of probability of all vectors in population having same set of values x_S for sub vector X_S and can be written as

$$p(x_S) = \sum_{X, X_S = x_S} P(X)$$

Note that *univariate marginal distribution* is a simple case of marginal distribution where sub vector consist of single variable.

Let X_A and X_B be disjoint sub vectors of X and x_A and x_B be the possible subset of values taken by them respectively then *conditional probability* $p(X_A = x_A \mid X_B = x_B)$ (or simply $p(x_A \mid x_B)$) can be calculated as

$$p(x_A \mid x_B) = \frac{p(x_A, x_B)}{p(x_B)}$$

According to the probability theory the *complete joint probability distribution* of a set of individuals $P_r(X = x)$ or simply $P_r(x)$ where each variable $X = \{X_1, X_2, ..., X_n\}$ takes values $x = \{x_1, x_2, ..., x_n\}$ can be calculated as

$$P_r(x) = p(x_1 \mid x_2, ..., x_n) p(x_2 \mid x_3, ..., x_n) ..... p(x_{n-1} \mid x_n) p(x_n)$$

This paper will differentiate PMBGAs according to their used probability model. We follow the approach taken by previous reviewers and categorise PMBGA into three different classes by their used probability model: Univariate, Bivariate and Multivariate Models. [Larranaga, Etxeberria, Lozano & Pena (1999)][Pelikan, Goldberg & Lobo (1999)]

3. Univariate Model

Algorithms in this category do not consider any dependencies among variable in individual i.e. considers building blocks of order one. So the joint probability distribution becomes simply the product of univariate marginal probability of all variables in individual.

$$P(x) = \prod_{i=1}^{n} p(x_i)$$

Where,

n	: Total number of variable in an individual (chromosome length).
$p(x_i)$	: Univariate marginal probability of ith variable

Due to its simplicity the algorithm in this category are computationally very efficient and performs excellent in linear problem such as function optimization where the variables are not significantly interdependent. However these algorithms fail on complex problems where variables interact with each other. Population based Incremental Learning (PBIL) [Baluja (1994)], Univariate Marginal Distribution Algorithm (UMDA) [Mühlenbein & Paaß (1996)] and Compact Genetic Algorithm (cGA) [Harik, Lobo & Goldberg (1998)] uses univariate model of probability distribution. Figure 3.1 below is the graphical representation of the probability model used by these algorithms.

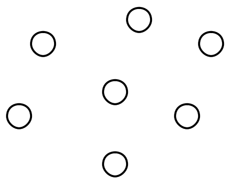


Figure 3.1: Graphical representation of probability model assuming no dependency among variables.

3.1. Population Based Incremental Learning (PBIL)

One of the earliest works in the PMBGA field was Population Based Incremental Learning (PBIL) algorithm purposed by [Baluja (1994)]. PBIL was motivated by the idea of combining GAs with *Competitive Learning* which is often used in training Artificial Neural Networks. PBIL considers binary representation of individual. It starts with initialization of a *probability vector* and maintains and updates it throughout the whole computation process. General PBIL algorithm follows:

1. Initialize probability vector $p = \{p_1, p_2, ..., p_n)\}$ with 0.5 at each position. Each variable represents probability of value '1' being presented in same position of child chromosomes.
2. Sample M >> 0 individuals according to probabilities in p and evaluate them.
3. Update probability vector according to fittest individual $S = \{s_1, s_2, ..., s_n\}$ using following rule:

$$p_i = p_i * (1.0 - LR) + s_i * LR$$

where LR is Learning Rate Value.

4. If mutation condition passed, Mutate Probability vector using following rule:

$$p_i = p_i * (1.0 - MS) + random(0 \;\; or \;\; 1) * MS$$

where MS is amount of mutation to affect the probability vector.

5. Go to step 2 until termination criteria satisfied.

Several different variant of PBIL has been purposed with some simple extension in method of updating probability vector. One of them is to move probability vector towards best N individuals, where $N << M$ rather than to move towards single best individual. Another possibility is to move not only towards best vector but also to move away from worst individual. Detail work and experiments on PBIL can be found in [Baluja (1994)]

3.2. Univariate Marginal Distribution Algorithm (UMDA)

Univariate Marginal Distribution Algorithm (UMDA) (extended to more complex model BMDA, FDA) was first purposed by [Mühlenbein & Paaß (1996)] and is one of the early work in the field of PMBGAs. UMDA captures general concept of PMBGA with Univariate model. Different variants of UMDAs has been purposed and their detail mathematical analysis has been carried out by Mühlenbein and his colleagues [Mühlenbein & Paaß (1996)],[Mühlenbein (1997)].

General UMDA is as follows:

1. Generate initial population of size $M >> 0$ randomly
2. Select N promising solution where $N<=M$
3. Calculate univariate marginal probability $p(x_i)$ from selected individuals
4. Replace parent with M new individuals generated according to the distribution $P(x) = \prod_{i=1}^{n} p(x_i)$
5. Go to step 2 until termination criteria satisfies

Unlike PBIL which uses univariate marginal probability to update probability vector, above algorithm relay only on univariate marginal probability $p(x_i)$ to generate new solution. However by introducing memory (probability vector as in PBIL) it is possible to incrementally change sampling distribution. Following algorithm so called Simple Univariate Marginal Distribution Algorithm (SUMDA) (also known Incrimental Univariate Marginal Distribution algorithm IUMDA) [Mühlenbein (1997)] uses memorised previous sampling distribution (probability vector) as well as current univariate marginal probability to calculate current sampling distribution.

1. Initialize probability vector $p = \{p_1, p_2, ..., p_n\}$
2. Generate M individuals according to distribution $P(x) = \prod_{i=1}^{n} p_i$
3. Select N promising solution and calculate univariate marginal probability of selected solution $p(x_i)$.
4. Update probability vector according to the following rule
   $$p_i = p_i + l\,(p(x_i) - p_i)$$
   Where, l is a controlling parameter for convergence. The smaller l is, the slower convergence speed. Authors provide evidence that performance is better when $l = 1$
5. Go to step 2 until termination criteria satisfies.

Apart from the updating rule, the procedure of SUMDA is similar to PBIL. The detail work on UMDA and its extensions can be found in [Mühlenbein & Paaß (1996)], [Mühlenbein (1997)] and [Mühlenbein, Mahnig, & Rodriguez (1999)].

3.3. Compact Genetic Algorithm (cGA)

Compact Genetic Algorithm (cGA) [Harik, Lobo & Goldberg (1998)] (later extended to Extended Compact Genetic algorithm (ECGA)) is motivated by the previous works done in the field of random walk model and also assumes no overlapping building blocks are contained in chromosome i.e conceders only building blocks of order 1.

cGA also maintains probability vector as in PBIL. However, unlike PBIL, cGA samples only two individual at a time, compete them and uses allele value of winning individual which is distinct from allele value of loosing individual to update the probability vector leaving probably vector unchanged in the position where winning and loosing allele contains same value. The process continues until probability vector converges.

General cGA is shown bellow:

1. Initialize probability vector $p = \{p_1, p_2, ..., p_n\}$ with 0.5 at each position.

2. Sample 2 individuals according to probabilities in *p* and label according to their fitness: *winner* to the fittest and *loser* to less fit

3. Update probability vector using following rule:

```
For i = 1 to n do
    If winner[i] ≠ loser[i] then
        If winner[i]=1 then $p_i = p_i + updaterate$
            Else $p_i = p_i - updaterate$
```

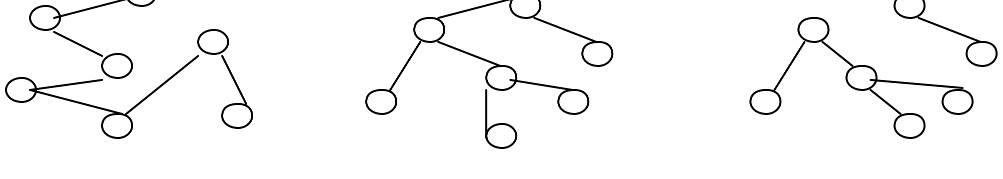
   Where *updaterate* is small value usually defined as 1/population size.

4. Go to step 2 until probability vector converges.

The detail work on cGA can be found in [Harik, Lobo & Goldberg (1998)].

4. Bivariate Model

Algorithms in this category consider pair wise dependencies among variables in chromosome i.e. consider the building blocks of order two. Similarly the probability model becomes more complex than one of univariate model and takes a form of probabilistic network between variables. This class of algorithm performs better in problems with pair wise interaction among variable. However fails in the problems with multiple variable interactions. Mutual Information Maximization for input clustering (MIMIC) [De Bonet, Isbell and Viola (1997)], Combining Optimizers with Mutual Information Trees (COMIT) [Baluja & Davies (1997)], Bivariate Marginal Distribution Algorithm (BMDA) [Pelikan & Mühlenbein (1999)] uses Bivariate Model of probability Distribution. Graphical representation of Probabilistic networks used by these algorithms is shown in figure 4.1.



a. Chain model (MMIC) b. Tree model (COMIT) c. Forest model (BMDA)

Figure 4.1: Graphical representation of probability model assuming dependency of order two among variables.

4.1. Mutual Information Maximization for input clustering (MIMIC)

Mutual Information Maximization for input clustering **(**MIMIC**)** purposed by [De Bonet, Isbell and Viola (1997)] uses chain structure of probability distribution [Figure 4.1 a] which can be written as:

$$P_p(x) = p(x_{i_1} \mid x_{i_2})\, p(x_{i_1} \mid x_{i_2}) \ldots\ldots\ldots p(x_{i_{n-1}} \mid x_{i_n})\, p(x_{i_n})$$

where, $p = i_1 i_2 \ldots\ldots i_n$ is a permutation of the numbers between 1….*n*. The distribution $P_p(x)$ uses p as an ordering for the pair wise conditional probabilities. The goal is to find the ordering p such that distribution $P_p(x)$ matches as closely as possible to the complete joint probability distribution $P_r(x)$.

$$P_r(x) = p(x_1 \mid x_2 \ldots x_n)\, p(x_2 \mid x_3 \ldots x_n) \ldots\ldots p(x_{n-1} \mid x_n)\, p(x_n).$$

Kullback-Liber divergence $D(P_r(x) \mid P_p(x))$ is used to measures the identically between $P_p(x)$ and $P_r(x)$. Searching over all possible permutation of p is computationally hard so MIMIC uses simple greedy algorithm to find p with optimal divergence, which however does not always gives accurate model. The detail work on MIMIC can be found in [De Bonet, Isbell and Viola (1997)].

4.2. Combining Optimizers with Mutual Information Trees (COMIT):

Combining Optimizers with Mutual Information Trees (COMIT) purposed by [Baluja & Davies (1997)] also uses bivariate model of distribution however in contrast with chain distribution used in MIMIC, COMIT uses tree distribution [Figure 4.1 b]. Tree structure has advantage over chain structure in the sense that every chain can be seen as a tree however vice versa is not always true. COMIT used Maximum Weight Spanning Tree (MWST) algorithm to construct a tree structure and further uses it as its probability model which can be written as

$$P(x) = \prod_{i=1}^{n} p(x_i \mid x_j)$$

Where, *j* represents the parent position for *i*. However in the case when parent of *i* th variable does not exist i.e if *i* is the root variable then $p(x_i \mid x_j)$ is generalised to $p(x_i)$. Experimental results presented by authors shows better performance of COMIT over MIMIC, PBIL and GA. The detail work on COMIT can be found in [Baluja & Davies (1997)].

4.3. Bivariate Marginal Distribution Algorithm (BMDA)

Bivariate Marginal Distribution Algorithm (BMDA) purposed by [Pelikan & Mühlenbein (1999)] is an extension to UMDA. BMDA is a more generalised than above two algorithms in this class as it can cover both linear problem as well as problems with pair wise interaction among genes. In contrast to chain and tree structure used in above two algorithms, BMDA uses Forest (set of mutually independent tree) structure to represent probability mode [Figure 4.1 c]. Dependencies among variables are detected during optimization process. BMDA uses *Pearson's chi-square statistics* to measure dependencies which is defined by:

$$X^2 = \sum \frac{(observed - \exp ected)^2}{\exp ected}$$

In term of univariate and bivariate marginal frequency, for the variable position $i \neq j$ we get:

$$X_{i,j}^2 = \sum_{x_i, x_j} \frac{(Np(x_i, x_j) - Np(x_i)p(x_j))^2}{Np(x_i)p(x_j))}$$

The variables *i* and *j* are said to be 95% independent if $X_{i,j}^2 < 3.84$.

Pseudo-code for BMDA is shown below:

1. Generate initial population of size *M >> 0* randomly
2. Select *N* promising solution where *N<=M*
3. Calculate univariate marginal probability $p(x_i)$ and bivariate marginal probability $p(x_i, x_j)$ of selected individuals
4. Construct probability model by measuring dependencies among each pair of individual using Pearson's chi-square statistics.
5. Replace *K* individuals in parent with *K* new individual generated according to constructed probability model.
6. Go to step 2 until termination criteria are satisfies.

The detail work on BMDA can be found in [Pelikan & Mühlenbein (1999)].

5. Multivariate Model

Any algorithms considering interdependency between variable of order more than two can be placed in this class. The probability network representing interdependency of variables obviously becomes more complex and the computation time to construct such network hugely increases making it almost impossible to search through all possible models. Due to its simplicity most of the algorithms in this class uses Greedy heuristic to search a good model, however greedy heuristics does not always guarantees accuracy. Some other complex search algorithms have also been successfully used for this purpose and lots of current research in PMBGAs is focused on finding good heuristic. Extended Compact Genetic Algorithm (ECGA) [Harik (1999)], Factorised Distribution Algorithm (FDA) [Mühlenbein & Mahnig (1999a)], [Mühlenbein & Mahnig (2002)], Bayesian Optimization algorithm (BOA) [Pelikan, Goldberg & Cantú-Paz (1999)], Learning Factorised Distribution Algorithm (LFDA) [Mühlenbein & Mahnig (1999b)], Estimation of Bayesian Network (EBNA) [Etxeberria & Larrañaga (1999)] uses multivariate model of probability distribution. Figure 5.1 below shows the graphical representation of different probabilistic network used by these algorithms.

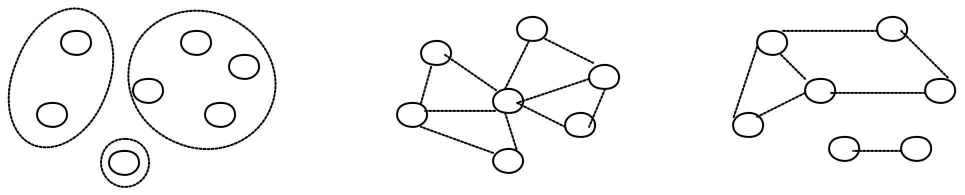


Figure 5.1: Graphical representation of probability model considering multivariate dependency among variables.

5.1. Extended Compact Genetic Algorithm (ECGA)

Extended Compact Genetic Algorithm (ECGA) [Harik (1999)] as claimed by author is a different approach which uses probabilistic model building GA concept to solve the linkage learning problem [Thierens & Goldberg (1993)]. Linkage Learning in GA is to find the building blocks (in this context is a group of variables related with each other where the relationship is not previously known) which should be preserved after crossover and then make GA work with found building Blocks (group of alleles). ECGA is an extension of cGA. The probability model used in ECGA and so called *Marginal Product Model* (MPM) [Figure 5.1 a] is distinct from other previously described model in the sense that MPM includes both univariate marginal distribution as well as multivariate marginal distribution in its probability model (But does not include conditional probability) i.e. It assumes no directional dependency among variables and takes in account marginal probability of set of variables at once. ECGA uses greedy search to find the good MPM model and runs cGA on that model.

Probability model used in ECGA

$$P(x) = \sum_{c \in C} p(x_c)$$

Where, marginal probability of a group of dependent variable c as a whole is represented as $p(x_c)$ and C is a set of grouped alleles. If the constructed model is correct and the problem domain does not contain *overlapping dependencies* then ECGA works well, however not all real life problem are of this kind and can often contain overlapping dependencies in which case ECGA might fail.

The detail work on ECGA can be found in [Harik (1999)].

5.2. Factorised Distribution Algorithm (FDA)

Factorised Distribution Algorithm purposed by [Mühlenbein & Mahnig (1999a)] is an extension of UMDA. UMDA was first extended to *Boltzmann Estimated Distribution Algorithm* (BEDA) which computes *Boltzmann Distribution* by using *Boltzmann selection*. Authors claims Boltzmann Distribution as a good candidate for optimization using a search distribution [Mühlenbein & Mahnig (2002)]. It is proven that BEDA converges to the set of all global optima. However, BEDA is not a practical algorithm as calculation of distribution requires a sum over exponentially may parameters of distributions. Given an Additive decomposition of function (ADF) *factorization theorem* [Mühlenbein & Mahnig (1999a)] can be used to effectively compute factorization of Boltzmann distribution [Figure 5.1 b]. In the context of schemata theorem [Goldberg, D.E. (1989)], factorization theorem tells which previously given schemata are necessary to generate the whole distribution. By using factorization theorem BEDA can be extended into a practical algorithm FDA. If the condition of factorization theorem is fulfilled the convergence proof of BEDA will apply to FDA.

Pseudo-code for FDA taken from [Mühlenbein & Mahnig (2002)] is shown below:

1. Using Given Additive decomposition of function calculate b_i (residuals) and c_i (separators): the sets of variables correlated to each other.
2. Generate M individuals according to uniform distribution $P(x) = \prod_{i=1}^{n} p(x_i)$
3. select $N<=M$ individuals using Boltzmann selection where probability of an individual X being selected is computed as

$$P_b(x) = \prod_{i=1}^{k} p_b(x_{b_i} \mid x_{c_i}) = \frac{\prod_{i=1}^{k} p_b(x_{b_i}, x_{c_i})}{\prod_{i=2}^{k} p_b(x_{c_i})}$$

where, k is a number of correlated set of variables; p_b is a Boltzmann distribution calculated as $p_b(a) = \frac{e^{-\frac{f(a)}{T}}}{Z}$ where, $Z = \sum_b e^{-\frac{f(b)}{T}}$ is a partition function, *f(a)* is fitness of individual a and T is a temperature value which make s Boltzmann selection suitable for optimization problems.

4. Estimate the conditional probabilities $p(x_{b_i} \mid x_{c_i})$ from selected individuals

5. Generate M new individuals according to conditional distribution $P(x) = \prod_{i=1}^{n} p(x_{b_i} \mid x_{c_i})$

6. Go to step 3 until termination criteria satisfies.

The pseudo code shown above is one and the most recent out of several different variant of FDA that can be found in different papers produced during the evolution phase of algorithm. However the key concept that *the Boltzman selection is an essential part of FDA* remains same. FDA however can be run with any selection method but the convergence proof of BEDA will no longer be valid.

FDA requires problem structure in advance in the form of decomposition of function which might not be available in the real world problem. However if the given decomposition is accurate FDA can solve GA hard problems very efficiently and effectively. An extension of FDA not requiring problem structure in advance and so called Learning Factorised Distribution Algorithm (LFDA) has later been purposed by [Mühlenbein & Mahnig (1999b)] and will be discussed later in this paper.

Detailed definition and mathematical analysis of FDA can be found in [Mühlenbein & Mahnig (1999a)], [Mühlenbein & Mahnig (1999b)], [Mühlenbein & Mahnig (2002)].

5.3. Bayesian Optimization algorithm (BOA)

Bayesian Optimization algorithm (BOA) purposed by [Pelikan, Goldberg & Cantú-Paz (1999)] encodes joint probability distribution in the form of *Bayesian Network* [Figure 5.1 c]. Bayesian Network is learnt from the selected set of promising solution. In recent year use of Bayesian Network as estimation of Distribution has been popular among researchers and several different PMBGAs using Bayesian network has been purposed. Learning Factorization Distribution Algorithm (LFDA) [Mühlenbein & Mahnig (1999b)] and Estimation of Bayesian Network (EBNA) [Etxeberria & Larrañaga (1999)] also uses Bayesian network. The score to measure the quality of network is an important factor in Bayesian Network. BOA uses Bayesian-Dirichlet (BD) metric as a measure of quality of network. However they mention possibility of use of other metrics such as *Minimal Description Length* (MDL) metric in their algorithm. BOA uses Greedy algorithm to search through the possible space of networks.

The General BOA follows:

1. Generate initial population of size $M >> 0$ randomly
2. Select N promising solution where $N<=M$
3. Construct a Bayesian Network B using chosen metric and *constraints k*
4. Generate K new individual according to the joint distribution encoded by constructed Bayesian Network B.
5. Create new population by replacing K individuals in parent with generated K new individuals.
6. Go to step 2 until termination criteria are satisfies.

Constraint k in BOA represents the maximum number of incoming edges into each node. The order of dependency considered by the Network and the time taken to construct the Network directly depends on value k.

BOA has been later extended to Hierarchical BOA (HBOA) [Pelikan & Goldberg (2000)]. Underlying concept in HBOA is to decompose problem into some kind of Hierarchical form and to execute BOA on them. HBOA has been later reported to solve two class of complex optimization problem so called Ising Spin Glasses and MAXSAT [Pelikan & Goldberg (2003)].

For detail information on BOA and HBOA see [Pelikan, Goldberg & Cantú-Paz (1999)], [Pelikan & Goldberg (2000)] and [Pelikan & Goldberg (2003)].

5.4. Learning Factorised Distribution Algorithm (LFDA)

Learning Factorised Distribution Algorithm (LFDA) purposed by [Mühlenbein & Mahnig (1999b)] is an extension to FDA. Unlike FDA, LFDA does not need problem structure in advance, rather uses Bayesian network to compute the probabilistic model in each step. The approach is almost similar to the one used in

BOA however difference is being that BOA uses *Bayesian Dirichlet* (BD) score were as LFDA uses a modification of *Minimal Description Length* (MDL) score known as *Bayesian Information Criterion* (BIC) score to measure quality of Bayesian network. More about LFDA can be found in [Mühlenbein & Mahnig (1999b)].

5.5. Estimation of Bayesian Network (EBNA)

Estimation of Bayesian Network (EBNA) purposed by [Etxeberria & Larrañaga (1999)] also uses Bayesian network as encoding of its joint probability distribution. EBNA follows similar approach to LEFDA and BOA. Three different variant of EBNA: $EBNA_{PC}$, $EBNA_{BIC}$, $EBNA_{K2+pen}$ using different Network quality measuring method has been purposed [Larrañaga, Etxeberria, Lozano & Peña (2000a)]. $EBNA_{PC}$ uses Chi square test to check the conditional interdependencies in network. Similar to LFDA, $EBNA_{BIC}$ uses *Bayesian Information Criterion* (BIC) score to measure quality of Bayesian network. $EBNA_{K2+pen}$ combine $EBNA_{BIC}$ approach with a penalizing term introduced to avoid a more complex Bayesian Network. Experimental results presented by authors showed that out of these three algorithms $EBNA_{K2+pen}$ returned best result were as $EBNA_{PC}$ returned worst result. The detail work on EBNA can be found in [Etxeberria & Larrañaga (1999)], [Larrañaga, Etxeberria, Lozano & Peña (2000a)].

6. Learning a Probabilistic Network

From the survey done so far we can conclude that the success and failure of PMBGAs solely depends on used probability model (also known as dependency network). The Computation of dependency network can be seen as a combination of two procedures. First is to choose a correct scoring metric to measure the goodness of network and second is to choose a good search heuristic to find network such that the found network optimizes the scoring metric. Unfortunately, to find the best network, it is required to search through space of all possible networks which in fact is a NP–hard problem in itself. To overcome this problem researchers tend to use simple local search heuristics such as greedy algorithm due to its efficiency, however in many cases, greedy algorithm does not guarantees good solution. The use of other global search algorithm is also restricted because of their computational cost.

In recent years some research has been done on parallel computation of dependency network such that to efficiently compute a good Network in reasonable amount of time. However significant time efficiency is yet to be reported. For more information see [Lozano, J. A., Sagarna, R., Larranaga, P. (2001)], [Ocenasek, J., Schwarz, J. (2000)], [Ocenasek, J., Schwarz, J., Pelikan, M. (2003)].

Another approach that also has prospective to tackle this problem is purposed by [Tomoyuki, H. et. El.(2003)]. They purpose a variant of PMBGA so called distributed probabilistic model building GAs (DPMBGA**)** which uses Principle Component Analysis (PCA) to transform set of selected individual to different space where no correlation among variables exists. After generating new individuals using simple univariate distribution in that space, it transformed them back to original space. However it is reported that PCA is not always effective and is problem dependent. For detail information on DPMBGA see [Tomoyuki, H. et. El.(2003)].

Lots of interdisciplinary researches is being carried out at the moment aiming on finding a good way of calculating probability distribution. As stated by [Mühlenbein & Mahnig (2002)], It combines statistics (graphical models), Artificial intelligence (computer vision, Evolutionary algorithms, and belief propagation), statistical physics (advanced mean field method) and probabilistic logic (maximum entropy).

7. Conclusion

In this paper, we surveyed a class of optimization algorithm algorithms so called PMBGAs that uses probabilistic model of promising solution to further explore the search space so as to preserve important pattern in chromosome. We first describe motivation of emergence of PMBGA and its general frame work. We categorise different PMBGAs according to their used probability model and describe their workflow. We figure out the effectiveness and weekness of PMBGAs and conclude that bottle neck of this new technique remains same that is to find out the effective way of building a problem specific Probabilistic model.

Presented paper can be seen as an introductory text on basic PMBGAs survey. For more advanced topics and more detail information on surveyed algorithms it is recommended to see the referenced papers.

References

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning.* Pittsburgh, PA: Carnegie Mellon University (Technical Report No. CMU-CS-94-163).

Baluja, S., & Davies, S. (1997). *Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space*. In Proceedings of the 14th International Conference on Machine Learning (pp. 30--38). Morgan Kaufmann.

Brown D.F., Garmendia-Doval, A.B., McCall, J. A. W. (2001). *Markov Random Field Modelling of Royal Road Genetic Algorithms*. in Evolution Artificielle 2001, Le Creusot, France, October 2001.

De Bonet, J. S., Isbell, C. L., & Viola, P. (1997). *MIMC: Finding optima by estimating probability densities.* In Mozer, M. C., Jordan, M. I., & Petsche, T. (editors), Advances in Neural Information Processing Systems, Volume 9 (pp. 424). The MIT Press, Cambridge.

Etxeberria, R. and Larrañaga, P. (1999). *Global optimization with Bayesian networks.* In II Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization (pp. 332-339).

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization and machine learning.* Addison-Wesley.0-201-15767-5

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). *The compact genetic algorithm.* In Proceedings of the IEEE Conference on Evolutionary Computation 1998 (ICEG'98) (pp. 523-528). Piscataway, NJ: IEEE Service Centre.

Harik, G. (1999). *Linkage learning via probabilistic modeling in the ECGA.* Urbana, IL: University of Illinois Genetic Algorithms Laboratory (IlliGAL Report No. 99010).

Jues, A., Baluja, S., Sinclair, A.(1993). *The Equilibrium Genetic Algorithm and the Role of Crossover.*

Larranaga, P., Etxeberria, R., Lozano, J. A. and Pena, J. M., (1999). *Optimization by learning and simulation of Bayesian and Gaussian networks*. Technical Report, University of the Basque Country, KZAA-IK- 99-04.

Larrañaga, P., Etxeberria, R., Lozano, J. A., & Peña, J. M. (2000a). *Combinatorial optimization by learning and simulation of Bayesian networks*. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (pp. 343-352). Stanford.

Larrañaga, P., Etxeberria, R., Lozano, J. A., & Peña, J. M. (2000b). *Optimization in continuous domains by learning and simulation of Gaussian networks*. In Wu, A. S. (editor), Proceedings of the 2000 Genetic and Evolutionary Computation Conference
Workshop Program (pp. 201-204).

Lozano, J. A., Sagarna, R., Larranaga, P. (2001): *Parallel Estimation of Distribution Algorithms*. Estimation of Distribution Algorithms. A new Tool for Evolutionary Computation. P. Larranaga, J. A. Lozano (eds.). Kluwer Academic Publishers, (pp. 129-145), 2001.

Mühlenbein, H., & Paaß, G. (1996). *From recombination of genes to the estimation of*
distributions I. Binary parameters. Parallel Problem Solving from Nature, eds. Voigt, H.-M and Ebeling, W. and Rechenberg, I. and Schwefel, H.-P., LNCS 1141, Springer:Berlin, (pp. 178-187).

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

Mühlenbein, H., Mahnig, T., & Rodriguez, A. O., (1999). *Schemata, distributions, and graphical models in evolutionary optimization. Journal of Heuristics*, Volume 5 (pp.215-247).

Mühlenbein, H., & Mahnig, T. (1999a). *Convergence theory and applications of the factorized distribution algorithm. Journal of Computing and Information Technology*,
Volume 7 (pp. 19-32).

Mühlenbein, H., & Mahnig, T. (1999b). *FDA - A scalable evolutionary algorithm for the optimization of additively decomposed functions.* Evolutionary Computation 7(4), (pp.353-376).

Mühlenbein, H., & Mahnig, T. (2002). *Evolutionary Algorithms and the Boltzmann Distribution*. Foundations of Genetic Algorithms (FOGA2002).

Ocenasek, J., Schwarz, J. (2000). *The Parallel Bayesian Optimization Algorithm*, In Proceedings of the European Symposium on Computational Inteligence, Physica-Verlag, Kosice, Slovak Republic, (pp. 61-67).

Ocenasek, J., Schwarz, J., Pelikan, M. (2003). *Design of Multithreaded Estimation of Destribution Algorithms.* In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2003 (pp. 1247-1258)

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In Roy, R., Furuhashi, T., & Chawdhry, P. K. (Eds.), Advances in Soft

Computing - Engineering Design and Manufacturing (pp. 521--535). London: Springer-Verlag.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In Banzhaf, W., Daida,, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (editors). In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Volume I (pp. 525-532). Orlando, FL: Morgan Kaufmann Publishers, San Francisco, CA.

Pelikan, M., Goldberg, D. E., & Lobo, F. G. (1999). A survey of optimization by building and using probabilistic models. Urbana, IL: University of Illinois Genetic AlgorithmsLaboratory (IlliGAL Report No. 99018).

Pelikan, M., & Goldberg, D. E. (2000). Hierarchical Problem Solving by the Bayesian Optimization Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000 (pp. 267-274), Las Vegas, Nevada.

Pelikan, M., & Goldberg, D. E. (2003). Hierarchical BOA solves Ising Spin Glass and MAXSAT. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2003 (pp. 1271-1282)

Thierens, D. and Goldberg, D.E. (1993). *Mixing in genetic algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms (pp. 38-45), 1993.

Tomoyuki, H. et. El.(2003): Distributed Probabilistic Model-Building Genetic Algorithm: In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2003.