DEUM: A framework for an Estimation of Distribution Algorithm based on Markov Random Fields

Siddhartha K. Shakya

School of Computing
The Robert Gordon University
Aberdeen, UK

A thesis submitted in partial fulfilment of the
requirements of
The Robert Gordon University
for the degree of Doctor of Philosophy

April 2006

Abstract

Estimation of Distribution Algorithms (EDAs) belong to the class of population based optimisation algorithms. They are motivated by the idea of discovering and exploiting the interaction between variables in the solution. They estimate a probability distribution from population of solutions, and sample it to generate the next population. Many EDAs use probabilistic graphical modelling techniques for this purpose. In particular, directed graphical models (Bayesian networks) have been widely used in EDA.

This thesis proposes an undirected graphical model (Markov Random Field (MRF)) approach to estimate and sample the distribution in EDAs. The interaction between variables in the solution is modelled as an undirected graph and the joint probability of a solution is factorised as a Gibbs distribution. The thesis describes a model of fitness function that approximates the energy in the Gibbs distribution, and shows how this model can be fitted to a population of solutions to estimate the parameters of the MRF. The estimated MRF is then sampled to generate the next population. This approach is applied to estimation of distribution in a general framework of an EDA, called Distribution Estimation using Markov Random Fields (DEUM). The thesis then proposes several variants of DEUM using different sampling techniques and tests their performance on a range of optimisation problems. The results show that, for most of the tested problems, the DEUM algorithms significantly outperform other EDAs, both in terms of number of fitness evaluations and the quality of the solutions found by them. There are two main explanations for the success of DEUM algorithms. Firstly, DEUM builds a model of fitness function to approximate the MRF. This contrasts with other EDAs, which build a model of selected solutions. This allows DEUM to use fitness in variation part of the evolution. Secondly, DEUM exploits the temperature coefficient in the Gibbs distribution to regulate the behaviour of the algorithm. In particular, with higher temperature, the distribution is closer to being uniform and with lower temperature it concentrates near some global optima. This gives DEUM an explicit control over the convergence of the algorithm, resulting in better optimisation.

To my parents

Acknowledgements

I would like to take this opportunity and thank everyone who shared their moments with me in various walk of life.

First and foremost, I would like to thank my family: my parents Buddha Kumar Shakya and Chameli Shakya for being an unlimited source of inspiration, encouragement and support for me, my brother Bijaya (Tinku) and his wife Rakhee for being my best friends, and also Denisa Benkova, who is an integrated part of my family, for standing by me and supporting me along the way.

I would like to express my deepest gratitude and thanks to my supervisor, John McCall, for being such a great source of information and motivation. It is his invaluable advices and support that guided me to successfully complete this thesis. I am also grateful to my second supervisor, Deryck Brown, for all the interesting discussions and thoughts that he gave me along the way.

I would also like to thank all the staffs in the school of computing for making the past three and half years an enjoyable and memorable time. In particular, I would like to thank Sudha B. Reddy, Muhammed Basharu, Kefeng Zhang, Zea Syed, Daniel Fredouille, Ralf Bierig, Ratiba Kabli, Stella Asiimwe, Rahman Mukras, Selpi, Fiona Walsh, Nirmalie Wiratunga, Miki Sun, Sutanu Chakraborti, Ganesan Bathumalai, Sandy Brownlee and Stewart Massie from computing technologies centre for being such a great friends. I would also like to thank Susan Craw, Andrei Petrovski, Roger McDermott, Ines Arana, Garry Brindley, Chris Bryant and other academic staffs for their helpful advices and support. Finally, I would like to thank Kathy Deen-Smith, Marie Duncan, Ann Gardner and Dianne Godsman from the school office for their ever-smiling assistance, and Caroline Campbell, Colin Beagrie and all the technical support team for their prompt support.

I am grateful to Jose Antonio Lozano and Pedro Larrañaga for inviting me and giving me opportunity to work with them in the Intelligent System Group at the University of the Basque Country. I met a lot of good friends there: Rubén Armañanzas, Roberto Santana,

Borja Calvo, Aritz Pérez, Ramón Sagarna and Guzmán Santafé. I am particularly thankful to Roberto Santana for his valuable comments and very interesting discussions.

I would also like to thank my viva committee, Jose Antonio Lozano, Frank Herrmann, Maureen Melvin, and John McCall for agreeing to serve as the committee members and providing me with valuable comments and suggestions. Specially, I would like to thank Jose and Frank for their very useful thoughts, and interesting suggestions.

There are number of other peoples who supported me throughout the course of my research. Particularly, I would like to thank Susan Copeland, and also my cousin Minu and her husband Surendra Shrestha for their constant support.

Finally, I am grateful to the Robert Gordon University for providing the grant that allowed me to carry this research work.

Publications

Some parts of the work presented in this thesis has appared in following publications.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2006). Solving the ising spin glass problem using a bivariate eda based on markov random fields. In *proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2006)*, IEEE press, Vancouver, Canada (in press).

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2005c). Incorporating a metropolis method in a distribution estimation using markov random field algorithm. In *proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2005)*, vol. 3, 2576–2583, IEEE press, Edinburgh, UK.

Shakya, S., McCall, J. & Brown, D. (2005b). Using a Markov Network Model in a Univariate EDA: An Emperical Cost-Benefit Analysis. In *proceedings of Genetic and Evolutionary Computation COnference (GECCO2005)*, 727–734, ACM, Washington, D.C., USA.

Shakya, S., McCall, J. & Brown, D. (2005a). Estimating the distribution in an EDA. In B. Ribeiro, R.F. Albrechet, A. Dobnikar, D.W. Pearson & N.C. Steele, eds., *In proceedings of the International Conference on Adaptive and Natural computiNG Algorithms (ICANNGA 2005)*, 202–205, Springer-Verlag, Wien, Coimbra, Portugal.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2004b). Updating the probability vector using MRF technique for a Univariate EDA. In E. Onaindia & S. Staab, eds., *Proceedings of the Second Starting AI Researchers' Symposium, volume 109 of Frontiers in artificial Intelligence and Applications*, 15–25, IOS press, Valencia, Spain.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2004a). Preliminary results on Evolution without Selection. In *Proceedings of Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science (PREP 2004)*, Hertfordshire, UK.

Contents

List of Figures

List of Tables

List of Abbreviations

ADF: Additively Decomposable Functions

BEDA: Boltzmann Estimated Distribution Algorithm

BGS: Bitwise Gibbs Sampler

BMDA: Bivariate Marginal Distribution Algorithm

BOA: Bayesian Optimization Algorithm

BZTM: Bitwise Zero-Temperature Metropolis method

cGA: compact Genetic Algorithm

COMIT: Combining Optimizers with Mutual Information Trees

DAG: Directed Asyclic Graph

DEUM: Distribution Estimation Using Markov Random Field

DEUM$_d$: DEUM with direct sampling of Gibbs distribution

DEUM$_{pv}$: DEUM with probability vector approach to sampling

EBNA: Estimation of Bayesian Network Algorithm

EBNA$_{BIC}$: EBNA using Bayesian Information Criterion

EBNA$_{K2+pen}$: EBNA using $K2$ algorithm with a penalising factor

EBNA$_{PC}$: EBNA using PC algorithm

ECGA: Extended Compact Genetic Algorithm

EDA: Estimation of Distribution Algorithm

FDA: Factorised Distribution Algorithm

GEMGA: Gene Express Messy Genetic Algorithm

gpdf: generalised probability density function

GRF: Gibbs Random Field

GS: Gibbs Sampler

hBOA: hierarchical BOA

Is-DEUM: DEUM with Ising model

Is-DEUM$_g$: Is-DEUM with Gibbs Sampler

Is-DEUM$_m$: Is-DEUM with Metropolis Sampler

jgpdf: joint generalised probability density function

jpd: joint probability distribution

LDFA: Linkage Detection Factorization Algorithm

LFDA: Learning Factorised Distribution Algorithm

LLGA: Linkage Learning Genetic Algorithm

MCMC: Markov Chain Monte Carlo

MDL: Minimum Description Length

mGA: messy Genetic Algorithm

MIMIC: Mutual Information Maximization for Input Clustering

MN-EDA: Markov Network Estimation of Distribution Algorithm

MN-FDA: Markov Network Factorised Distribution Algorithm

MPM: Marginal Product Model

MRF: Markov Random Field

MWST: Maximum Weight Spanning Tree

PBIL: Population Based Incremental Learning

PGM: Probabilistic Graphical Model

PLS: Probabilistic Logic Sampling

RBGS: Repeated Bitwise Gibbs Sampler algorithm

RLD: Run Length Distribution

SVD: Singular Value Decomposition

UMDA: Univariate Marginal Distribution Algorithm

Chapter 1

Introduction

An optimisation problem is the problem to find the optimal or near optimal solution from a specified set of feasible solutions using some measure for evaluating each individual solution. An algorithm to solve such problem is called an optimisation algorithm. This thesis focuses on a class of general optimisation algorithm known as Evolutionary Algorithm (EA).

Evolutionary Algorithms are inspired by Darwin's theory of natural evolution. In nature, the better species are evolved by means of natural selection and random variation. EA follows this approach and simulates the natural selection and variation to evolve a better solution to a problem. Both selection and variation have their distinct role in EA evolution. Selection puts pressure on evolution of high quality solutions by selecting fitter solutions from a population of solutions. Variation reproduces the next generation of population based on the selected fitter solutions, and ensures the proper exploration of possible set of solutions. Three well known EAs, Genetic Algorithm (GA) (Holland, 1975), Evolution Strategy (ES) (Rechenberg, 1973) and Evolutionary Programming (EP) (Fogel, 1962), all maintain the selection and variation concept of evolution.

In a GA, two genetic operators, *crossover* and *mutation*, are used to simulate the variation. Crossover forms the new population by exchanging some parts between the selected solu-

tions. Mutation slightly modifies some parts of the newly formed solutions to introduce some genetic variation in the new population.

The traditional crossover and mutation approach of variation in GAs has been found to be limited for many optimisation problems, and therefore, most of the early research in GAs has been focused on the modification of these operators to improve GA performance. In recent years, a probabilistic approach to variation has been proposed where crossover and mutation was replaced by two other operators: *distribution estimation* and *sampling*. Distribution estimation is to estimate a probability distribution of solutions from population, and sampling is to sample the distribution to generate a new population. Algorithms using such approach to variation are called Estimation of Distribution Algorithms (EDAs) (Mühlenbein & Paaß, 1996; Larrañaga & Lozano, 2002).

EDAs have been recognised as a powerful technique for optimisation. They have been found to perform better on problems, where traditional GAs failed to give satisfactory performance (Pelikan & Goldberg, 2003).

The performance of an EDA highly depends on how well it estimates and samples the probability distribution. Lots of EDA research is focused in this area. In particular, directed graphical models (Bayesian networks) have been widely studied and are established as a useful approach to estimate and sample the distribution in EDAs. In this thesis we propose an undirected graphical model (Markov Random Field) approach to estimate and sample the distribution in EDAs.

1.1 Research Objective

The primary objective of our research is to propose and evaluate a framework of an EDA based on Markov Random Field (MRF) approach to estimating and sampling the distribution. This primary objective can be further divided into 5 parts:

1. To conduct an extensive survey of MRF in the context of EDAs

2. To study the use of solution fitness in approximating a MRF in EDAs

3. To conduct research on various ways to sample the MRF in EDAs

4. To implement a framework of an EDA using the MRF approach to estimate and sample the distribution

5. To evaluate the performance of the implemented EDA in the range of optimisation problems and compare the results with that of other EDAs

1.2 Outline of the thesis

The thesis is divided into nine chapters.

Chapter 2. Estimation of Distribution Algorithms

Chapter 2 describes Estimation of Distribution Algorithms. It starts by describing GAs: the parent algorithm of EDAs, motivates the use of the probabilistic approach to variation in the GA evolution and describes the general workflow of EDA. This chapter also presents a survey of discrete EDAs.

Chapter 3. Probabilistic Graphical Models and EDAs

Many EDAs use the concept of Probabilistic Graphical Models (PGM) to estimate and sample the distribution. Chapter 3 reviews the PGM in context of EDAs. It starts by describing two well known PGMs: Bayesian networks and Markov Random Fields, reviews different approaches to estimate and sample them and reviews their use in EDAs. More focus is given to describing MRF in EDAs as this thesis proposes an estimation and sampling technique based on MRFs.

Chapter 4. Fitness modelling approach to estimating parameters in Markov Random Fields

Chapter 4 presents the proposed *fitness modelling* approach to estimate the parameters of the MRF from the population of solutions. It describes the Hammersley-Clifford theorem that establishes the equivalence between joint probability distribution for a MRF and the Gibbs distribution. Using this theorem, a model of fitness function is derived that approximates the energy function in the Gibbs distribution in terms of the fitness of a solution. This model is called the MRF Fitness Model (MFM). A least square technique is then presented that fits MFM to the population of solution and estimates the parameters of the MRF.

Chapter 5. DEUM algorithm and a probability vector approach to sampling

Chapter 5 introduces DEUM algorithm: a general framework of an EDA using the proposed fitness modelling approach to estimate the MRF. It also proposes a probability vector approach to sample the MRF and incorporates it into the DEUM framework. The resulting DEUM is called DEUM$_{pv}$. DEUM$_{pv}$ uses a simple model of distribution that assumes no interaction between variables in the solution. This chapter also presents the experimental results on the performance of DEUM$_{pv}$, and compares them with that of other EDAs with a similar model of distribution.

Chapter 6. DEUM$_d$: direct sampling from Gibbs distribution

Chapter 6 presents DEUM$_d$: an extension of the DEUM$_{pv}$ algorithm that directly samples from the Gibbs distribution. It also presents the extensive experimental analysis on the performance of DEUM$_d$ in a wide range of optimisation problems. Further, it describes the cost benefit analysis of using the MRF approach to probabilistic modelling in DEUM algorithms. Finally, it describes an important property of DEUM algorithms, that is, their

use of fitness to model the distribution and therefore their ability to perform optimisation without using any *explicit* selection operators.

Chapter 7. Is-DEUM: A step towards multivariate DEUM

Chapter 7 extends DEUM algorithm to use a bivariate model of probability distribution. The resulting DEUM is called Is-DEUM. This chapter presents two variants of Is-DEUM that use different sampling technique, 1) Is-DEUM$_m$ with Metropolis sampling and 2) Is-DEUM$_g$ with Gibbs sampling, and apply them to a well known Ising spin glass problem. Further, it presents the experimental results on the performance of these algorithms and compares them with that of other EDAs.

Chapter 8. Future Work

Chapter 8 presents the future directions to this research study. It outlines some of the immediate future works that may be of significance to the development of more effective DEUM algorithms.

Chapter 9. Conclusion

Chapter 9 highlights some of the important contributions made by our research study and concludes the thesis.

Chapter 2

Estimation of Distribution Algorithms

Since its introduction about a decade ago, the Estimation of Distribution Algorithm (EDA) has established itself as a promising area of research for the evolutionary algorithm community. A growing number of EDA papers are being published each year including a number of recent PhD theses (Pelikan, 2002; Bosman, 2003; Santana, 2003b). The EDA was first proposed as a modification to the traditional GA. Since then, it has become a discipline of its own within the field of evolutionary computation.

The aim of this chapter is to give a general introduction to EDAs. This chapter is in two parts. The first part describes the motivation behind emergence of EDA. It starts by introducing the GA: the parent algorithm of EDA. It then presents the motivation to the emergence of the EDAs and describes the general framework of EDAs.

The second part reviews the EDA literature. Following Larrañaga *et al.* (1999) and Pelikan *et al.* (1999b), EDAs are categorised according to the order of interaction between variables taken into account by their model of distribution, and their workflow is briefly described. Finally, this chapter concludes by identifying the need for further research on probabilistic modelling and sampling in EDAs.

2.1 From Genetic Algorithms to the Estimation of Distribution Algorithms

Genetic Algorithms (GAs) (Holland, 1975) are a class of optimisation algorithm inspired by Darwin's theory of evolution. A GA encodes a solution as a string of symbols, or *chromosome*, and evolves a population of chromosomes to obtain better solutions. GAs have been successfully applied to solve a wide variety of problems from different application areas such as engineering, science and business (Goldberg, 1989; Mitchell, 1997), and are a subject of active research in the field of computational intelligence.

2.1.1 Genetic Algorithm

In a GA, a solution $x = \{x_1, x_2, .., x_n\}$ is encoded as a set of values, x_i. The string of values is known as a chromosome. Depending upon the problem type, a bit, real or integer string can be used for the chromosome. In this thesis, we will mainly be concerned with bit-string chromosomes. The number of characters in the string is known as *chromosome length* and will be defined by n. Each solution, x, has an extra value associated with it known as its fitness value, which measures the goodness of that solution. The fitness value is calculated from given optimisation criteria that are modelled in the form of a function known as *fitness function* $f(x)$. For example, Figure 2.1 shows a bit-string chromosome with chromosome length $n = 6$. Here, the fitness function is simply the sum of all the bits in the chromosome [1].

The set of all possible solutions is known as the *search space*. For the 6 bit long chromosome shown in Figure 2.1, the search space consist of 2^6 solutions.

The GA starts by initialising a population of solutions known as the *parent population*. The main iteration then starts by performing *selection*, *crossover* and *mutation* operations, and forms a *child population* that replaces the parent population. This process is then iterated until some termination criteria are satisfied (see Figure 2.2). We will now consider

[1] In literature, this function is also known as *Onemax function*(Mühlenbein & Paaß, 1996)

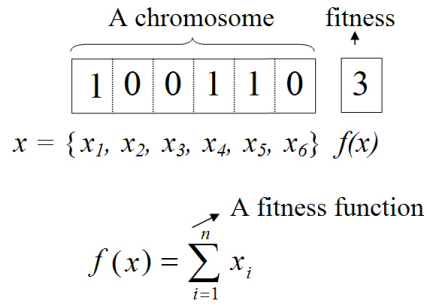


Figure 2.1: A 6 bit long chromosome and its fitness function

the GA iteration in more detail.

Simple Genetic Algorithm

1. Generate initial population of solutions P of size M
2. Form a breeding pool by selecting N promising solutions from parent population P, where $N \leq M$
3. Perform crossover and mutation on breeding pool to get a new population (also known as child population)
4. Replace parent population by child population and Go to step 2 if termination criteria are not satisfied.

Figure 2.2: The pseudo-code of the Simple Genetic algorithm

Generation of initial population (parent population) P is done usually by choosing each solution x at random. However, the output solutions of another search algorithm could also be used as the initial population. Sometimes, the initial population is seeded with solutions that are not random.

The selection operator is then applied to the population. The Selection operator selects a set of solutions, D, from the parent population, P, according to some selection criteria. Different selection methods can be used depending on the design of the algorithm. These methods can be categorised into two groups 1) *Proportional selection* and 2) *Ordinal selection.*

In proportional selection, first, the selection probability for each individual in the current population is determined, and then sampled to make the breeding pool. The *Fitness proportionate selection* (also known as *Roulette wheel selection*) (Goldberg, 1989) and *Boltzmann selection* (de la Maza & Tidor, 1993) fall in this category.

In fitness proportionate selection, the selection probability $p^s(x)$ for a solution x is determined as

$$p^s(x) = \frac{f(x)}{\sum_{y \in P} f(y)} \tag{2.1}$$

In Boltzmann selection, the selection probability $p^s(x)$ for a solution x is determined as

$$p^s(x) = \frac{e^{f(x)/T}}{\sum_{y \in P} e^{f(y)/T}} \tag{2.2}$$

Here, T is a parameter for the selection known as *temperature*.

In ordinal selection, selection probability is not calculated from the numerical value of the fitness function. Instead the selection decision is based on the ranked order of fitness values. Some of the popular ordinal selection methods include *tournament selection*, and *truncation selection* (Goldberg, 1989; Mitchell *et al.*, 1994; Davis, 1991).

In a typical tournament selection, the fittest out of two (or more) randomly chosen chromosomes from parent population is selected. In truncation selection, the N fittest solutions from parent population is selected at once.

However, the general motive behind all selection methods is the same, namely to provide a selection pressure in favour of better solution. As such, the selection process models the idea of survival of the fittest.

The crossover operator exchanges some of the *partial solution* (substring) between subset of the selected promising solutions. Different crossover operators can be used for this purpose such as *one point crossover*, *two point crossover* and *uniform crossover* (Mitchell *et al.*, 1994; Davis, 1991). Crossover occurs with a predefined probability known

as *crossover probability*. Typically the crossover probability is greater than 50%. As an example, Figure 2.3 shows one point crossover applied to two solutions resulting in two offspring. The crossover point is usually chosen at random.

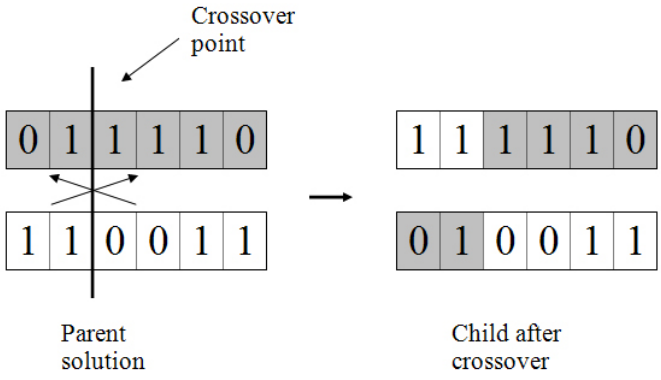


Figure 2.3: One point crossover between two solutions resulting in two offspring

The mutation operator is then applied to the resulting set of solutions. The mutation operator models the random genetic variation between parent and offspring. In optimisation, the mutation operator helps to explore other parts of the solution space that are not attainable through crossover alone. It randomly changes the value of a part of the solution to another possible value. Mutation also occurs with some probability. However, the probability is usually so small that very little change is expected to occur. Figure 2.4 shows a typical single bit mutation occurring in a solution.

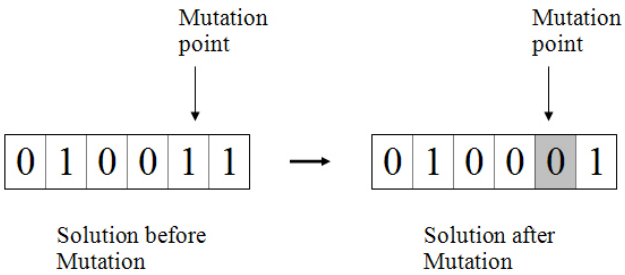


Figure 2.4: A single bit mutation

The new child population after mutation replaces the old parent population, and, if the termination criteria are not met, the next GA iteration executes. Some of the common

termination criteria are to terminate after fixed number of iterations or to terminate when a sufficiently good solution is found.

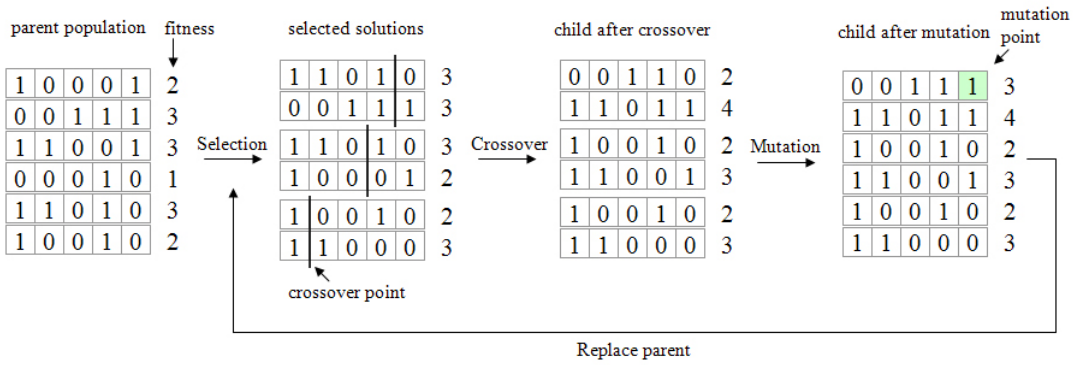


Figure 2.5: A simulation of a simple GA iteration

Figure 2.5 shows a single iteration of a GA to maximise the Onemax function. We can see how selection, crossover and mutation interact together in the evolution of child population.

2.1.1.1 Selection and Variation in GA

As shown in Figure 2.6, the process of evolution in GAs, can be seen as the combination of two processes; (1) *Selection* and (2) *Variation*. Selection drives evolution towards better solutions by giving a high pressure to the selection of high-quality solutions. Crossover and mutation together form the variation operator which helps to explore the possible space of the candidate solutions.

In a GA, the crossover operator *implicitly* recombines partial solutions from the selected set of *good* solutions. The mutation operator slightly distracts the recombined solutions to explore its immediate neighbours. As the probability of mutation is very low, for most of the GA, the main source of variation remains the crossover operator.

The key contribution of variation is to introduce novel solutions and therefore help to maintain the diversity in the population. This prevents the population from converging too fast to a premature solution and further allows an evolutionary algorithm to explore

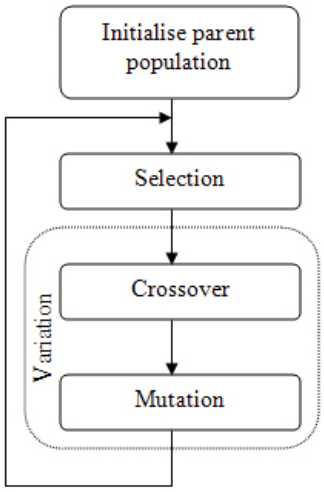


Figure 2.6: GA evolution in terms of selection and variation

more of the search space. The selection and variation processes together form the basis for the better evolution.

2.1.1.2 Linkage and the GA variation

In many optimisation problems, the variables in the solution interact with each other in order to have a positive effect in the fitness of the solution. An example is the Plateau problem shown in (2.3) (Mühlenbein, 1994), where the group of 3 adjacent bits interacts with each other to have a positive contribution to the fitness.

$$f(x) = \sum_{i=1}^{m} g(x_{3i-2}, x_{3i-1}, x_{3i}) \tag{2.3}$$

where,

$$g(x_1, x_2, x_3) = \begin{cases} 1 & \text{if} \quad x_1 = 1 \quad \text{and} \quad x_2 = 1 \quad \text{and} \quad x_3 = 1 \\ 0 & \text{otherwise} \end{cases}$$

In Onemax problem (Figure 2.1), however, each bits individually contributes to the fitness function. This means there is no interaction between variables in the problem.

We use term *linkage* to refer to the interaction between variables in a solution. Although,

for the above described problems, the linkage is known in advance, for most of the real world problem this information is not known. In order to successfully optimise a problem, it is desirable to have the knowledge of the linkage between variables. This can then be used to direct the search more efficiently.

Now let us recall the variation in GAs. Variation is intended to recombine good solutions in the population by means of crossover and mutation operators, assuming that this will reproduce even better solutions. However, neither crossover, nor mutation operator tries to learn and exploit the linkage. Instead, they randomly choose the crossover and mutation point. This random nature of crossover and mutation operators may sometime disrupt the values of the interacting variables having positive effect in the fitness. This may either lead the algorithm to take more computation time before converging to a good solution, or lead the algorithm to converge to a sub-optimal or poor quality solution.

Much research in GAs has been focused on how to discover and exploit linkage. This work can be categorised into two basic approaches.

The first approach is based on changing the representation of the problem. The idea is to manipulate the string representation of solutions to prevent disruption of the values of the interacting variables. Different reordering and mapping techniques have been introduced for this purpose. The messy Genetic Algorithm (mGA) (Goldberg *et al.*, 1989), the Gene Express Messy Genetic Algorithm (GEMGA) (Kargupta, 1996), and the Linkage Learning Genetic Algorithm (LLGA) (Harik, 1997) fall in this category.

The second approach is based on changing the variation process. The idea is to learn and exploit the linkage by estimating a distribution from the population and sampling it to generate the child population. Here, the traditional crossover and mutation approach to variation is completely replaced by estimating and sampling a probability distribution (see Figure 2.7). Next section describes this approach in more detail.

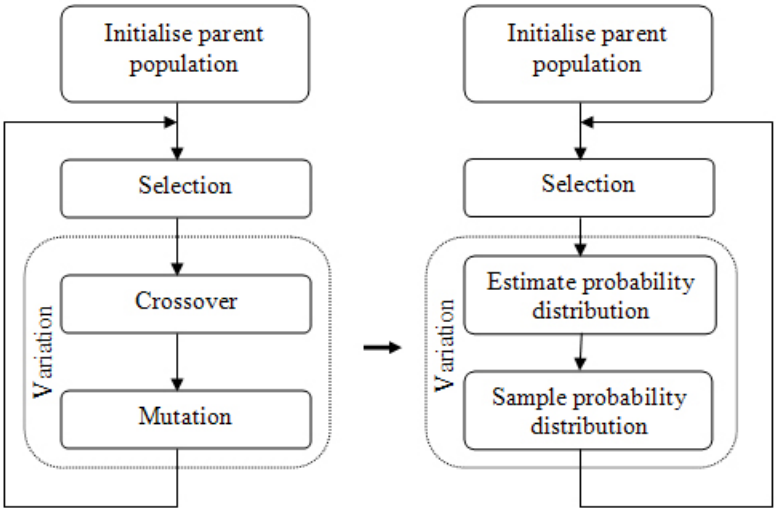


Figure 2.7: From crossover and mutation approach of variation to probabilistic approach of variation

2.1.2 Estimation of distribution and sampling: An alternative approach to variation

In statistics, distribution of a data (or dataset) is the probability of its observed (or expected) occurrence in the database. Therefore, sometimes, distribution is referred to as probability distribution. The estimation of distribution is the task of calculating the probability distribution of the data in the database. In the case of GAs, generally, the estimation of distribution, is to approximate the probability distribution of the good solutions in the solution space given only the small subset of it. Once the probability distribution is estimated, it can then be sampled to generate other solutions which are *close* to the given set of good solutions. In order to make this concept clearer, let us give a simple example of estimation of distribution and sampling. However, before doing that, we need to introduce some notations used here after.

2.1.2.1 Notation

We follow the approach taken by Larrañaga *et al.* (1999).

Let X_i be a random variable, and x_i be one of its possible values. We use $\rho(X_i = x_i)$ (or simply $\rho(x_i)$) to represent the **generalised probability density function** (**gpdf**) over the point x_i. Now let $X = \{X_1, X_2, .., X_n\}$ be a vector of n random variables, and $x = \{x_1, x_2, .., x_n\}$ be a vector of values taken by each variable of vector X, $\rho(X = x)$ (or simply $\rho(x)$) 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 $\rho(X_i = x_i|X_j = x_j)$ (or simply $\rho(x_i|x_j)$).

If the problem domain is discrete, i.e., if each X_i is a discrete random variable with a finite set of values, $\rho(X_i = x_i) \equiv p(X_i = x_i)$ (or simply $p(x_i)$) is the **univariate marginal distribution** of the variable X_i. If all the variables in X are discrete then $\rho(X = x) \equiv p(X = x)$ (or simply $p(x)$) will be the **joint probability distribution** (**jpd**). Similarly, the **conditional probability** that X_i will take value x_i given value x_j for the variable X_j can be denoted as $\rho(X_i = x_i|X_j = x_j) = p(X_i = x_i|X_j = x_j)$ (or simply $p(x_i|x_j)$).

Let X_S be a sub-vector of X, and x_S be a possible set of values taken by X_S, then $p(X_S = x_S)$ (or simply $p(x_S)$) is the **marginal distribution** of the set X_S. Note that univariate marginal distribution is a simple case of marginal distribution, where sub-vector consist of a 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 $p(X_A = x_A|X_B = x_B)$ (or simply $p(x_A|x_B)$) denotes the conditional probability of $X_A = x_A$ given $X_B = x_B$ and can be defined as

$$p(x_A|x_B) = \frac{p(x_A, x_B)}{p(x_B)} \tag{2.4}$$

Here, $p(x_A, x_B)$ is the joint probability of subsets $X_A = x_A$ and $X_B = x_B$.

The factorisation of the jpd, $p(x)$, then follows

$$p(x) = p(x_1|x_2, .., x_n)p(x_2|x_3, .., x_n)...p(x_{n-1}|x_n)p(x_n) \tag{2.5}$$

(2.5) is also known as the *chain rule* of probability distribution.

2.1.2.2 An example of variation by estimation and sampling of a probability distribution

We now regard a solution $x = \{x_1, x_2, .., x_n\}$ as a set of values taken by a set of random variables $X = \{X_1, X_2, .., X_n\}$ where $x_i \in \{0, 1\}$. Then, the estimation of distribution is to approximate the jpd $p(x) = p(x_1, x_2, ..., x_n)$, from the population of solutions P. In general, calculation of $p(x)$ involves calculation of probability for all 2^n combinations of x and therefore is not a feasible approach. However, depending on the linkage, $p(x)$ can be often factorised in terms of calculable marginal (or conditional) probability of its interacting variables. Therefore, in order to estimate $p(x)$, we need to 1) learn the linkage and 2) estimate the marginal (or conditional) probabilities of the interacting variables. Let us follow both of these tasks in a simple example.

Learn the linkage between variables : In general, given a set of solutions, a range of statistical techniques can be used to learn the linkage between variables. They will be reviewed in chapter 3. For our simple example, let's assume that the linkage is known in advance, where each variable, $X_i \in X$, only interacts with itself and does not interacts with other variables. For such linkage, the formulation of jpd $p(x)$ from (2.5), can be factorised in terms of univariate marginal probabilities of its individual variables in X, $p(x_i)$ [2], as

$$p(x) = \prod_{i=1}^{n} p(x_i) \tag{2.6}$$

Estimate probabilities: Now that we have got a factorised model for $p(x)$, the next task is to estimate the probabilities of the interacting variables. There are various ways to do so. In our simple example, given a set of good solutions, D, from the population P, the univariate marginal probability of x_i being 1, $p(x_i = 1)$, can be estimated by dividing the number of solutions in D having 1 in the i^{th} position by the total number of solution,

[2]The task of factorising $p(x)$ is often known as **probabilistic modelling** and the factorised model itself is known as the **probabilistic model** of $p(x)$. The estimation of distribution is, therefore, sometimes referred to as a task of **estimating (or building) a probabilistic model**.

N, in D.

$$p(x_i = 1) = \frac{1}{N} \sum_{x \in D, x_i = 1} 1 \tag{2.7}$$

$p(x_i = 0)$ can also be calculated in similar way.

Sample distribution : Once we estimate $p(x)$, (i.e. all the marginal probabilities), we can then sample it to generate other instances of X. In our simple example, once we estimate $p(x_i = 1)$, we can sample it to generate an x_i for a child solution as

$$x_i = \begin{cases} 1, & \text{if} \quad rand(0,1) \leq p(x_i = 1) \\ 0, & \text{otherwise} \end{cases}$$

Here, $rand(0,1)$ is a random number between 0 and 1. Repeating this for each $i \in \{1, 2, .., n\}$ will give us a child solution. This process can be repeated until we generate a complete population, which can then be used to replace the parent population P.

Figure 2.8 shows our simple example of estimation of distribution and sampling for Onemax problem.

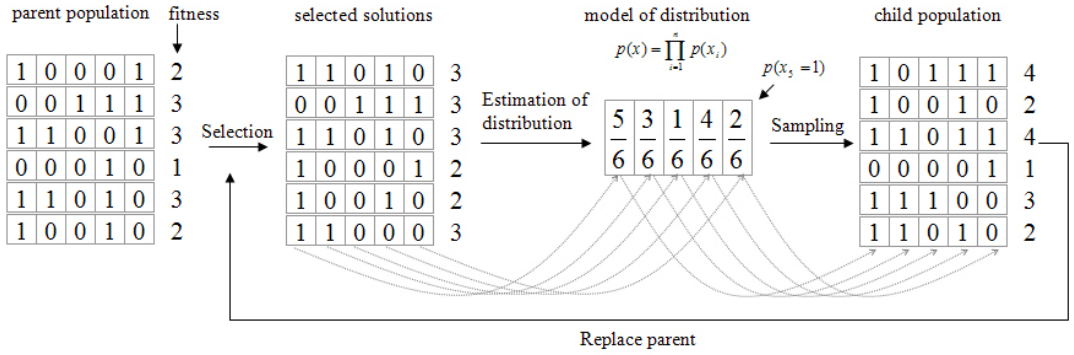


Figure 2.8: A simulation of variation by means of estimation of distribution and sampling

As we can see, after selection, the crossover is not applied as would have been in GAs. Instead, univariate marginal probabilities, $p(x_i)$, are calculated from the selected set of solutions and sampled to generate the child population. [3]

Although, the example presented here is one of the simplest examples of estimation of

[3] In Figure 2.8, only $p(x_i = 1)$ are calculated and sampled, however $p(x_i = 0) = 1 - p(x_i = 1)$ can also be calculated and sampled in the similar way

distribution and sampling, it clearly distinguishes the probabilistic approach to variation with the traditional crossover and mutation approach.

2.1.3 Estimation of Distribution Algorithm

The class of algorithms following the estimation of distribution and sampling approach to variation is known as Estimation of Distribution Algorithms (EDAs) (Mühlenbein & Paaß, 1996; Larrañaga & Lozano, 2002). EDA is a developing area in the field of evolutionary algorithms. A wide variety of EDAs using different techniques to estimate and sample the probability distribution have been proposed and are the subject of active research among the evolutionary algorithm community.

2.1.3.1 Motivation

The first and the foremost motivation, as we have stated earlier, behind the emergence of EDAs is to identify and exploit the linkage between variables in the solution in order to assist the evolution. However, there are two more factors, as noted by Larrañaga & Lozano (2002), that also motivated the researchers towards a probabilistic approach to evolution.

Firstly, the performance of a GA depends on a choice of parameters and design factors, such as different crossover and mutation operators, probabilities associated with crossover and mutation, population size and so on. Therefore, choosing an effective configuration for a GA can become an optimisation problem in itself (Grefenstette, 1986). One of the motivation behind the EDA was to minimise (or at least make it easy to set) the parameters for the algorithm.

Secondly, the theoretical analysis of the GA is an extremely difficult task. Several theories have been proposed to explain the GA evolution however a convincing theory is yet to be developed. Another motivation behind the emergence of EDA is to achieve better and more rigorous theoretical analysis of the evolutionary process (Larrañaga & Lozano, 2002).

2.1.3.2 EDA workflow

As do traditional GAs, all EDAs start by generating an initial population P consisting of M solutions. The set D consisting of N solutions is then selected from P according to some criteria. The estimation of distribution is then carried out from set D and used to sample offspring to replace P. This iteration continuous until the termination criteria is satisfied. The workflow of a general EDA is shown in Figure 2.9

Estimation of Distribution Algorithm

1. Generate initial (parent) population P of size M
2. Select set D from P consisting of N solutions, where $N <= M$
3. Estimate the probability distribution $p(x)$ from D
4. Sample $p(x)$ to generate offspring, and replace parent
5. Go to step 2 until termination criteria are meet

Figure 2.9: The pseudo-code of the general Estimation of Distribution Algorithm

As we can see from the pseudo-code (and also from the simulation), the task of estimating the distribution and sampling lies in the very heart of the EDA workflow. The success of an EDA heavily depends on these two tasks. The task of estimating the distribution further depends on two more factors: 1) accuracy of the linkage 2) accuracy of estimation of probabilities in the model of distribution. The general estimation of distribution algorithm, therefore, can be seen as the combination of the three processes:

1. Linkage estimation
2. Probabilities estimation
3. Sampling of the distribution

We will devote a whole chapter (chapter 3) to describing different approaches to each

of these processes in EDAs. In next section, however, we present a survey of the EDA literature.

2.2 Survey of discrete EDAs

In this section, we present a survey of *discrete* EDAs. The aim here is to categorise EDAs according to the type of linkage used by their model of distribution (similar approach to EDA categorisation can also be found in (Larrañaga *et al.*, 1999; Larrañaga & Lozano, 2002; Pelikan *et al.*, 1999b)), describe the motivation behind their emergence, and briefly describe their workflow and any related works.

Depending upon type of linkage used by their model of distribution, EDAs can be divided into three groups: Univariate, Bivariate and Multivariate.

2.2.1 Univariate EDAs

EDAs in this category assume each variable to be independent, i.e. do not consider any interactions among variables in the solution. Therefore, the joint probability distribution, $p(x)$, becomes simply the product of univariate marginal probabilities of all variables in the solution. Figure 2.10 shows the graphical representation of the linkage used by these algorithms where each node represents a variable in the solution.

$$p(x) = \prod_{i=1}^{n} p(x_i) \tag{2.8}$$

The EDA presented in Figure 2.8 falls into this category.

Due to the simplicity of the model of distribution used, the algorithms in this category are computationally inexpensive, and perform well on problems with no significant interaction among variables. However, these algorithms tend to fail on the problems, where higher order interactions among variables exist. Population Based Incremental Learning (PBIL) (Baluja, 1994), Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein

& Paaß, 1996), and Compact Genetic Algorithm (cGA) (Harik *et al.*, 1999) all use univariate models of probability distribution.

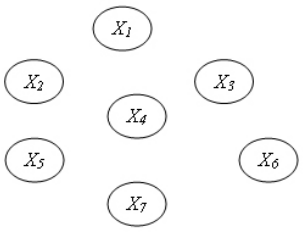


Figure 2.10: Graphical representation of linkage with no interaction between variables

Population Based Incremental Learning (PBIL)

One of the earliest works in the EDA field was Population Based Incremental Learning (PBIL) algorithm proposed by Baluja (1994). PBIL was motivated by the idea of combining GAs with *Competitive Learning* used in training *Artificial Neural Networks*. PBIL starts with initialisation of a probability vector. In each iteration, it updates and samples the probability vector to generate new solutions. Pseudo-code for general PBIL algorithm is shown in Figure 2.11.

λ and μ are the parameters of the PBIL known as *learning rate* and *mutation shift* respectively. Several different variants of PBIL have been proposed with some extension in the method of updating probability vector. One of them is to update the probability vector towards the single best solution rather than to update towards N best solutions. Another possibility is to update not only towards the best solution but also to move away from the worst solution. Some theoretical work and experimental results on PBIL can be found in (Baluja, 1994, 1995; Höhfeld & Rudolph, 1997; Berny, 2000; González *et al.*, 2001). Also some applications of PBIL can be found in (Galić & Höhfeld, 1996; Inza *et al.*, 2001c; Petrovski *et al.*, 2006).

Population Based Incremental Learning (PBIL)

1. Initialise a probability vector $p = \{p_1, p_2, ..., p_n\}$ with 0.5 at each position. Here, each p_i represents the probability of 1 for the i^{th} position in the solution.
2. Generate a population P of M solutions by sampling probabilities in p
3. Select set D from P consisting of N promising solutions, where $N <= M$
4. Estimate univariate marginal probabilities $p(x_i)$ for each x_i
5. For each i, update p_i using $p_i = p_i + \lambda(p(x_i) - p_i)$
6. For each i, if mutation condition passed, mutate p_i using
   $p_i = p_i(1 - \mu) + random(0 \; or \; 1)\mu$
7. Go to step 2 until termination criteria are meet

Figure 2.11: The pseudo-code of the Population Based Incremental Learning

Univariate Marginal Distribution Algorithm (UMDA)

Univariate Marginal Distribution Algorithm (UMDA) was proposed by Mühlenbein & Paaß (1996), and is one of the early works in the field of EDAs. Different variants of UMDA have been proposed, and the mathematical analysis of their workflows has been carried out (Mühlenbein, 1998; Mühlenbein *et al.*, 1999; González *et al.*, 2003). Pseudo-code for general UMDA algorithm is shown in Figure 2.12.

Univariate Marginal Distribution Algorithm (UMDA)

1. Generate a population P of M solutions
2. Select set D from P consisting of N promising solutions, where $N <= M$
3. Estimate univariate marginal probabilities $p(x_i)$ from D for each x_i
4. Sample $p(x_i)$ to generate M new individual and replace P
5. Go to step 2 until termination criteria are meet

Figure 2.12: The pseudo-code of the Univariate Marginal Distribution Algorithm

The simulation presented in previous section (Figure 2.8) can be seen as the simulation of UMDA for Onemax problem. Note that, UMDA can be seen as a variant of PBIL when $\lambda = 1$ and $\mu = 0$. The theoretical work on convergence of UMDA can be found in (Mühlenbein & Paaß, 1996; Mühlenbein, 1998; Mühlenbein *et al.*, 1999; Mühlenbein & Mahnig, 1999a; González *et al.*, 2003).

compact Genetic Algorithm (cGA)

Compact Genetic Algorithm (cGA) (Harik *et al.*, 1999) is motivated by the previous work done in the field of random walk model (Harik *et al.*, 1997), and also assumes no interaction between variables in solution.

cGA also maintains probability vector as in PBIL. However, unlike PBIL, cGA samples only two solutions at a time, compares their fitness, and uses allele value of the winning solution (i.e solution with the highest fitness) to update the probability vector, leaving the probability vector unchanged in the position, where winning and loosing solution contains the same value. The process continues until the probability vector converges. Pseudo-code for general cGA algorithm is shown in Figure 2.13.

Compact Genetic Algorithm (cGA)

1. Initialise a probability vector $p = \{p_1, p_2, ..., p_n\}$ with 0.5 at each position. Here, each p_i represents the probability of 1 for the i^{th} position in the solution.
2. Generate two solutions by sampling probabilities in p, and label *winner* to the fittest and *loser* to the less fit solution.
3. For each i, update p_i using following rule
   if *winner[i]* $\neq$ *loser[i]* then
   if *winner[i]*=1 then $p_i = p_i + \lambda$
   else $p_i = p_i - \lambda$
4. Go to step 2 until termination criteria are meet

Figure 2.13: The pseudo-code of the Compact Genetic Algorithm

In Harik *et al.* (1999), the λ is defined as $1/M$, where M is the parameter of the algorithm. Some recent work on parallelisation of cGA can be found in Lobo *et al.* (2005) and a mathematical analysis on the performance of cGA can be found in Droste (2005).

2.2.2 Bivariate EDAs

Algorithms in this category consider pair-wise interactions among variables in the solution. Therefore, in contrast with univariate case, the probability model contains factors involving the conditional probability of pairs of interacting variables. Obviously, in comparison to univariate EDAs, this class of algorithms performs better in problems, where pair-wise interaction among variable exists. However, it fails in problems with multiple variable interactions. Mutual Information Maximization for Input Clustering (MIMIC) (de Bonet *et al.*, 1997), Combining Optimizers with Mutual Information Trees (COMIT) (Baluja & Davies, 1997) and Bivariate Marginal Distribution Algorithm (BMDA)(Pelikan & Mühlenbein, 1999) all use bivariate models of probability distribution.

A graphical representation of the linkage used by these algorithms are shown in Figure 2.14.

Mutual Information Maximization for input clustering (MIMIC)

Mutual Information Maximization for input clustering (MIMIC) proposed by (de Bonet *et al.*, 1997) uses a chain model of probability distribution which can be written as:

$$p(x) = p(x_{\pi_1}|x_{\pi_2})p(x_{\pi_2}|x_{\pi_3})....p(x_{\pi_{n-2}}|x_{\pi_{n-1}})p(x_{\pi_n}) \tag{2.9}$$

Here, $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$ is a permutation of the numbers $\{1, .., n\}$ used as an ordering for the pair wise conditional probabilities. At each iteration, algorithm first tries to learn the linkage, i.e. find π such that the model of $p(x)$ in (2.9) approximates the model in (2.5) as closely as possible. *Kullback-Leibler divergence* (Kullback & Leibler, 1951) is used to measure the identicality between these two models. In most cases, searching over

all possible permutations of π is not feasible. MIMIC uses a greedy algorithm to find a π, which however, does not always gives accurate model. Once the π is learnt, MIMIC then estimates the pair wise conditional probabilities and sample them to get next set of solutions.

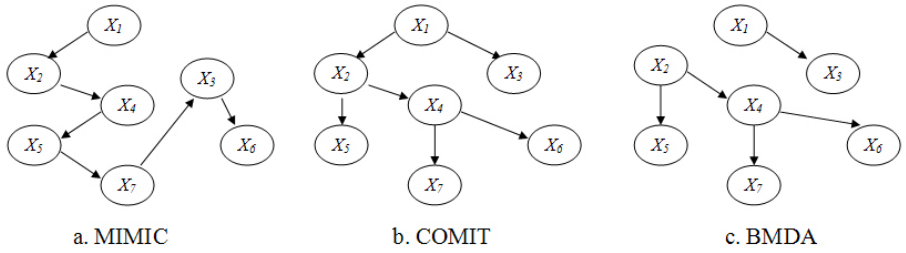


Figure 2.14: Graphical representation of linkage with pair-wise interaction between variables

Combining Optimizers with Mutual Information Trees (COMIT)

Combining Optimizers with Mutual Information Trees (COMIT) proposed by Baluja & Davies (1997, 1998) also uses pair-wise interaction among variables. The model of distribution used by COMIT can be written as

$$p(x) = \prod_{i=1}^{n} p(x_i|x_j) \tag{2.10}$$

Where, X_j is known as parent of X_i and X_i is known as a child of X_j. This model follows a tree structure, i.e. no children (or *grand-children*) of a variable can be its parent.

This model is more general than the chain model used by MIMIC as two or more variables can have a common parent. COMIT uses the *Maximum Weight Spanning Tree (MWST)* (Chow & Liu, 1968) algorithm to construct the model. However, the tree model has its limitations, i.e. excluding the root variable, every other variable should have a parent. In (2.10), if X_i is the root variable then $p(x_i|x_j)$ is generalised to univariate marginal probability, $p(x_i)$.

Bivariate Marginal Distribution Algorithm (BMDA)

The Bivariate Marginal Distribution Algorithm (BMDA) was proposed by (Pelikan & Mühlenbein, 1999) as an extension to UMDA. The model of distribution used by BMDA can be seen as an extension to the COMIT model and can be written as

$$p(x) = \prod_{X_k \in Y} p(x_k) \prod_{x_i \in \{X \setminus Y\}} p(x_i|x_j) \tag{2.11}$$

where, $Y \subseteq X$ is the set of root variables. Unlike COMIT, this model does not require every variable to have a parent variable, i.e. can have more than one root variable. BMDA, therefore, is a more generalised algorithm in this class and can cover both univariate interaction as well as bivariate interaction among variables. In order to detect the interaction between two variables, BMDA uses *Pearson's chi-square statistics*.

2.2.3 Multivariate EDAs

Any algorithms considering interaction between variables of order more than two can be placed in this class. The model of probability distribution obviously becomes more complex than the one used by univariate and bivariate EDAs. The complexity of constructing such model increases exponentially to the order of interaction making it infeasible to search through all possible models. Extended Compact Genetic Algorithm (ECGA) (Harik, 1999), Factorised Distribution Algorithm (FDA) (Mühlenbein *et al.*, 1999), Bayesian Optimization algorithm (BOA) (Pelikan *et al.*, 1999a), Learning Factorised Distribution Algorithm (LFDA) (Mühlenbein & Mahnig, 1999b), Estimation of Bayesian Network Algorithm (EBNA) (Etxeberria & Larrañaga, 1999) all use multivariate model of probability distribution. In recent years two more algorithm have been proposed by Santana (2003a, 2005) falling in this category. They are Markov Network Factorised Distribution Algorithm (MN-FDA) and Markov Network Estimation of Distribution Algorithm (MN-EDA).

Figure 2.15 shows the graphical representation of linkages used by these algorithms.

Extended Compact Genetic Algorithm (ECGA)

Extended Compact Genetic Algorithm (ECGA) has been proposed by Harik (1999) as an extension to cGA. The model of distribution used in ECGA, and so called *Marginal Product Model (MPM)*, is distinct from other previously described models as they only consider the marginal probabilities and do not include conditional probabilities. Also it assumes that there is no overlapping interaction between variables i.e. a variable appearing in a set of interacting variables cannot appear in another set. MPM used in ECGA can be defined as

$$p(x) = \prod_{c \in m} p(x_c) \tag{2.12}$$

Where, m is the set of disjoint subsets in n and $p(x_c)$ is the marginal probability of set of variables x_c in the subset c. ECGA uses *Minimum Description Length (MDL)* metric (Rissanen, 1978) to measure the goodness of fit of the MPM model and uses greedy search to find a good MPM model.

If the constructed model is correct, and the problem domain does not contain overlapping interactions, then ECGA works well. However, not all real life problems are of this kind and can often have overlapping interactions. In such case, ECGA can fail. Some recent works of Sastry *et al.* (2004); Sastry & Goldberg (2004) has applied efficiency enhancement techniques, such as *Building blocks-wise crossover* and *Building blocks-wise mutation*, to ECGA.

Factorised Distribution Algorithm (FDA)

The Factorised Distribution Algorithm (FDA) was proposed by Mühlenbein *et al.* (1999) as an extension to UMDA. In order to estimate the distribution, FDA requires the linkage between variables to satisfy the *running intersection property* [4] (Lauritzen, 1996). A jpd, $p(x)$, for such linkage can be factorised in terms of conditional probabilities between sets of interacting variables. In general, FDA requires the linkage information in advance, which

[4] see chapter 3 for more on running intersection property

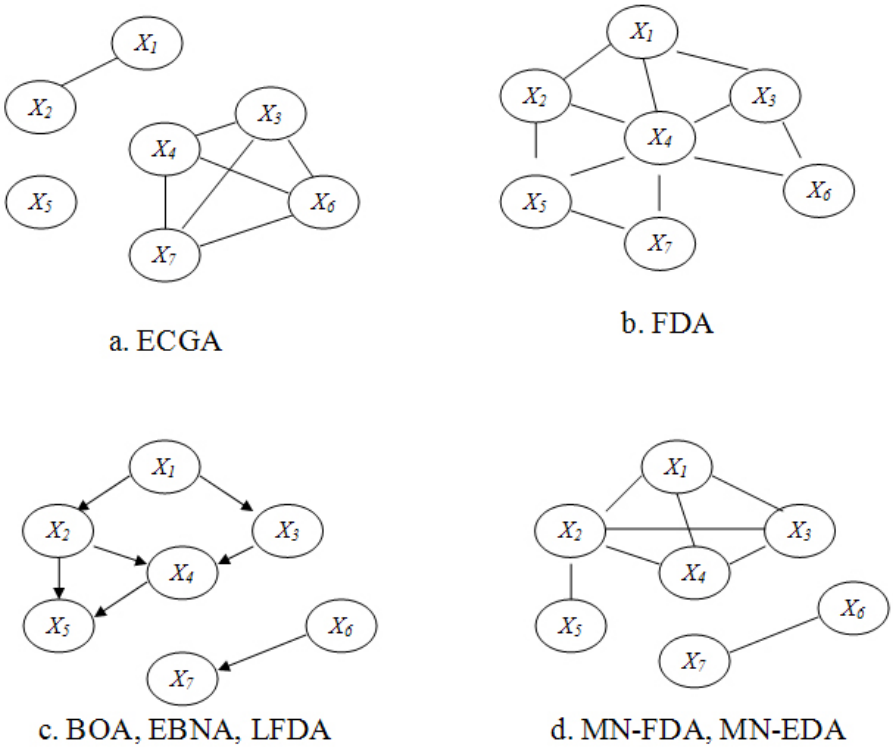


Figure 2.15: Graphical representation of linkage with multivariate interaction between variables

may not be available in a real world problem. However, if the linkage satisfies running intersection property, FDA has been shown to solve *GA hard* problems very effectively. An extension of FDA called the Learning Factorised Distribution Algorithm (LFDA), has been proposed by Mühlenbein & Mahnig (1999b). LFDA does not require advance knowledge of linkage. Some theoretical analysis on FDA can be found in (Mühlenbein *et al.*, 1999; Mühlenbein & Mahnig, 1999a,b, 2001a; Zhang & Mühlenbein, 1999; Zhang & Muehlenbein, 2004).

Bayesian Optimization algorithm (BOA)

Bayesian Optimization algorithm (BOA) proposed by Pelikan *et al.* (1999a) models the jpd, $p(x)$, in terms of a set of conditional probabilities as

$$p(x) = \prod_{i=1}^{n} p(x_i|\Pi_i) \tag{2.13}$$

Where, Π_i is reffered to as parents of X_i, and X_i is referred to as child of Π_i. Π_i is a set of variables having conditional interaction with X_i. Also no variable in Π_i can have X_i or any children of X_i as their parent. This last condition is known as a Directed Acyclic Graph (DAG) principle.

The model of distribution shown in (2.13) is known as Bayesian network [5] (Pearl, 1988). The General workflow of BOA is shown in Figure 2.16.

An extension to BOA called hierarchical BOA (hBOA) has also been proposed by Pelikan & Goldberg (2000). The idea is to improve the efficiency of BOA by using a Bayesian network with a local structure (Chickering *et al.*, 1997) to model the distribution and a restricted tournament replacement strategy based on work of Harik (1994) to form the new population. hBOA has been reported to solve two class of complex optimisation problem known as Ising spin glasses and MAXSAT (Pelikan & Goldberg, 2003). Some recent works on efficiency enhancements of hBOA can be found in (Pelikan *et al.*, 2005a,b).

BOA, EBNA, LFDA

1. Generate a population P of M solutions
2. Select N promising solution from P, where $N <= M$
3. Estimate a Bayesian network from selected solutions
4. Sample Bayesian network to generate M new individual and replace P
5. Go to step 2 until termination criteria are meet

Figure 2.16: The pseudo-code of BOA, EBNA, LFDA

Estimation of Bayesian Network Algorithm (EBNA)

Estimation of Bayesian Network Algorithm (EBNA) was proposed by (Etxeberria & Larrañaga, 1999; Larrañaga *et al.*, 2000) and also uses Bayesian networks (2.13) as its

[5] more detail description of Bayesian networks is in chapter 3

model of probability distribution. The workflow of EBNA is similar to that of BOA (Figure 2.16). Depending on the metric used to measure the quality of a Bayesian network, three different variants of EBNA have been proposed. They are:

1. EBNA$_{PC}$ using the PC algorithm of Spirtes *et al.* (1991)
2. EBNA$_{BIC}$ using Bayesian Information Criterion (BIC) metric (Schwarz, 1978)
3. EBNA$_{K2+pen}$ using $K2$ algorithm with a penalising factor (Cooper & Herskovits, 1992)

EBNA has been applied for a range of different optimisation problems, such as graph matching (Bengoetxea *et al.*, 2000, 2001b,a), partial abductive inference in Bayesian networks (de Campos *et al.*, 2001), feature subset selection (Inza *et al.*, 2000, 2001b,a), job scheduling problem (Lozano *et al.*, 2001b), rule induction task (Sierra *et al.*, 2001), travelling salesman problem (Robles *et al.*, 2001), partitional clustering (Roure *et al.*, 2001), Knapsack problems (Sagarna & Larrañaga, 2001), and software testing (Sagarna & Lozano, 2005, 2006). Also some parallel approach to EBNA has been proposed in (Lozano *et al.*, 2001a; Mendiburu *et al.*, 2005).

Learning Factorised Distribution Algorithm (LFDA)

Learning Factorised Distribution Algorithm (LFDA) has been proposed as an extension to the FDA (Mühlenbein & Mahnig, 1999b). Unlike FDA, LFDA does not require linkage in advance. Rather, in each iteration, it computes a Bayesian network and samples it to generate new solutions. By such, workflow of LFDA is very similar to that of BOA and EBNA algorithms (Figure 2.16).

Markov Network Factorised Distribution Algorithm (MN-FDA) and Markov Network Estimation of Distribution Algorithm (MN-EDA)

Markov Network Factorised Distribution Algorithm (MN-FDA) and Markov Network Estimation of Distribution Algorithm (MN-EDA) has been recently proposed by Santana (2003a, 2005). They use *Markov network* (Pearl, 1988; Li, 1995) as the model of distribution for $p(x)$. MN-FDA uses a technique called *junction graph approach*, whereas MN-EDA uses a technique called *Kikuchi approximation* to estimate a Markov network. Detail description of Markov network and both of this approach is described in chapter 3. Please see Santana (2003a), Santana (2005), and Santana *et al.* (2005) for more on workflow and experimental results of these algorithms.

2.3 Summary

Evolution in GAs can be seen as combination of two processes: Selection and Variation. Selective pressure favours the evolution of high-quality solutions. Variation helps to explore the search space and exploit those regions containing better solutions. An important factor in the success of this process is the linkage between variables which tells how variables in the solution interact to have a positive effect in the fitness function. Variation that is incompatible with linkage may not effectively optimise the fitness function. The need to discover linkage has lead to the development of the probabilistic approach to variation, where the linkage is used to estimate the distribution of the solutions and sampled to generate the child population. Algorithms using this approach to variation are known as EDAs.

The aim of this chapter was to give a general introduction to EDAs. We described GAs as the parent algorithms to EDAs. Then we described the motivation for probabilistic approach to variation followed by the workflow of EDA. We also presented a literature survey on discrete EDAs where we categorised them according to the types of linkage they use, briefly described their model of distribution and their workflow, and described some

related works on their applications.

Being the main source of variation, the success and failure of an EDA mainly depends on how well it detects the linkage to model the probability distribution and samples it to generate the child population. Therefore, one of the key areas of interest in EDA research is the development of effective techniques for estimation and sampling of the distribution. An increasing ammount of research work is being carried out in this area. The presented thesis introduces one such approach based on probabilistic graphical models and can be seen as a step forward towards this area of research.

Chapter 3

Probabilistic Graphical Models and EDAs

The success or failure of an EDA depends on how well it estimates and samples the joint probability distribution (jpd), $p(x)$, of the solutions in the population. In the previous chapter we have shown how $p(x)$ can be estimated by factorising it in terms of the marginal (or conditional) probability of sets of interacting variables in the solutions. A factorised jpd can be efficiently sampled. All of the EDAs we have reviewed in the previous chapter factorise $p(x)$ in terms of marginal/conditional probabilities [1].

One of the most effective ways to represent any factorised distribution is as a **Probabilistic Graphical Model (PGM)** (Pearl, 1988; Whittaker, 1990; Lauritzen, 1996; Jordan *et al.*, 1999; Jordan, 2004). All three models of distribution described in the previous chapter can be represented as PGM. In fact many EDAs use the concept of PGM to estimate and sample the probability distribution. Furthermore, our approach to the estimation of distribution and sampling is also motivated by the use of a class of PGM. Therefore, before going further to describe our approach, it is essential to understand the PGM in the context of EDA.

[1] except MN-EDA, which will be discussed later in this chapter

In this chapter, we describe PGM. The objective here is twofold:

1. To introduce and review the PGM in context of EDAs
2. To form the basis for the remaining part of the thesis by introducing the concepts and terminology associated with the Markov Random Field approach to estimation of distribution proposed in this thesis.

We start by introducing the basic components of PGM. We then categorise PGM into two groups according to the structure they use: 1) Directed and 2) Undirected. We then describe each of these categories in detail. As the work presented in this thesis uses an undirected PGM approach to estimation and sampling, more focus will be given to describe them and their use in EDAs.

3.1 Probabilistic Graphical Models

Probabilistic Graphical Models (PGM) can be seen as a merger of two disciplines, probability theory and graph theory (Jordan, 1998). They consist of two components: *structure* and *parameters*.

The structure of a PGM is the interaction between random variables depicted in the form of a graph. In context of EDAs the structure of a PGM is the linkage between variables in the solution. Each node of the graph represents a random variable. The presence of an edge between two nodes represents the existence of an interaction between them. Similarly, the absence of an edge between two nodes represent the absence of interaction between them.

The parameters of a PGM are a collection of *potential functions* associated with a node (or set of nodes) in the structure of the model (Jordan, 1998; Smyth, 1998). The potential function represents the strength of the interaction between variables. In general EDAs, the parameters of the PGM are usually a collection of marginal (or conditional) probabilities.

According to the type of structure used, PGMs can be categorised into two groups.

1. Directed models (Bayesian networks)
2. Undirected models (Markov Random Fields/Markov networks)

Traditionally, undirected models have been extensively used in the physics and vision communities and directed models have been widely used in the AI and Bayesian statistics communities (Smyth, 1998; Murphy, 2002). In the EDA literature, Bayesian networks have been frequently applied and are established as a useful approach for modelling the distribution. However, works published in recent years (Shakya *et al.*, 2004b, 2005a,b,c; Santana, 2003a, 2005) have used Markov Random Field approaches to probabilistic modelling in EDAs.

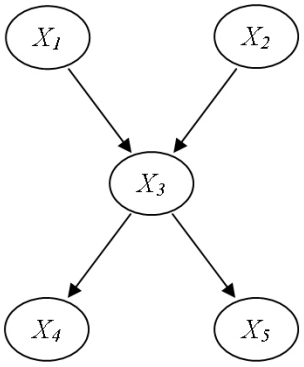
3.2 Bayesian networks

A Bayesian network can be regarded as a pair (B, Θ), where B is the structure of the model and the Θ is a set of parameters of the model. The structure B is a *Directed Acyclic Graph (DAG)* [2], where each node corresponds to a variable in the modelled data set and each edge corresponds to a conditional dependency. A set of nodes Π_i is said to be the parent of X_i if there are edges from each variable in Π_i pointing to X_i.

The parameter $\Theta = \{p(x_1|\Pi_1), p(x_2|\Pi_2), ..., p(x_n|\Pi_n)\}$ of the model is the set of conditional probabilities, where each $p(x_i|\Pi_i)$ is the set of probabilities associated with a variable $X_i = x_i$ given the different configuration of it's parent variables Π_i.

Figure 3.1 shows the structure and the parameters of a Bayesian network, where each variable X_i is binary. i.e. $x_i \in \{0, 1\}$. By the chain rule of probability (2.5), the jpd of

[2] A DAG is a graph where each edge joining two nodes is a *directed edge*, and also there is *no cycle* in the graph i.e. it is not possible to start from a node and travelling towards the correct direction return back to the starting node



a. Structure

$p(x_1 = 0)$, $p(x_1 = 1)$

$p(x_2 = 0)$, $p(x_2 = 1)$

$p(x_3 = 0 \mid x_1 = 0, x_2 = 0)$, $p(x_3 = 0 \mid x_1 = 1, x_2 = 0)$,
$p(x_3 = 0 \mid x_1 = 0, x_2 = 1)$, $p(x_3 = 0 \mid x_1 = 1, x_2 = 1)$,
$p(x_3 = 1 \mid x_1 = 0, x_2 = 0)$, $p(x_3 = 1 \mid x_1 = 1, x_2 = 0)$,
$p(x_3 = 1 \mid x_1 = 0, x_2 = 1)$, $p(x_3 = 1 \mid x_1 = 1, x_2 = 1)$

$P(x_4 = 0 \mid x_3 = 0)$, $p(x_4 = 0 \mid x_3 = 1)$,
$P(x_4 = 1 \mid x_3 = 0)$, $p(x_4 = 1 \mid x_3 = 1)$

$p(x_5 = 0 \mid x_3 = 0)$, $p(x_5 = 0 \mid x_3 = 1)$,
$p(x_5 = 1 \mid x_3 = 0)$, $p(x_5 = 1 \mid x_3 = 1)$

b. Parameter

Figure 3.1: A Bayesian network on 5 binary random variables

network presented in Figure 3.1 is

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4) \quad (3.1)$$

However, the conditional independence relationships encoded in the Bayesian network says that a variable X_i is independent of all other variables given its parents Π_i. In Figure 3.1, this follows X_5 is independent of X_1, X_2 and X_4 given X_3. Therefore, for this network, we can say that $p(x_5|x_1, x_2, x_3, x_4) = p(x_5|x_3)$. Similarly we can say $p(x_4|x_1, x_2, x_3) = p(x_4|x_3)$ and $p(x_2|x_1) = p(x_2)$. This gives the compact factorisation for the joint probability distribution shown in (3.1) and can now be written as:

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)p(x_5|x_3) \quad (3.2)$$

In general, given a set of variables $X = \{X_1, X_2, .., X_n\}$ a joint probability distribution $p(X = x)$ (or simply $p(x)$) for any Bayesian network is

$$p(x) = \prod_{i=1}^{n} p(x_i|\Pi_i) \quad (3.3)$$

In recent years, Bayesian networks have been the subject of growing interest in the Artificial Intelligence (AI) community. There is now a substantial literature in this field, with

major publications including (Pearl, 1988; Jensen, 1996, 2001; Lauritzen & Spiegelhalter, 1988; Lauritzen, 1996). As we can see from the survey in previous chapter, Bayesian networks have been widely used to model the distribution in EDAs. In fact, apart from EcGA, FDA, MN-EDA and MN-FDA, the model of distribution used by rest of the reviewed EDAs can be represented as a Bayesian network.

3.2.1 Learning the Structure of a Bayesian network

Bayesian networks provide a compact representation for the interaction between variables in a solution. This provides a tool to factorise a jpd. Nevertheless, the problem of determining the structure of the network remains. It has been shown that this problem in itself is NP hard (Chickering *et al.*, 1994). However, researchers have come up with different techniques to address this problem. Following the approach taken by Larrañaga & Lozano (2002) we categorise them into two groups.

1. Detecting conditional independencies
2. Score+Search methods

3.2.1.1 Detecting conditional independencies

This technique starts with a complete undirected graph, i.e. the graph where each node is connected to every other node. Then, by removing the edges corresponding to the mutually independent nodes, simplifies the graph. These mutually independent nodes are identified by applying a statistical independence test [3]. Once a simplified undirected graph has been found, some *adjacency rule* described in Spirtes *et al.* (1993); Larrañaga & Lozano (2002) is used to add the direction for the edges. EBNA$_{pc}$ (Etxeberria & Larrañaga, 1999) and BMDA (Pelikan & Mühlenbein, 1999) [4] are based on detecting conditional independencies to find the structure for Bayesian networks.

[3] In section 3.3.1.2 one such test is described
[4] described earlier in chapter 2

3.2.1.2 Score+Search methods

Score+Search techniques are composed of two elements: a *Scoring metric* and a *Search procedure.*

Scoring metrics

Given a network structure, a scoring metric measure the quality of that structure. In GA terms, it can be seen as the fitness function of the structure. As mentioned in (Pelikan, 2002), two approaches to scoring has been used in EDAs: *Bayesian Metrics* and *Minimum Description Length Metrics.*

Bayesian metrics (Heckerman *et al.*, 1994; Cooper & Herskovits, 1992) use Bayes rule to measure the quality of a network structure by computing a *marginal likelihood*, $p(B|D)$, of the structure B with respect to the given data set, D. The higher the marginal likelihood of the structure is, the better the structure represents the dependences in the data set. In the context of EDAs, the dataset D is the set of selected solutions from the population P. In general, computing the marginal likelihood of a structure requires the computation of a *normalisation constant* involving all possible network structures and their parameters. However, some general assumptions about the data are usually made in order to efficiently compute the marginal likelihood (Cooper & Herskovits, 1992; Heckerman *et al.*, 1994; Larrañaga & Lozano, 2002).

EBNA_{K2+pen} and BOA (with BD metric) use Bayesian metrics to compute Bayesian networks. Bayesian metrics tend to be more sensitive to the noise in data set D (Pelikan, 2002). This often results in a complicated network structure with unnecessary dependencies. To avoid this situation, researchers often restrict the number of dependencies allowed to a node to a fixed value.

Minimum Description Length (MDL) Metrics (Rissanen, 1978) are based on the idea that good models are those that minimise the amount of space required to represent the model and at the same time maximise the number of regularities in the data encoded by

the model. MDL tries to give a higher score to networks that balance these two criteria. In contrast to Bayesian metrics, MDL metrics tend to be less sensitive to the information contained by data, often resulting in network model that does not capture all the necessary interactions (Pelikan, 2002). To overcome this situation, MDL requires large number of samples.

The three EDAs described in previous chapter: EcGA (Harik, 1999), BOA (with BIC) (Pelikan, 2002) and EBNA$_{BIC}$ (Larrañaga & Lozano, 2002) uses MDL metrics to construct a Bayesian network.

Search procedures

Once a scoring metric to measure the goodness of the structure is chosen, a search through the space of possible models should be performed in order to find a good model. This task in itself is NP hard (Chickering *et al.*, 1994). Different search heuristics can be used for this purpose. In EDAs, researchers tend to use greedy heuristics, mainly due to their simplicity and the acceptable quality of results produced by them. Here, we describe a greedy search method known as Algorithm B (Buntine, 1991). It starts with an initial structure which is often a structure with no edges. Then at each step adds an edge to the structure that gives maximum improvement to its quality. This continues until no further improvement can be made. It is important to ensure that the added edge do not discard the DAG principle, i.e. do not create cycles, in the structure.

3.2.2 Parameter Learning and Sampling a Bayesian network

In the context of EDAs, parameter learning for a Bayesian network is a fairly simple task in comparison to structure learning. This is because the value of each variable in the selected set of solutions is known. In other words, the dataset D is complete. Parameters, Θ, are calculated by counting the frequency of variable instances from the D according to the structure of the network. An example is shown in (2.7).

The sampling process is again a trivial task. This is mainly due to the fact that the

constructed network structure is a DAG and therefore naturally gives the order in which to sample the variables. The idea is to sample a variable X_i in the sequence such that all the parent variable of X_i, Π_i, are generated prior to sampling X_i. In literature this method for sampling a Bayesian network is also known as *Probabilistic Logic Sampling* (PLS) (Henrion, 1988).

3.3 Markov Random Fields

Markov Random Fields (MRFs) are an emerging approach to estimating the distribution in EDAs (Shakya *et al.*, 2004b, 2005b,a,c; Santana, 2003a, 2005). In comparison to Bayesian networks, they have not been as thoroughly studied in the EDA literature. This thesis proposes a novel technique based on MRFs to estimate the distribution in EDAs. Therefore, in this section we describe MRFs and also explain their relevance to EDAs.

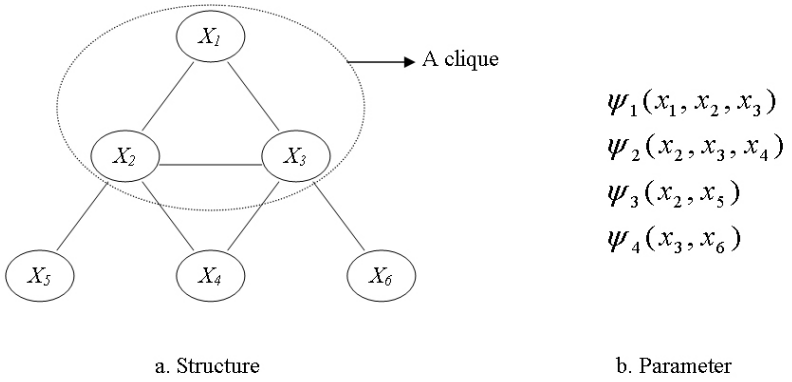


Figure 3.2: A Markov Random Field on 6 random variables

A Markov Random Field is a pair (G, Ψ), where G is the structure and the Ψ is the parameter set of the network. G is an undirected graph where each node corresponds to a random variable in the modelled data set and each edge corresponds to conditional dependencies between variables. However, unlike Bayesian networks, the edges in Markov Random Fields are undirected. Here, the relationship between two nodes should be seen as a *neighbourhood relationship*, rather than a parenthood relationship. We use $N =$

$\{N_1, N_2, ..., N_n\}$ to define a *neighbourhood system* on G, where each N_i is the set of nodes neighbouring to a node X_i [5].

A MRF is characterised by its *local Markov property* known as *Markovianity* (Besag, 1974; Li, 1995) which states that a node X_i can be completely defined by knowing only its neighbouring nodes N_i. In terms of probability, this can be written as $p(x_i|x - \{x_i\}) = p(x_i|N_i)$. N_i is sometimes referred to as *Markov Blanket* for X_i (Murphy, 2002).

A MRF can be also characterised by its *global Markov property*, which is defined in terms of the structure of a graphical model as follows: two (sets of) nodes, A and B, are conditionally independent given a third set, C, if all paths between the nodes in A and B are separated by a node in C (Murphy, 2002). In Figure 3.2, node X_1 is conditionally independent of X_5, X_4 and X_6 given X_2 and X_3. Global Markov property helps to formulate the joint probability distribution, $p(x)$, for an MRF.

Let us first define some of the related terms.

Definition 3.3.1 (Clique) *Given an undirected graph G, a clique is a fully connected subset of the nodes.*

Definition 3.3.2 (Sub Clique) *Given an undirected graph G, a sub clique of a clique is a fully connected subset of nodes within that clique.*

Definition 3.3.3 (Maximal Clique) *A clique is called maximal, if it is not a sub clique of any other clique.*

Definition 3.3.4 (Singleton Clique) *A clique is called singleton if it consist of a single node from G.*

In order to formulate $p(x)$ for a MRF, we first need to define its parameters in terms of cliques in G.

[5] In literatures, MRF is also defined in terms of sites and neighbourhood system (Besag, 1974; Li, 1995), where a site corresponds to a node X_i

The parameters of a MRF, $\Psi = \{\psi_1(c_1), \psi_2(c_2), ..., \psi_m(c_m)\}$, is a set of positive *potential functions* defined on the set of cliques $C = \{C_1, C_2, ..., C_m\}$ of structure G. In general, the set C can consist of all possible cliques in G, i.e. all maximal cliques, their sub cliques including singleton cliques. However, it is always possible to consider only the maximal cliques in C and specify the parameters Ψ. We use c_i to denote the set of values taken by the set of variables in the clique C_i. $\psi_i(c_i)$ is a potential function defined over clique C_i reflecting the neighbourhood relationship between the nodes in C_i. They are also known as *clique potential functions.*

For example, the graph shown in Figure 3.2 has four maximal cliques

$$C_1 = \{X_1, X_2, X_3\},\ C_2 = \{X_2, X_3, X_4\},\ C_3 = \{X_2, X_5\},\ C_4 = \{X_3, X_6\}$$

The jpd $p(x)$ for the structure shown in Figure 3.2 can be factorised in terms of the clique potential functions as

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{1}{Z}\psi_1(x_1, x_2, x_3)\psi_2(x_2, x_3, x_4)\psi_3(x_2, x_5)\psi_4(x_3, x_6) \tag{3.4}$$

Where, Z is a *normalising constant.* In general, given a set of variables $X = \{X_1, X_2, .., X_n\}$ a joint probability distribution $p(X = x)$ or simply $p(x)$ for any MRF is

$$p(x) = \frac{1}{Z}\prod_{i=1}^{m}\psi_i(c_i) \tag{3.5}$$

Where, m is the number of cliques in the structure G. $Z = \sum_{x\in\Omega}\prod_{i=1}^{m}\psi_i(c_i)$ is the normalising constant known as the *partition function* which ensures that $\sum_{x\in\Omega} p(x) = 1$. Here, Ω is the set of all possible solutions.

3.3.1 Learning the Structure of a Markov Random Field

As we shall show in the subsequent sections of this chapter, learning the structure of a MRF in EDAs is a comparatively simpler task than the parameter learning and sampling.

There are several methods that can be applied for learning structure of a MRF. We review two of them due to their use in EDAs.

1. Statistical independence test (Santana, 2005)
2. Linkage detection algorithm (Heckendorn & Wright, 2004)

3.3.1.1 Statistical independence test

The general idea behind this method is as follows: it starts with a complete undirected graph, then by removing the edges corresponding to the mutually independent nodes, simplifies the graph. These mutually independent nodes are identified by applying an independence test. Different statistical test can be applied for this purpose. Here we restrict them to the *chi-square test* (Marascuilo & McSweeney, 1977).

Given a dataset D, chi-square statistic test states that, if the bivariate distribution of a pair, $X_i = x_i$ and $X_j = x_j$ in D, is equal to the product of their univariate distribution then they are statistically independent, i.e. in terms of probability, if $p(x_i, x_j) - p(x_i)p(x_j) = 0$, variables are fully independent for that particular configuration (however, in practice, a threshold value instead of 0 is used, so that partial dependency can also be considered). To conduct the independency test between X_i, X_j in general, we need to extend this for all of their configurations. This is shown below

$$X^2_{i,j} = \sum_{x_i,x_j} \frac{(p(x_i, x_j) - p(x_i)p(x_j))^2}{p(x_i)p(x_j)} \tag{3.6}$$

For binary case, variables X_i and X_j are said to be 95% independent if the threshold of $X^2_{i,j} < 3.84$ is true.

In general, chi-square statistic compares the *observed* against *expected* values and can be written as:

$$X^2 = \sum \frac{(observed - expected)^2}{expected} \tag{3.7}$$

Some use of chi-square to find both undirected and directed network structure can be

found in Santana (2003b, 2005) and Pelikan & Mühlenbein (1999) respectively.

The chi-square statistic can be further extended to find the interaction between two variables given a third variable. We call it a conditional independency test of order 1. This can be written as follows:

$$X^2_{i,j|k} = \sum_{x_i,x_j,x_k} \frac{(p(x_i,x_j|x_k) - p(x_i|x_k)p(x_j|x_k))^2}{p(x_i|x_k)p(x_j|x_k)} \tag{3.8}$$

Now let us define the steps of the algorithm to find the structure using the chi-square statistic:

1. Start with a complete undirected graph.

2. For each pair of variables X_i and X_j, perform the chi square test with some threshold. If the result is less than the threshold, the variables are independent to each other and the edge between them is removed.

3. For the resulting network, perform conditional independence test of order 1 and if independency is found, remove the edge. This step may further simplify the structure.

   Theoretically, this process can be extended to check higher order conditional independency of order up to $n-1$. Then the resulting structure will be the exact dependency structure for the given data set D. However doing this is computationally expensive. Therefore, usually in real life applications, 1^{st} order dependency check is taken as sufficient criteria to get a useful model.

3.3.1.2 Linkage Detection Algorithm

A recently proposed EDA called Linkage Detection Factorization Algorithm (LDFA) proposed by Wright & Pulavarty (2005) uses Linkage detection algorithm of Heckendorn & Wright (2004) to find the interaction between variables. Linkage detection algorithm requires binary representation of solution x and also requires the maximum degree of in-

teraction between bits, k, to be specified in advance. To test the interaction between two bits, this algorithm records the change in fitness while flipping each bit individually and also while flipping them together. If the sum of change in fitness for flipping each bit individually is different than the change in fitness of flipping both bits together, then there is an interaction between them. This can be easily extended to identify higher order interaction between variables. The complexity of linkage detection algorithm grows exponentially to the order of interaction taken into account (Heckendorn & Wright, 2004; Wright & Pulavarty, 2005).

3.3.2 Parameter Learning and Sampling a Markov Random Field

Once the structure has been found, parameter learning and sampling can be done. As stated earlier, in MRF, this task is comparatively more complex than in Bayesian networks. In general, the parameter learning in MRF involves the calculation of the partition function, Z, that is exponential in the number of nodes and therefore is not feasible to calculate in general conditions. Sampling the network is also a difficult task, mainly because of the undirected structure of the network. It does not provide the ordering in which to learn or sample. There exist several techniques to overcome the problem of learning and sampling the MRFs. We review three of them as they have been exploited in EDAs.

1. Junction tree approach
2. Junction graph approach
3. Kikuchi approximation approach

Before describing these approaches, we need to give definitions to some of the terms used here after. Figure 3.3 shows the structure of a MRF on 10 random variables, where a clique and a maximal clique are also shown.

Definition 3.3.5 (Maximum Clique) *A clique is called maximum, if it has the highest*

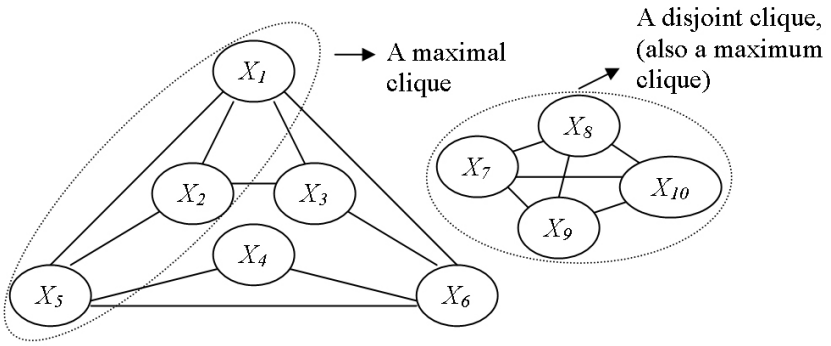


Figure 3.3: An undirected graph on 10 random variables, showing a clique, a maximal clique and a maximum clique

number of edges among all cliques in G. There may be more than one maximum clique in a graph.

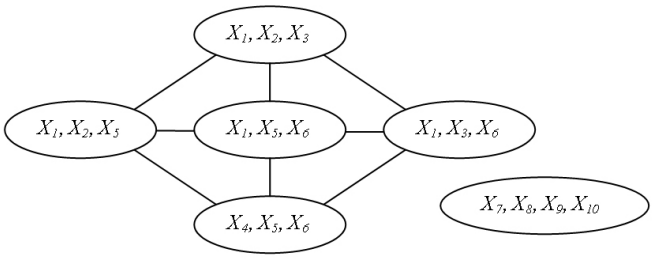


Figure 3.4: A junction graph for the undirected graph shown in Figure 3.3

Definition 3.3.6 (Junction Graph) *A junction graph of the undirected graph G is a graph, where each node corresponds to a maximal clique of G. There is an edge between two nodes if their corresponding cliques overlap.*

Definition 3.3.7 (Chordal Graphs) *An undirected graph G is said to be chordal, if all the cycles made of more than three edges in G has a chord. Chordal graph is also known as triangulated graph.*

Definition 3.3.8 (Junction Tree) *In a chordal graph, if there is a common node X_i in any two maximal cliques C_j and C_k, and it is possible to construct a tree connecting all*

the maximal cliques where, C_j and C_k are either neighbouring nodes or all the nodes in the path joining C_j and C_k contains X_i, then such tree is called a junction tree.

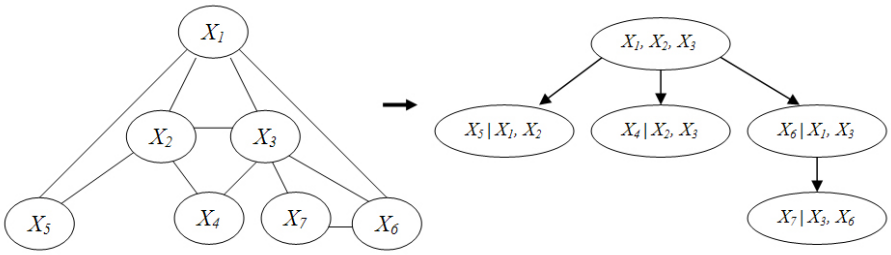


Figure 3.5: An undirected chordal graph and a junction tree associated with it

Definition 3.3.9 (Running Intersection Property) *A junction tree follow the running intersection property, which states that, if there is a common variable X_i in any two nodes C_j and C_k, then they are either a neighbouring nodes in the tree, or all the nodes in the path connecting them also contains X_i.*

3.3.2.1 Junction tree approach

The junction tree approach, (also known as a *valid factorisation* approach) (Lauritzen & Spiegelhalter, 1988) can be applied for the case where the discovered structure of the MRF can be correctly represented as a junction tree. If the structure of the network is a junction tree then *the running intersection property* becomes true and the jpd shown in (3.5) can be written in terms of marginal/conditional probabilities as follows:

$$p(x) = \prod_{i=1}^{m} p(c_i) \tag{3.9}$$

where,

$$p(c_i) = p(r_i|s_i)$$

Here, a clique $C_i = c_i$ is devided into two sets $R_i = r_i$ and $S_i = s_i$. R_i is known as the *residual* and S_i is known as the *separator* of the clique C_i (Mühlenbein *et al.*, 1999). For example, for the junction tree shown in Figure 3.5, the jpd in terms of residual and

separator can be written as

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = p(x_1, x_2, x_3)p(x_4|x_2, x_3)p(x_5|x_1, x_2)p(x_6|x_1, x_3)p(x_7|x_3, x_6) \tag{3.10}$$

The parameter estimation and sampling of a junction tree is a process similar to the parameter estimation and sampling of a Bayesian network [6]. The main idea being estimate (or sample) S_i before estimating R_i.

For cases, where, the network structure cannot be represented as a junction tree, the method known as *triangulation* should be performed in order to first triangulate the network structure. There are two ways to triangulate a graph.

1) **By adding some edges**: The advantage of this approach is that the structure considers more dependencies which may result in better approximation of distribution. However, the disadvantage is that the graph may become dense and the size of maximum clique may increase resulting in high complexity to compute the marginal/conditional probabilities and therefore increasing the computational complexity.

2) **By removing some edges**: The advantage of this approach is that the graph gets less dense increasing the efficiency of estimating marginal/conditional probability. However, it may remove important dependency and therefore can result in less accurate approximation of distribution.

In the context of EDAs, the junction tree approach has been first used in FDA (Mühlenbein *et al.*, 1999; Mühlenbein & Mahnig, 1999a,b). A recent EDA called LDFA proposed by Wright & Pulavarty (2005) also uses junction tree approach.

Note that, by applying *moralisation*: a procedure to transform the directed graph of a Bayesian networks to an undirected graph, the junction tree approach has also been used to learn the parameters for Bayesian networks. For more information on this topic, interested readers are recommended to see (Jensen & Jensen, 1994; Jensen, 1996; Jordan

[6] In fact, any MRFs that can be factorised as a junction tree can also be represented as a Bayesian network

et al., 1999).

3.3.2.2 Junction graph approach

The junction graph approach, (also known as *invalid factorisation approach* (Santana, 2003a)), extends the junction tree approach and can be applied even when the structure of a MRF cannot be correctly represented as a junction tree. The MN-FDA proposed by Santana (2003a,b), uses such approach to estimate and sample the distribution.

Definition 3.3.10 (Ordered Junction Graph) *An ordered junction graph is a junction graph that satisfies following conditions:*

- *It has an associated ordering of the nodes.*
- *It has a distinguished node called the root.*
- *A node belongs to the graph if at least one of the variables in the clique is contained in the previous nodes in the ordering.*

The factorisation of jpd in this case is based on *ordered junction graph*, that, not being a junction tree, does not guarantee the accuracy of the distribution encoded in the structure. General steps for parameter learning and sampling using a junction graph approach are presented below. It is assumed that the structure learning is done using a statistical independence test with chi-square statistic.

1. **Refine the structure of the network**: If the found network structure is dense, i.e., if there is a large number of edges to a single node, the structure should be *refined* by reducing the number of edges to a node. One of the ways to do this is by allowing to a node only r number of edges having best chi-square values. However, there is a trade off between accuracy verses simplicity. i.e., the edges needed to accurately model the interaction may be greater than r. Therefore, determining accurate r may be problematic.

2. **Find all maximal cliques in the graph and calculate their weight**: The weight of a clique is the product of the chi-square value of all edges in that clique. There are different algorithms that can be used to find the maximal cliques. In Santana (2003b) *Born and Kerbosch algorithm* (Born & Kerbosch, 1973) has been used.

3. **Construct an ordered junction graph**: It can be done by first ordering the maximal cliques by their weights, then choosing a clique with maximal weight as the root. The ordered junction graph is then populated by adding one clique at a time to the graph. The clique to be added should be chosen such that among remaining of the cliques, the added clique has the maximum number of overlapping variables to one of the already added base nodes of the graph.

4. **Construct a junction tree from the ordered junction graph**: This is done by removing any cycles contained by the ordered junction graph. Therefore constructed junction tree may not cover all the dependency covered by the ordered junction graph (Santana, 2003b).

5. **Calculate probabilities and sample**: Using (3.9), calculate the marginal/conditional probabilities as factorised by the junction tree and sample them to generate child population.

3.3.2.3 Kikuchi approximation approach

The Kikuchi approximation approach (Kikuchi, 1951) (also known as *messy invalid factorisation*) further extends the representation capability of the factorisation based on junction graph. It has been used in Markov Network EDA (MN-EDA) proposed by (Santana, 2003b, 2005).

The Kikuchi approximation is a generalisation of well-known *Bethe approximation* in statistical physics (Bethe, 1935; Yedidia *et al.*, 2001, 2002, 2005). Kikuchi approximation approach is based on the idea of finding an appropriate set of marginals from which to obtain an approximation of the distribution. Here, a structure G is decomposed into a

sets of regions $\Re = \{\Re_0, \Re_1, ..., \Re_s\}$ where each $\Re_i \in \Re$ is a set of cliques. A set $\Re_i$ contains the cliques of same order, and for each i the order is different. Each region (clique) $R \in \Re_i$ has a *counting number* c_R. Given the set of region $\Re_i$, the counting number c_R of each region $R \in \Re_i$ can be calculated as follows:

$$c_R = 1 - \sum_P c_P \vdots R \subset P \tag{3.11}$$

Where, c_P is the counting number of any region P that contains region R.

A regional decomposition of a graph G is said to be *valid* if the counting numbers for all the regions consisting of a variable X_i (including singletion region consisting of only X_i) sums up to 1 i.e:

$$\sum_{\substack{R \in \Re \\ X_i \subseteq R}} c_R = 1 \tag{3.12}$$

Here, each $R \in \Re$ is a region containing the variable X_i.

The approximation of distribution done from the factorisation of structure according to valid region based decomposition is said to be acceptable. The key idea in this approach is to find a valid regional decomposition of the structure.

To clarify the concept of regions and counting numbers, let us give an example: The factorisation of the jpd for the junction tree of the graph in Figure 3.5 can be written as:

$$\begin{aligned} p(x) &= p(x_1, x_2, x_3)p(x_4|x_2, x_3)p(x_5|x_1, x_2)p(x_6|x_1, x_3)p(x_7|x_3, x_6) \\ &= p(x_1, x_2, x_3)\frac{p(x_4, x_2, x_3)p(x_5, x_1, x_2)p(x_6, x_1, x_3)p(x_7, x_3, x_6)}{p(x_2, x_3)p(x_1, x_2)p(x_1, x_3)p(x_3, x_6)} \\ &= p(x_1, x_2, x_3)p(x_4, x_2, x_3)p(x_5, x_1, x_2)p(x_6, x_1, x_3) \\ &\quad p(x_7, x_3, x_6)p(x_2, x_3)^{-1}p(x_1, x_2)^{-1}p(x_1, x_3)^{-1}p(x_3, x_6)^{-1} \end{aligned} \tag{3.13}$$

This structure, being a junction tree, can be validly decomposed into set of two regions, $\Re_0$ containing all the maximal cliques and $\Re_1$ containing all cliques of order 2 involved in

the factorisation.

$$\begin{aligned}\Re_0 &= \{x_1, x_2, x_3\}\{x_4, x_2, x_3\}\{x_5, x_1, x_2\}\{x_6, x_1, x_3\}\{x_7, x_3, x_6\} \\ \Re_1 &= \{x_2, x_3\}\{x_1, x_2\}\{x_1, x_3\}\{x_3, x_6\}\end{aligned}$$

Applying (3.11) to our chosen regional decomposition, we get $c_R = 1 \ \forall \ R \in \Re_0$. This is because P for a maximal clique $R \in \Re_0$ does not exist. In other words, the counting number for all the maximal clique (all region in top level) is 1. Similarly, from (3.11), we get $c_R = -1 \ \forall \ R \in \Re_1$. For example: counting number for region $\{x_2, x_3\}$, from (3.11) is

$$\begin{aligned}c_{\{x_2,x_3\}} &= 1 - (c_{\{x_1,x_2,x_3\}} + c_{\{x_4,x_2,x_3\}}) \\ &= 1 - (1 + 1) \\ &= -1\end{aligned}$$

and therefore, we can write following for our chosen decomposition.

$$\begin{aligned}c_R &= 1 \ \forall \ R \in \Re_0 \\ c_R &= -1 \ \forall \ R \in \Re_1\end{aligned}$$

The Kikuchi approximation, $k(x)$, of the jpd for a structure with valid decomposition is

$$k(x) = \prod_{R \in \Re} p(R)^{c_R} \tag{3.14}$$

And therefore, applying (3.14) to $\Re$, we get

$$\begin{aligned}p(\Re_0) &= p(x_1, x_2, x_3)p(x_4, x_2, x_3)p(x_5, x_1, x_2)p(x_6, x_1, x_3)p(x_7, x_3, x_6) \\ p(\Re_1) &= p(x_2, x_3)p(x_1, x_2)p(x_1, x_3)p(x_3, x_6) \\ k(x) &= p(\Re_0)p(\Re_1)^{-1} \\ &= p(x_1, x_2, x_3)p(x_4, x_2, x_3)p(x_5, x_1, x_2)p(x_6, x_1, x_3) \\ &\quad p(x_7, x_3, x_6)p(x_2, x_3)^{-1}p(x_1, x_2)^{-1}p(x_1, x_3)^{-1}p(x_3, x_6)^{-1} \\ &= p(x)\end{aligned}$$

From this example, two things should be noted.

1. A regional decomposition of a structure based on a junction tree is a valid decomposition.
2. The Kikuchi approximation of a structure that can be represented as a junction tree is equal to the factorisation of the jpd based on the junction tree

This shows the equivalence of the Kikuchi approximation approach and the junction tree approach.

However, the Kikuchi approximation is not limited to the junction tree structure and can be further extended to any junction graph. The main idea here is to find the regional decomposition of an structure such that it remains valid, i.e., satisfies (3.12). The two step procedure for the algorithm to find a valid regional decomposition for a structure is as follows:

1. Form the set $\Re_0$ by taking one region for each maximal clique in the structure.
2. Form the subsequent sets $\Re_i$ containing all the sub cliques from the region in $\Re_{i-1}$, where each sub clique will have the order decreased to one level than that of cliques in $\Re_{i-1}$ and will have the counting number $c_R \neq 0$. c_R for any region R can be calculated from (3.11)

For example: For the junction graph shown in the Figure 3.4, a valid regional decomposition using above procedure will be as follows:

$$\begin{aligned}
\Re_0 &= \{x_1, x_2, x_3\}\{x_1, x_2, x_5\}\{x_1, x_3, x_6\}\{x_4, x_5, x_6\}\{x_1, x_5, x_6\}\{x_7, x_8, x_9, x_{10}\} \\
\Re_1 &= \{x_1, x_2\}\{x_1, x_3\}\{x_1, x_5\}\{x_1, x_6\}\{x_5, x_6\} \\
\Re_2 &= \{x_1\}\{x_5\}\{x_6\}
\end{aligned}$$

Here $\Re_0$ contains all maximal cliques, $\Re_1$ contains all sub cliques in $\Re_0$ having $c_R \neq 0$ and

$\Re_2$ contains all sub cliques in $\Re_1$ having $c_R \neq 0$.

Note, that the maximal clique $\{x_7, x_8, x_9, x_{10}\}$ in $\Re_0$ does not need to be decomposed further as it is a fully connected disjoint clique and cannot be factorised further.

Applying (3.11) to the above decomposition we get:

$$c_R = 1 \; \forall \; R \in \Re_0$$
$$c_R = -1 \; \forall \; R \in \Re_1$$
$$c_R = 1 \; \forall \; R \in \Re_2$$

And therefore the Kikuchi approximation $k(x)$ applying (3.14) will be

$$\begin{aligned}
p(\Re_0) &= p(x_1, x_2, x_3)p(x_1, x_2, x_5)p(x_1, x_3, x_6)p(x_2, x_3, x_4)p(x_1, x_5, x_6)p(x_7, x_8, x_9, x_{10}) \\
p(\Re_1) &= p(x_1, x_2)p(x_1, x_3)p(x_1, x_5)p(x_1, x_6)p(x_5, x_6) \\
p(\Re_2) &= p(x_1)p(x_5)p(x_6) \\
k(x) &= \prod_{R \in \Re} p(R)^{c_R} = p(\Re_0)p(\Re_1)^{-1}p(\Re_2)
\end{aligned}$$

Once $k(x)$ is constructed, following sampling procedure is used by Santana (2005) to sample a new solution.

1) Create a solution $x = \{x_1, x_2, ..., x_n\}$ at random.

2) Then for r iterations, randomly choose a position $i \in n$, approximate the marginal probabilities $\widetilde{p}(x_i) = k(x_i)$ and sample to replace x_i in x.

We denote $k(x)$ having a variable $x_i = 1$ as $k(x|x_i = 1)$. Then, the marginal probabilities can be approximated as

$$\widetilde{p}(x_i = 1) = k(x_i = 1) = \frac{k(x|x_i = 1)}{k(x|x_i = 0) + k(x|x_i = 1)}$$

Similarly,

$$\widetilde{p}(x_i = 0) = k(x_i = 0) = \frac{k(x|x_i = 0)}{k(x|x_i = 0) + k(x|x_i = 1)}$$

3) Return the resulting x as a new solution.

The above sampling procedure can be seen as a variant of Gibbs sampler (Geman & Geman, 1987). Use of more sophisticated version of Gibbs sampler with a *cooling schedule* in EDA can be found in (Shakya *et al.*, 2005a,b). This will be discussed in more detail in Chapter 6 and Chapter 7.

3.4 Summary

PGM provides an effective and elegant tool to represent a factorisation for a distribution. Wide range of EDA uses the PGM approach to estimate and sample the distribution. In this chapter we have described PGM in context of EDAs. The objective was twofold: 1) to review the use of PGM in EDAs, and 2) to form the basis for the remaining part of the thesis by introducing the concepts associated with the undirected PGM.

We started by introducing the basic components of PGM. We then categorised PGM into two groups according to the structure they use: 1) Directed PGM (Bayesian networks) and 2) Undirected PGM (Markov Random Fields). We reviewed different structure learning and parameter learning techniques for both PGMs. We also reviewed different sampling techniques to sample from a PGM.

As the estimation of distribution technique proposed in this thesis is based on MRF, special focus was given to review three different techniques to learn and sample the MRFs. They were: junction tree approach, junction graph approach and Kikuchi approximation approach. In the remaining part of this thesis, we present our approach to estimation and sampling of distribution in EDAs based on MRF.

Chapter 4

Fitness modelling approach to estimating parameters in Markov Random Fields

As we have stated in earlier chapters, the variation process in EDA involves estimating the joint probability distribution (jpd), $p(x)$, from the population of solutions and sampling it to generate the child population. Many EDAs use Probabilistic Graphical Models (PGM) for this task, as they provide an elegant and effective way to represent a jpd, $p(x)$. Our approach to EDA is inspired by using undirected PGM (Markov Random Field (MRF)) to estimate and sample $p(x)$. The novelty of our approach, in comparison to other EDAs, is that we build a *model of fitness function* to approximate the parameters in MRF. Therefore, we call this approach the *fitness modelling approach* to estimating the MRF.

The aim of this chapter is to present the proposed fitness modelling approach. Note that, we do not focus on the structure learning part of the MRF [1], instead our focus will be on parameter learning and sampling.

The rest of the chapter is organised as follows. We start by presenting the concept of

[1] see chapter 3 for two different methods to learn the structure for the MRF

equivalence relationship between Markov Random Field and the Gibbs distribution. We then use this relationship to derive a model of the fitness function that relates the energy function of a Gibbs distribution with the fitness of a solution. We then present how to define energy for different structure of MRF. And finally, we present the way to learn the parameters of the MRF using derived model of fitness function.

4.1 Factorising MRF as a Gibbs distribution

Equation (3.5) in previous chapter defines jpd, $p(x)$, for any MRF as the product of positive potential functions $\psi_i(c_i)$, over the cliques C_i, in the structure of the model G. In this section, we describe its equivalence in terms of Gibbs distribution.

4.1.1 Gibbs distribution

A Gibbs distribution over a set of random variables X has the following form

$$p(x) = \frac{e^{-U(x)/T}}{Z} \tag{4.1}$$

where,

$$Z = \sum_{y \in \Omega} e^{-U(y)/T} \tag{4.2}$$

is a normalising constant, Ω is the set of all possible solutions, T is a parameter of the distribution known as the *temperature* and $U(x)$ (or more precisely $U(X = x)$) is known as the *energy* of the distribution. Given an undirected graph, G, on X, $U(x)$ is defined as a sum of potential functions over the cliques, C_i, in G.

$$U(x) = \sum_{i=1}^{m} u_i(c_i) \tag{4.3}$$

Therefore, (4.1) can also be written as

$$p(x) = \frac{e^{-\sum_{i=1}^{m} u_i(c_i)/T}}{Z} \tag{4.4}$$

Here, $u_i(c_i)$ (or more precisely $u_i(C_i = c_i)$) is a potential function defined over a clique C_i. Note that the relationship between $\psi_i(c_i)$ in (3.5) and $u_i(c_i)$ in (4.4) is defined as

$$\psi_i(c_i) = e^{-u_i(c_i)/T} \tag{4.5}$$

We use $C = \{C_1, C_2, .., C_m\}$ to denote the set of all considered clique in $U(x)$. As stated in previous chapter, the set C can consist of all possible cliques in G, i.e., all maximal cliques, their sub cliques including singleton cliques. However, it is always possible to consider only the maximal cliques in C and define the energy $U(x)$.

Temperature, T, has a very important role in Gibbs distribution. It controls the *sharpness* of the jpd. i.e. when the temperature is high, all configurations of X tends to be equally distributed. Conversely, near the zero temperature, the jpd concentrates around the *global energy minima*.

4.1.2 MRF-Gibbs equivalence

A jpd for any MRF on X obeys following three conditions:

1. $p(x) -> (0, 1)$, Probability of each x lies between 0 and 1
2. $p(x) > 0$, Positivity condition
3. $\sum_x p(x) = 1$ Sum over probability of all possible x is 1

In addition, an MRF also follows a fourth condition: its local Markov property [2], which states that, a variable X_i is conditionally independent of rest of the variables in X given its neighbouring variables N_i (Li, 1995; Besag, 1974). In terms of probability it can be written as

$$p(x_i|x - \{x_i\}) = p(x_i|N_i)$$

[2]Local Markov property has been also described in Chapter 3

We can also define a *Gibbs Random Field* over X, which is characterised by its global property: the Gibbs distribution.

Definition 4.1.1 (Gibbs Random field) *A set of random variables X with neighbourhood system N is said to be Gibbs Random Field (GRF), if and only if they obey a Gibbs distribution.*

The Hammersley-Clifford theorem (Hammersley & Clifford, 1971) then establishes the equivalence between the local Markov property of MRF and global Gibbs property of GRF.

Theorem 4.1.1 (The Hammersley-Clifford theorem) *Any set of random variables X with a neighbourhood system N is an MRF if and only if X is also a GRF on N*

Proof. can be found in (Besag, 1974). ∎

In another words, Hammersley-Clifford theorem states that a jpd for any MRF can be equivalently specified as a Gibbs distribution (4.1). The practical value of the theorem is that, the behaviour of a system using Gibbs distribution completely depends on the chosen form of potential functions, $u_i(c_i)$, and the temperature, T. These parameters can be varied in order to achieve desired system behaviour. We exploit this property of Gibbs distribution to estimate and sample the MRF in EDAs.

4.2 Using fitness to model the energy for the Gibbs distribution

This section is based on an early work of Brown *et al.* (2002) on use of MRF to model the GA fitness function, where they derive the equivalence relationship between energy of Gibbs distribution $U(x)$ and the fitness $f(x)$ of the solution.

Assuming that the probability of a solution is proportional to its fitness, the jpd, $p(x)$, can be modelled in terms of fitness as

$$p(x) = \frac{f(x)}{Z} \tag{4.6}$$

Where, $Z = \sum_{y \in \Omega} f(y)$ is the partition function and Ω is the set of all possible solutions.

For such $p(x)$, we have

1. $p(x) -> [0, 1]$, Probability of each x lies between 0 and 1
2. $p(x) > 0$, Positivity condition: assumes $f(x) > 0$. This can be maintained by mapping $f(x)$.
3. $\sum_{x \in \Omega} p(x) = 1$, Sum over probability of all solution is 1

Now, from (4.1) and (4.6), we can deduce following equivalence of jpd for MRF in terms of fitness function.

$$p(x) = \frac{e^{-U(x)/T}}{\sum_{y \in \Omega} e^{-}U(y)/T} \equiv \frac{f(x)}{\sum_{y \in \Omega} f(y)} \tag{4.7}$$

From which, in Brown *et al.* (2002), following relationship between fitness and the energy is deduced.

$$-ln(f(x)) = U(x) \tag{4.8}$$

For simplicity, here we assume T from (4.7) to be 1. (4.8) defines the equivalence shown in (4.7).

We refer to (4.8) as **M**RF **F**itness **M**odel **(MFM)**. From (4.3), MFM can also be written in terms of potential functions as:

$$-ln(f(x)) = \sum_{i=1}^{m} u_i(c_i) \tag{4.9}$$

Energy, $U(x)$, in MFM (4.8) gives the full specification of the jpd (4.1), so MFM can be regarded as a probabilistic model of the fitness function. Also notice that, minimising

$U(x)$ here is equivalent to maximising $f(x)$. In subsequent sections, we show how MFM is used to estimate the parameters for the MRF.

4.3 Defining energy in terms of potential functions

In general, the form of energy, $U(x)$ in MFM, will model the different order of interaction between variables in X. The form of energy, however, will depend on our chosen potential functions over the cliques in the structure G. In this section we give several examples on how we can *numerically* define potential functions for three different structure of MRF: univariate, bivariate and multivariate.

4.3.1 Univariate structure

A univariate structure represents no interaction between variables in X. The graph G will be an edge less graph (see Figure 2.10 in chapter 2). Therefore, the set of maximal cliques, C, in G would consist of n singleton cliques $C_i = \{X_i\}$. For each clique, C_i, we can associate a potential function as follows:

$$u_i(x_i) = \alpha_i x_i \tag{4.10}$$

From (4.8) the MFM can then be written as:

$$-ln(f(x)) = U(x) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n \tag{4.11}$$

In terms of jpd (4.1), it can also be written as

$$p(x) = \frac{e^{-\sum_{i=1}^{n} \alpha_i x_i}}{Z} \tag{4.12}$$

where,

$$Z = \sum_{x \in \Omega} e^{-\sum_{i=1}^{n} \alpha_i x_i} \tag{4.13}$$

Here, α_i is the parameter associated with each clique $\{X_i\}$. α_i being the only unknown parameter of the potential function (4.10), completely specifies the $U(x)$ and therefore completely specifies the Gibbs distribution (4.12). Therefore, they are also known as *MRF parameters* (Li, 1995). We use θ to refer to vector of all MRF parameters in the model. For univariate case, the vector $\theta = \alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. In terms of MFM, (4.8), an MRF parameter measures the effect that the interaction between variables in a clique have on the fitness of the solution, $f(x)$. Obviously, in univariate case (4.11), α_i measures the effect of a single variable, X_i, on fitness.

4.3.2 Bivariate structure

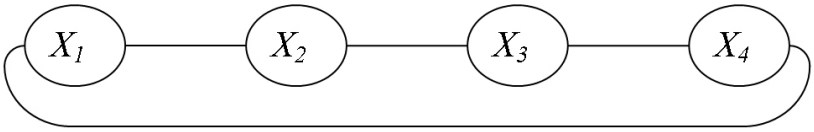


Figure 4.1: An undirected graph showing a chain model of interaction between 4 variables

The bivariate structure represents the pair-wise interaction between variables. For example in Figure 4.1, a bivariate structure of a *chain* formation on four random variable is shown. The set of maximal cliques, C, in this case, contains four cliques: $C_1 = \{X_1, X_2\}$, $C_2 = \{X_2, X_3\}$, $C_3 = \{X_3, X_4\}$ and $C_4 = \{X_4, X_1\}$. In general, for any bivariate clique $C_i = \{X_i, X_j\}$, we can define the potential function as:

$$u_{ij}(x_i, x_j) = \beta_{ij} x_i x_j \tag{4.14}$$

The MFM, can then be written as:

$$-ln(f(x)) = U(x) = \beta_{12} x_1 x_2 + \beta_{23} x_2 x_3 + \beta_{34} x_3 x_4 + \beta_{41} x_4 x_1 \tag{4.15}$$

In general, MFM for a chain graph with n nodes and with above defined potential function will be

$$-ln(f(x)) = U(x) = \beta_{12} x_1 x_2 + \beta_{23} x_2 x_3 + ... + \beta_{n1} x_n x_1 \tag{4.16}$$

In terms of jpd, it can also be written as

$$p(x) = \frac{e^{-\sum_{i=1}^{n} \beta_{i(i+1)} x_i x_{i+1}}}{Z} \tag{4.17}$$

where, $i + 1 = 1$ if $i = n$.

The MRF parameters, $\beta_{i(i+1)}$, in this case represents the combined effect of interaction between two variables X_i and X_{i+1}.

Let us define two types of MFM.

Definition 4.3.1 (Minimal MFM) *We define a Minimal MFM as the MFM where the potential functions in $U(x)$ are defined on all the maximal cliques and not on any of their sub-cliques.*

Definition 4.3.2 (Complete MFM) *We define Complete MFM as MFM where the potential functions in $U(x)$ are defined on all the maximal cliques, their sub-cliques including singleton cliques.*

Equation (4.15) is an example of a minimal MFM for the chain graph shown in Figure 4.1.

Now, let us define the complete MFM for the same graph. In order to do so, we need to further define the potential functions for all sub-cliques in G. In Figure 4.1, there are all together n singleton sub-cliques $\{X_i\}$ in G. For each $\{X_i\}$, we define potential function as

$$u_i(x_i) = \alpha_i x_i \tag{4.18}$$

Adding this to (4.16), we get the following formulation for MFM

$$-ln(f(x)) = U(x) = \sum_{i=1}^{n} \alpha_i x_i + \beta_{i(i+1)} x_i x_{i+1} \tag{4.19}$$

where, $i + 1 = 1$, if $i = n$. Here, the vector of MRF parameters θ will consist of two sets:

α and β. Equation (4.19) is an example of a complete MFM.

4.3.3 Multivariate structure

The multivariate structure represents the interactions of order more than 2. Obviously the MFM for such structure will be more complex than the one for univariate and bivariate structure.

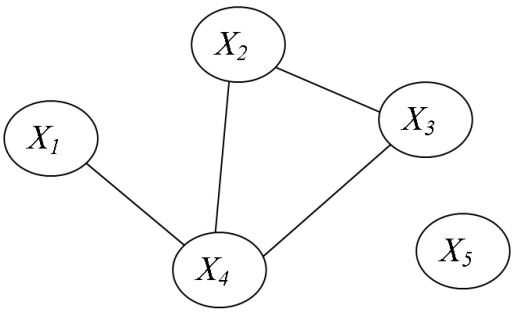


Figure 4.2: An undirected graphical network showing a multivariate model of dependency between 5 variables

As an example, in Figure 4.2, a structure of an MRF on 5 random variables with interactions of order up to 3 is shown. The MFM can then be written as

$$\begin{aligned} -ln(f(x)) = U(x) \quad &= \quad \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 + \alpha_5 x_5 + \\ &\quad \beta_{14} x_1 x_4 + \beta_{23} x_2 x_3 + \beta_{24} x_2 x_4 + \beta_{34} x_3 x_4 + \\ &\quad \gamma_{234} x_2 x_3 x_4 \end{aligned} \tag{4.20}$$

Here, $\alpha_i x_i$, $\beta_{ij} x_i x_j$ and $\gamma_{ijk} x_i x_j x_k$ are the potential fiunctions associated with cliques of order 1, 2 and 3 respectively. Therefore, the vector of MRF parameters, $\theta = \{\alpha, \beta, \gamma\}$.

Equation (4.20) is a complete MFM for Figure 4.2. However, there may be other simplified MFM (for e.g. a minimal MFM) that could be similarly defined. In this thesis we will only consider either minimal or complete MFM.

4.4 Estimating the parameters of MRF

Once we define the potential function for the given structure of MRF, next step is to estimate the parameters of the MRF, θ. In EDAs, we can do so by fitting the derived MFM to a dataset (i.e. set of solution), D. Let us explain it in more detail.

Each solution in a given population provides an equation satisfying the MFM with defined potential functions, where MRF parameters will be the unknown part. Applying this to a set of solution D consisting of N solutions therefore allow us to estimate MRF parameters, θ, by solving the system of equations:

$$F = A\theta^T \tag{4.21}$$

Here, F is the column vector containing $-\ln(f(x))$ of all solutions in D, θ, is the vector of all MRF parameters [3], and A is the matrix of values in D.

To clarify this, let us give an example. For simplicity, we assume the univariate structure for an MRF with clique potential function defined as $u_i(c_i) = \alpha_i x_i$. The jpd, $p(x)$, can then be defined as (4.12) and corresponding *univariate MFM* can be defined as

$$-ln(f(x)) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n \tag{4.22}$$

A solution, x, in the set D will then provide an equation satisfying (4.22). Where, left hand side of the equation will be the $-ln(f(x))$ and the right hand side will be the sum over the product of each $x_i \in x$ with the MRF parameter α_i. α_i is the unknown part of the equation. Note that, for mathematical reason, {-1,1} should be used as the values for x_i rather than $\{0, 1\}$. This ensures the arithmetical symmetry between possible values of x_i and is a standard practice in MRF modelling techniques. Applying (4.22) to the whole set, D, therefore allows us to estimate MRF parameters, α, by solving the system of equations:

$$F = A\alpha^T \tag{4.23}$$

[3] θ^T is the transpose of vector θ to make it a column vector

Here, F is the N-dimensional column vector containing $-\ln(f(x))$ for the set of solutions in D. A is the $N \times n$ dimensional matrix of allele values in the set D, $\theta = \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is the vector of MRF parameters. Depending on the relationship between N and n, the system will be under-, over-, or precisely-specified. A standard least square fitting algorithm can be used to give a estimation of the α_i. We state one of the most stable algorithm for this purpose known as Singular Value Decomposition (SVD) (Press *et al.*, 1993). SVD can give useful results even when the system of linear equations are under-specified or over-specified.

Above example can be similarly extended for bivariate and multivariate structures. The size of the matrix A will depend on the number of MRF parameters in θ and the size of the set D. For example, if we consider a complete MFM for the chain model (Figure 4.1), the size of the matrix will be $N \times s$, where s, the length of θ, is $2n$ as θ will contain both $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and $\beta = \{\beta_{12}, \beta_{23}, \ldots, \beta_{n1}\}$.

Adding a constant in the system of equations

While solving the system of equations, it is suggested to add a constant (also known as the *intercept*) to the equation. It is a standard approach in statistics and widely used for doing regression analysis. In this case, the system of linear equation will have following form:

$$F = A\theta^T + C \tag{4.24}$$

where, C is known as the intercept of the equation

Figure 4.3 shows a set of solutions, D, and the corresponding set of linear equations for univariate MFM (4.22) that includes the constant C.

The least square fitting will then use the intercept to balance the error in fitting the system of equations resulting in better estimation of MRF parameters. Figure 4.4 shows a graphical illustration of least square fitting with a constant and without a constant. Here, each point represents an equation in two dimensional spaces and the straight line between

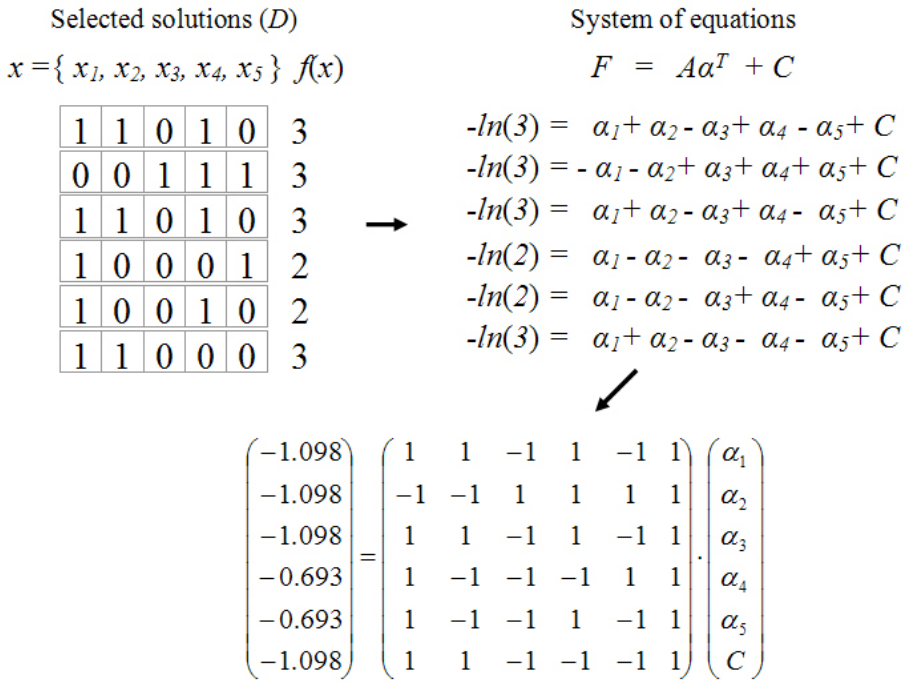


Figure 4.3: A set of solutions D and the corresponding set of linear equations including the intercept C for univariate MFM

them represents the fitted model. It can be noticed that the use of constant significantly reduces the error in fitting and therefore improves the quality of the MRF parameters.

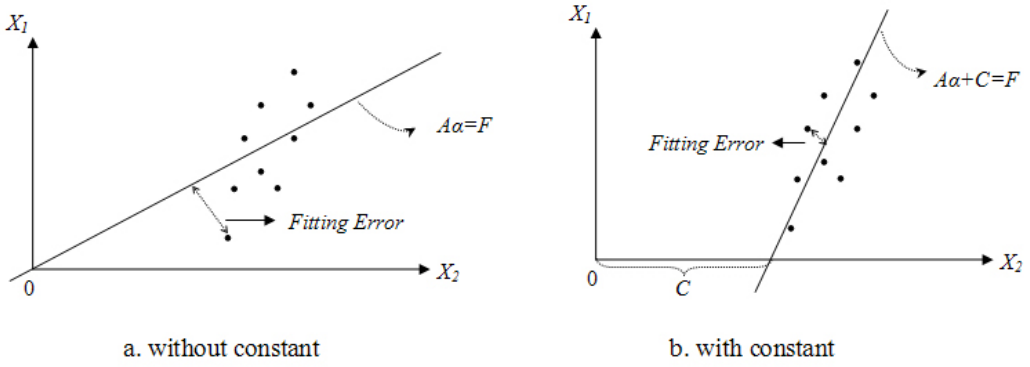


Figure 4.4: Graphical illustration of the effect of adding a constant while solving a system of linear equations

For all the experiments presented in this thesis, intercept has been used while solving the system of equations.

4.5 Summary

Our approach to EDAs is inspired by using Markov Random Field (MRF) to estimate and sample the jpd, $p(x)$. In chapter 3, we reviewed three approaches to estimating and sampling $p(x)$ based on MRF: junction tree, junction graph and Kikuchi approximation. In this chapter, we have presented the *fitness modelling* approach to estimating the MRF in EDAs. The novelty, here is that we model the fitness function to learn the parameters for the MRF.

We started by presenting the concept of equivalence relationship between MRF and the Gibbs distribution. We used this relationship to propose a model of the fitness function which we call MRF fitness model (MFM). MFM defines relationship between the energy, $U(x)$, of the Gibbs distribution and the fitness, $f(x)$, of the solution. We then presented the way to represent different MRF structure in terms of potential function in $U(x)$ and finally presented the way to learn the MRF parameters by using MFM. As MRF parameters completely defines the jpd, $p(x)$, it can be sampled to generate the instances of the X. In the subsequent chapters, we will propose several different techniques to sample the MRF.

Chapter 5

DEUM algorithm and a probability vector approach to sampling

In this chapter, we present a general framework of an EDA, which we call Distribution Estimation Using Markov Random Field (DEUM). DEUM uses MRF approach to estimate and sample the probability distribution. In chapter 4, we have proposed a fitness modelling approach to estimate the MRF in EDAs. Here, we introduce a probability vector approach to sample MRF. We also present a DEUM algorithm with probability vector approach to sampling. For simplicity, we use univariate model of probability distribution. This allows us to completely eliminate the structure learning task.

The chapter is structured as follows. We start by introducing a general framework for DEUM algorithm. We then describe how we can use the probability vector approach to sample MRF. We then present the workflow of DEUM with probability vector approach to sampling, followed by the experimental results. Finally we conclude this chapter by presenting some discussion on the experimental results.

5.1 DEUM: A general framework

The workflow of Distribution Estimation using Markov Random Field (DEUM) is similar to that of other EDAs. It starts by initialising a population of solution, P, then selects a set of promising solutions D from P. The fitness modelling approach is then used to estimate the MRF parameters from D, which is then sampled to generate new solutions. Figure 5.1 shows the general workflow of DEUM.

Distribution Estimation using MRF (DEUM)

1. Generate parent population P
2. Select a set of solutions D from P
3. Build a MFM and fit it to D to estimate a MRF.
4. Sample MRF to generate new solutions
5. If termination criteria is not satisfied, replace parent with new solutions and go to step 2

Figure 5.1: The pseudo-code of the Distribution Estimation Using MRF (DEUM) algorithm

An important property of DEUM, which distinguishes it from other EDAs is that, it builds a model of fitness function, (MFM), to estimate the distribution. Whereas other EDAs only builds a model of good solutions for this purpose. As we shall show in subsequent part of this thesis, this property of DEUM has an important implication to their performance. In the remaining part of this thesis, we propose several variants of DEUM using different techniques to sample MRF.

5.2 Using probability vector for maintaining and sampling the distribution

The concept of using a probability vector for sampling is not new in EDAs. In chapter 2, we reviewed PBIL (Baluja, 1994), where a vector of probabilities $p = \{p_1, p_2, ..., p_n\}$ is initialised, updated and sampled through out the iteration of the algorithm (see chapter 2, Figure 2.11 for more on PBIL workflow). Each $p_i \in p$ is a probability of $x_i \in x$ to be 1. Each p_i has the value of 0.5 in the initialisation stage. In each iteration, p is sampled to generate solutions for the new population. Then, depending upon the frequency distribution of 1 or 0 in selected set D, p_i is increased or decrease. The compact Genetic Algorithm (cGA) also implements the probability vector approach to sampling (see chapter 2, Figure 2.13 for cGA workflow).

Here, we present a similar approach based on maintaining and sampling a probability vector, p. Although, the general idea behind our approach is similar to that used in PBIL and cGA, it has its differences. Instead of using the frequency distribution of 0 or 1 in D, we use the MRF parameters θ to update p. Unless and otherwise stated, we assume that the optimisation criteria for any EDA is to maximise the fitness $f(x)$. Also the binary representation {0, 1} for x_i is replaced by {-1, 1}.

Let us recall the MFM with univariate model of energy $U(x)$ (4.22) derived in previous chapter:

$$-ln(f(x)) = U(x) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n$$

Here, each $\alpha_i x_i$ is a potential function defined over clique $\{X_i\}$. Each α_i can be seen as the contribution measurer of variable X_i towards the $U(x)$. As stated earlier, here, minimising $U(x)$ would be equivalent to maximising $f(x)$. In order to minimise $U(x)$, each contribution $\alpha_i x_i$ should be minimised. Now, once we approximate the α_i, we can then use it to predict the value for corresponding x_i that minimises the total contribution $\alpha_i x_i$. More precisely, a negative α_i indicates that x_i is more likely to be 1, so that the total contribution $\alpha_i x_i$ will be negative. Similarly, for same reason, a positive α_i indicates

that x_i is more likely to be -1.

This information would allow us to generate a chromosome that would be an *optimal* chromosome for the current generation. However, most of the time, current optimal chromosome does not imply the global optimum. There are two main reasons for this. Firstly, there may simply not be enough information in a population to predict the global optimal chromosome at once. Secondly, the problem domain considered may not have the exact (or similar) interaction between variables as assumed by the probabilistic model. Therefore, a better way to use this information is to update the probability vector towards the direction pointed by MRF parameters and sample them to generate next population.

We use following *updating rule* to update $p_i \in p$ towards the direction pointed by α_i

$$\begin{aligned} \text{For} \quad & i = 1..n \ \text{ do} \\ & \text{If } \ \alpha_i < 0 \ \text{ then } \ p_i = p_i(1-\lambda) + \lambda; \\ & \text{If } \ \alpha_i > 0 \ \text{ then } \ p_i = p_i(1-\lambda); \end{aligned} \tag{5.1}$$

Here, λ is a *learning rate* parameter, similar to the one introduced in PBIL, and takes a value between 0 and 1. In (5.1), if α_i is negative, we increase the probability, p_i, of corresponding x_i being 1 in the next generation towards the fixed learning rate λ. Similarly, if α_i is positive, we decrease p_i. Sampling such p_i, then allows us to generate more of the x_i with the value pointed by α_i in next generation. This, in turn, allows us to minimise $U(x)$ as generation progresses. Notice that when $\lambda = 1$, p_i tends to its limit, $\{0, 1\}$, depending on the sign of α_i.

This forms the basis for a DEUM algorithm that uses probability vector approach to sampling. Notice that DEUM with probability vector approach to sampling was first introduced in (Shakya *et al.*, 2004b) as an initial DEUM algorithm. Here we refer to it as DEUM$_{pv}$.

5.3 DEUM$_{pv}$: A DEUM with probability vector approach to sampling

The workflow of the DEUM with probability vector approach to sampling (DEUM$_{pv}$) is similar to that of other univariate EDAs such as PBIL, UMDA and cGA. It begins by initialising a probability vector, $p = \{p_1, p_2, .., p_n\}$, where each p_i is assigned the value 0.5. p is then sampled to generate the parent population P with M solutions. The set D consisting of N best solutions is then selected from P. MRF parameters are then calculated by fitting the univariate MFM (4.22) to D. Fitting is done by solving the system of linear equation (4.24). *Singular Value Decomposition* (SVD) (Press *et al.*, 1993) technique is used to solve the system of linear equations. The MRF parameters, α_i, are then used to update the probability vector p, which is then sampled M times to create a new population. This process continues until termination criteria are satisfied. Pseudo-code for DEUM$_{pv}$ is shown in Figure 5.2.

DEUM with probability vector approach to sampling (DEUM$_{pv}$)

1. Initialise probability vector $p = \{p_1, p_2, ..., p_n\}$ by assigning 0.5 to each p_i
2. Sample p to generate parent population P consisting of M solutions
3. Select a set D from P consisting of N fittest solutions, where $N \leq M$
4. Estimate MRF parameters $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ by applying univariate MFM on D and solving the system of linear equations
5. Use α to update p using following updating rule
   For $i = 1..n$ do
   If $\alpha_i < 0$ then $p_i = p_i(1 - \lambda) + \lambda$;
   If $\alpha_i > 0$ then $p_i = p_i(1 - \lambda)$;
6. Go to step 2 until termination criteria satisfies

Figure 5.2: The pseudo-code of DEUM with probability vector approach to sampling (DEUM$_{pv}$) algorithm

The learning rate, λ, (shown in Figure 5.2) has a direct effect on the convergence speed of DEUM$_{pv}$ where convergence is slow if λ is closer to 0 and convergence is fast if λ is closer

to 1. As stated in the previous section, to calculate α the binary representation {0, 1} of the solution is replaced by {-1, 1}.

Unlike the updating rules used in UMDA, PBIL and cGA, the DEUM$_{pv}$ updating rule uses the sign of the MRF parameter to direct the search towards favouring a particular value of x_i. As stated earlier, this is achieved by updating the probability vector p_i in the appropriate direction by a fixed learning rate λ. Note that in the extreme case when selection size $N = 1$, the DEUM$_{pv}$ updating rule will always have an identical effect to the PBIL updating rule. This is because, in order to make the $\sum_{i=1}^{n} \alpha_i x_i$ negative, solving a system of equation consisting of a single solution using SVD will always result in negative α_i for $x_i = 1$ and positive α_i for $x_i = -1$. This will make sure that each individual contribution $\alpha_i x_i$ in (4.22) will be negative and therefore the total contribution will also be negative, i.e., $-ln(f(x))$. The DEUM$_{pv}$ updating rule with such α will have identical effect to the PBIL updating rule on single solution.

Also, because updating rule (5.1) in DEUM$_{pv}$ is aimed at minimising $U(x)$ to maximise $f(x)$, it is essential to maintain the negative relationship between fitness and energy while fitting MFM to D. In other words, the $-ln(f(x))$ for all x in D should be ≤ 0. This is easily achieved on the problems we considered and held true on all runs of each of the experiments presented in the next section. For problems where $-ln(f(x)) \leq 0$ is likely to occur in D,i.e., if $f(x)$ is between 0 and 1, DEUM$_{pv}$ should be applied by adding 2 to the $f(x)$ for all x in D, while building the system of linear equations.

5.4 Experiments and Results

The aim of our experiment is to test if the fitness modelling approach to estimating the distribution used in DEUM$_{pv}$ has an advantage over the traditional frequency counting approach used in other univariate EDAs. Therefore, we compare the performance of DEUM$_{pv}$ with the performance of other univariate EDAs in different optimisation problems. We also find it informative to compare DEUM$_{pv}$ with a GA. We show that, for the problems we have tested, the performance (in term of number of fitness evaluation taken to find

the solution) of DEUM$_{pv}$ is significantly better than the performance of other univariate EDAs and a GA.

5.4.1 Methodology

As a test set, we choose three problems that have been widely used in EDA community to evaluate the performance of different EDAs. They are: Onemax problem (Mühlenbein & Paaß, 1996), Schaffer f6 Function (Davis, 1991), and trap function of order 5 (Pelikan, 2002). We empirically determined the parameters for DEUM$_{pv}$. For the rest of the algorithms, we used parameter settings from the literature or empirically determined parameters depending on which proved best for particular problems. The number of fitness evaluation taken by an algorithm to find the optimum solution has been taken as the measure of performance.

For Onemax problem, we perform the *scalability analysis*, which shows how an algorithm scales up (in terms of number of fitness evaluation) to a particular problem as the problem size grows. For Schaffer f6 Function, we use *Run Length Distribution* (RLD) curve (Hoos & Stutzle, 1999) to show the performance of each algorithm. RLD curve shows, for each algorithm, the cumulative percentage of successful runs that has been terminated in a certain number of fitness evaluation. RLD is a powerful technique used to understand the dynamic behaviour that is usually observed in stochastic algorithms like GAs and EDAs. For example, Figure 5.5 shows that, with DEUM$_{pv}$, 50% of runs found the optimum within 1700 function evaluations in comparison to 2200 function evaluations of PBIL.

In the subsequent chapters of this thesis, we will frequently use both scalability test and RLD in order to compare the performance of different algorithms.

5.4.2 Test problems

Onemax Problem

This is a simple linear problem which can be defined as follows.

$$f_{om}(x) = \sum_{i=1}^{n} x_i \tag{5.2}$$

Where, $x_i \in \{0, 1\}$ is the value of i^{th} variable of set x and the fitness is the sum of all bits in x. Therefore, the global optimum is located at the point (1,1,.....,1). This problem has the univaiate structure and therefore is an ideal problem for univariate EDAs. It has been shown that UMDA works very well on this problem even with a very small selection size (Mühlenbein & Paaß, 1996).

We compare performance of DEUM$_{pv}$ against a simple GA with uniform crossover (GA (uniform)) and two variants of UMDA: UMDA with selection size $N = 0.5M$ (UMDA (0.5)) and UMDA with selection size $N = 0.3M$ (UMDA (0.3)). 100 runs of each algorithm were executed for a series of Onemax problems of size ranging between 30 and 180. The number of fitness evaluations taken to find the optimum solution was recorded for each run. Uniform crossover with exchange probability of 0.5 was used for the GA (uniform), crossover was applied all the time and mutation was not applied. Population size M ranged from 40 -100 for GA (uniform), 50 - 170 for both variants of UMDA and was set to be $1.5n$ for DEUM$_{pv}$ (45 - 270). Learning rate λ was set to 1 for DEUM$_{pv}$. Truncation selection with selection size $N = 0.5M$ was used for GA (uniform). For DEUM$_{pv}$ whole population was selected, i.e., $D = P$ was used. No elitism was used and new populations were generated with complete replacement.

Figure 5.3 shows the scalability of each algorithm over the range of Onemax problems. The success ratio of converging to the optimum was 100% for DEUM$_{pv}$, 99% for UMDA (0.5), 98% for UMDA (0.3) and 100% for GA (uniform). GA with one point crossover is not shown in the experiment as its performance was much worse than the algorithms shown in Figure 5.3.

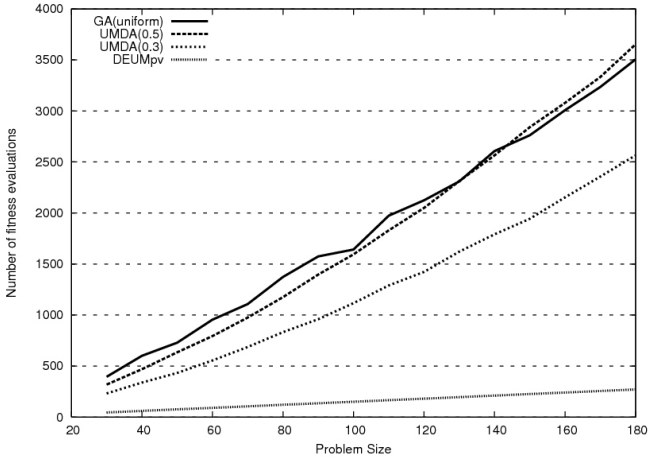


Figure 5.3: Average number of fitness evaluations for 30 to 180 sized Onemax problem where the population size was 40 -100 for GA (uniform), 50 - 170 for both variant of UMDA and $1.5n$ for DEUM$_{pv}$ which is 45 - 270. λ for DEUM$_{pv}$ was 1

As we can see from Figure 5.3, GA (uniform) as expected has a performance comparable to UMDA (0.5). UMDA (0.3) performs better than both GA (uniform) and UMDA (0.5). Finally, DEUM$_{pv}$ has the performance significantly better than that of rest of the algorithms and equals to about $1.5n$ fitness evaluations for all n. We found that MRF parameters, α, estimated from the initial population of $1.5n$ solutions accurately predicts the optimum solution. Therefore, by setting the learning rate to $\lambda = 1$, i.e, by converging each element of probability vector, p_i, to their extreme, DEUM$_{pv}$ was able to find the solution in single generation. In other words, the first solution sampled from such probability vector was the optimum solution.

Schaffer f6 function

The Schaffer f6 function, described in (Davis, 1991), is an interesting function for optimisation, which has been frequently used to evaluate the performance of GAs. A simplified version of it is presented below.

$$f_6(x) = 1 + \left(\frac{\cos(x)}{1 + 0.001x^2}\right) \tag{5.3}$$

Where, $-300 \leq x \leq 300$.

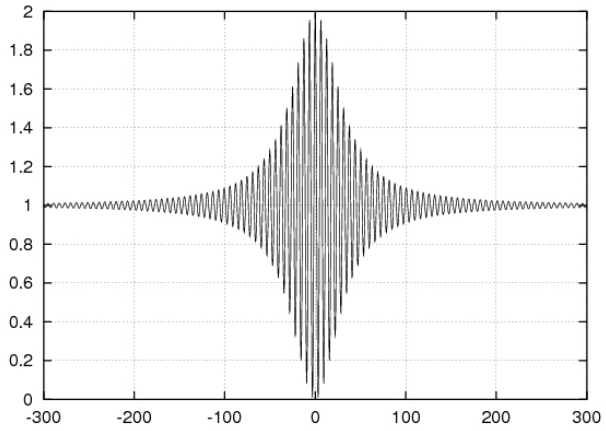


Figure 5.4: Fitness landscape for simplified version of Schaffer f6 function

An interesting feature of this function is that it has lots of local optima but a single global optimal solution (Figure 5.4). So a hill climbing algorithm will rapidly become trapped in one of the local optima. The optimal solution is $f_6(x) = 2$ when $x = 0$.

We performed experiments with two different 20-bit representations of the f6 function. Firstly with a simple binary representation, and secondly with a gray coded representation. Each algorithm with fixed parameter settings was run for a total of 1000 runs. For each run the number of evaluations taken to find the optimum solution was recorded.

For the binary representation, the parameter settings were as follows. For UMDA, the population size was 400, selection size was $N = 0.3M$ and 50% elitism was used. For GA (uniform), population size was 200 and truncation selection with selection size $N = 0.5M$ was used. Crossover was applied all the time, mutation was set to 0.01 and 50% elitism was used. For PBIL and DEUM$_{pv}$, identical parameter settings were used: population size was 160, learning rate λ was set to 0.15, selection size $N = 2$ was used. Mutation shift was not applied in PBIL. Within the limits of representational accuracy, the termination criterion was effectively $f(x) > 1.99999988079071$.

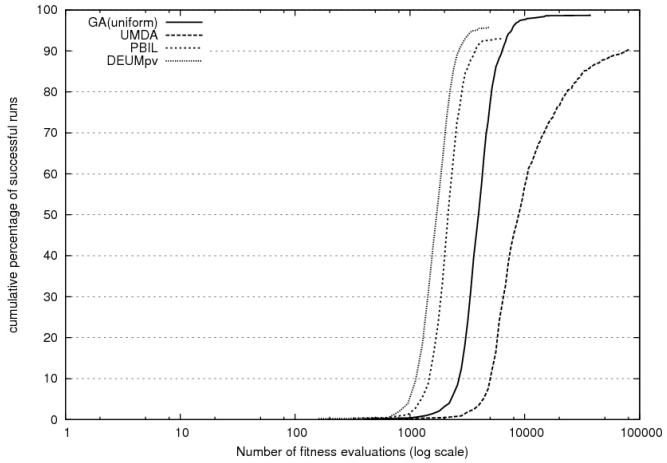


Figure 5.5: Experimental results in the form of RLD showing, for each algorithm running on the 20-bit binary representation of Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

For the gray code representation, the parameter settings were as follows. For GA (uniform), population size was 300 and truncation selection with selection size $N = 0.5M$ was used. Crossover was applied all the time, mutation was set to 0.01 and 50% elitism was used. For PBIL and DEUM$_{pv}$, identical parameter settings were used: population size was 500, learning rate λ was set to 0.1, selection size $N = 2$ was used. Mutation shift was not applied in PBIL.

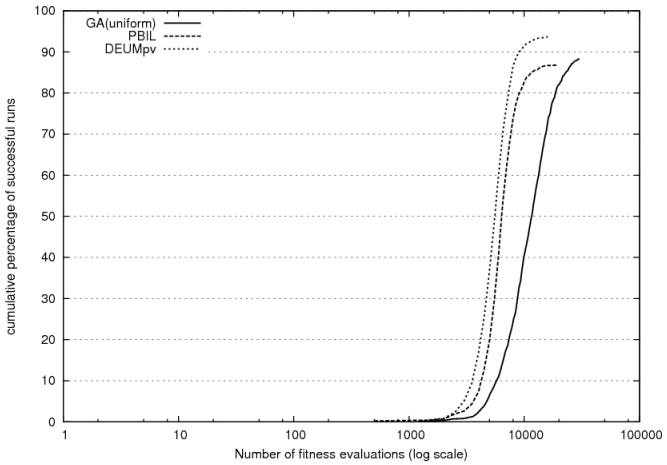
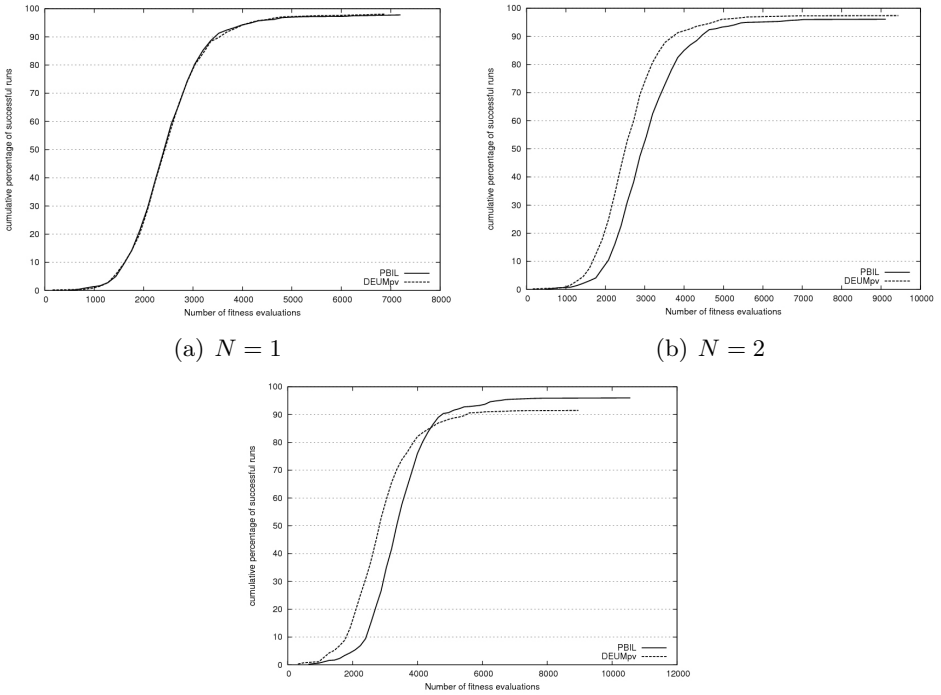


Figure 5.6: Experimental results in the form of RLD showing, for each algorithm running on the 20-bit gray code representation of Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

The experimental results for binary representation in the form of RLD is shown in Figure 5.5. The experimental results for gray code representation in the form of RLD is shown



(a) $N = 1$ (b) $N = 2$

(c) $N = 3$

Figure 5.7: Experimental results in the form of RLD comparing DEUM$_{pv}$ and PBIL for 20-bit binary representation Schaffer f6 function. For both algorithms population size was 160, learning rate was 0.1 and selection size, N, was 1 for (a), 2 for (b) and 3 for (c).

in Figure 5.6. The result for UMDA is not shown in the Figure 5.6, as its performance was much worst than other three algorithms. We can see that, for both binary and gray representations, the performance of DEUM$_{pv}$ was better than other univariate EDAs.

As we stated earlier, the performance of DEUM$_{pv}$ was identical to the performance of PBIL when selection size N was set to 1 (Figure 5.7a). The updating rule in this case is effectively identical. However, as selection size increases, the performance of DEUM$_{pv}$ exceeds that of PBIL (Figure 5.7b and 5.7c).

trap function of order 5

A Trap function of order k (Pelikan, 2002) can be defined as

$$f_{trap,k}(x) = \sum_{i=1}^{n/k} trap_k(x_{b_i,1} + ... + x_{b_i,k}) \tag{5.4}$$

Each block $(x_{b_i,1} + ... + x_{b_i,k})$ gives a fitness which can be calculated through a general trap function of order k

$$trap_k(u) = \begin{cases} f_{high}, & \text{if} \quad u = k \\ f_{low} - u\frac{f_{low}}{k-1}, & \text{otherwise} \end{cases}$$

Where, u is the number of ones in the input block of k bits. The trap function of order 5 (Figure 5.8) is the instance of the general trap function where $k = 5$, $f_{high} = 5$ and $f_{low} = 4$.

The important feature of a trap function is that the block of bits with $u < k$ has decreasing fitness as u increases and so misleads the algorithm away from the global optimum. The purpose of this experiment is to show that DEUM$_{pv}$, like other univariate EDAs and GA with uniform crossover, is misled by a trap function because it cannot detect the interaction between variables.

For GA (onepoint) crossover probability was 100%, mutation probability was 1%, population size was 600 and 50% elitism was used. For all experiments, truncation selection with $N = 0.5M$ was used except that the single best selection was used for PBIL.

Experiments done with problem size $n = 30$ showed that none of GA (uniform), UMDA, PBIL with learning $\lambda = 0.1$ or DEUM$_{pv}$ with different settings of λ could find the optimum even using a population size of 15000. However a simple GA with one point crossover (GA (onepoint)) could find the solution using average of 7500 fitness evaluations.

5.5 Discussion

Our experiments show that, for univariate problems, the use of MRF parameters instead of the simple univariate marginal frequency provide significantly better performance in terms of the number of function evaluations required by the algorithm to find the optimum solution.

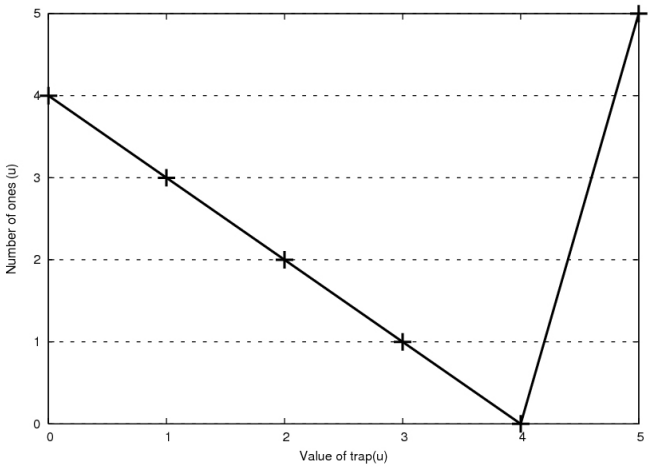


Figure 5.8: The trap function of order 5, where global optimum is in 11111 and local optimum is in 00000. Any block of bits with $u < 5$ deceives algorithm to the local optimum as u increases.

Our experiments with trap function of order 5 showed that DEUM$_{pv}$, as we expect, has a similar problem as other univariate EDAs, that is to converge to the local optimum. This is obvious, because the univariate model of probability distribution considered in DEUM$_{pv}$ cannot properly represent the deceptive nature of interaction between variables in the trap function.

5.6 Summary

In this chapter we have presented a general framework of an EDA using MRF approach to estimation of distribution, which we call Distribution Estimation using MRF (DEUM). The motivation behind DEUM is to model the fitness function to learn the MRF parameters. We have also presented a probability vector approach to sampling the MRF. Two of the previously proposed EDAs, known as PBIL and cGA, also use probability vector approach to sampling. However, our approach differs from the approach used in these two algorithms, as, instead of using marginal frequency, we use MRF parameters to update probability vector. We have also presented a DEUM algorithm that uses such approach to sample the MRF. We called it DEUM$_{pv}$.

Our experimental results show that, for most of the problems tested, the performance

of DEUM$_{pv}$ was better than that of other univariate EDAs. Specifically, for univariate problems, such as Onemax, DEUM$_{pv}$ was able to find the optimum in a single generation. It also did not require any *explicit* selection operator, such as truncation or tournament selection, for Onemax optimisation. Ability to perform optimisation without using any explicit selection operatures is an important property of DEUM algorithms, which also plays a significant role in their performance. We will explain this property in detail in the next chapter, where we also present another version of DEUM algorithm based on direct sampling from the Gibbs distribution.

Chapter 6

DEUM$_d$: direct sampling from Gibbs distribution

In the previous chapter, we proposed DEUM as a general framework of an EDA using fitness modelling approach to estimating the MRF parameters. We also introduced DEUM$_{pv}$: a version of DEUM using probability vector approach to sampling the MRF. In this chapter, we present a DEUM, which directly samples from the Gibbs distribution. We call it the DEUM with direct sampling of Gibbs distribution (DEUM$_d$). DEUM$_d$ has been first introduced in (Shakya *et al.*, 2005a,b).

The chapter is structured as follows. In next section we describe how to estimate marginals from the Gibbs distribution. We then present the workflow of the DEUM$_d$ algorithm. This will be followed by the experimental results on the performance of DEUM$_d$ on a range of different optimisation problems. We then present the analysis of our experimental findings, and also present the cost-benefit analysis of using the fitness approximation approach to estimating MRF in EDAs. Finally, we conclude this chapter by presenting an interesting property of DEUM algorithms, i.e., their ability to perform optimisation even without using an *explicit* selection operator.

6.1 Finding marginals from the Gibbs distribution

Let us recall the model of distribution used by DEUM$_{pv}$.

$$p(x) = \frac{e^{-U(x)/T}}{Z} = \frac{e^{-\sum_{i=1}^{n} \alpha_i x_i/T}}{Z} \tag{6.1}$$

Where, $Z = \sum_{y \in \Omega} e^{-U(y)/T}$ is a normalising constant called partition function, Ω is a set of all possible instances of X. T is a *temperature* constant and $U(x) = \sum_{i=1}^{n} \alpha_i x_i$ is the *energy function* defined as a sum of potential functions, $\alpha_i x_i$, over all $X_i \in X$.

We use x^+ to denote x having a particular $x_i = +1$, similarly, we use x^- to denote x having $x_i = -1$. The probability that the variable in position i is equal to 1, $p(x_i = 1)$, can then be written as

$$p(x_i = 1) = \frac{p(x^+)}{p(x^+) + p(x^-)} \tag{6.2}$$

Substituting $p(x)$ from (6.1) and cancelling the Z, we get

$$p(x_i = 1) = \frac{e^{-U(x^+)/T}}{e^{-U(x^+)/T} + e^{-U(x^-)/T}} \tag{6.3}$$

or,

$$p(x_i = 1) = \frac{1}{1 + e^{(U(x^+) - U(x^-))/T}} \tag{6.4}$$

As $U(x^+)$ and $U(x^-)$ agree in all terms other than $\alpha_i x_i$, the common terms in both $U(x^+)$ and $U(x^-)$ drop out and we get the following expression as the estimate of the marginal probability for $x_i = 1$:

$$p(x_i = 1) = \frac{1}{1 + e^{\beta \alpha_i}} \tag{6.5}$$

where, $\beta = 2/T$.

Similarly, we can get following expression as the estimate of the marginal probability for $x_i = -1$:

$$p(x_i = -1) = \frac{1}{1 + e^{-\beta \alpha_i}} \tag{6.6}$$

6.1.1 Role of temperature in sampling a Gibbs distribution

Temperature T has a very important role in Gibbs distribution. It controls the *convergence* of the distribution. In equation (6.5), as $T \to 0$, the value of β increases, and the value of $p(x_i = 1)$ tends to limit depending on the α_i. If $\alpha_i > 0$, then $p(x_i = 1) \to 0$ as $T \to 0$. Conversely, if $\alpha_i < 0$, then $p(x_i = 1) \to 1$ as $T \to 0$. If $\alpha_i = 0$, then $p(x_i = 1) = 0.5$ regardless of the value of T. Therefore, the α_i are indicators of whether the allele value at the position i should be 1 or -1. This indication becomes stronger as the temperature is *cooled* towards zero.

This forms the basis for the proposed DEUM$_d$, which combines the sampling of Gibbs distribution with a cooling scheme. We reduce T, i.e., increase β, as the population evolves, so the model becomes more exploitative rather than explorative as the evolution progresses.

6.2 Workflow of DEUM$_d$

DEUM$_d$ begins by initialising a population of solution P. Then the selection process takes place where N best solution is selected from P. MRF parameters, α, are then calculated by fitting the univariate MFM, (4.22), on the selected set of solution. This is achieved by solving the system of linear equations, (4.24). The $p(x_i)$ is then calculated from equation (6.5) and sampled to generate the child population. The child then replaces the parent, P, and this process continues until termination criteria are satisfied.

The five-step procedure for DEUM$_d$ workflow is shown in Figure (6.1).

As described earlier, β has a direct effect on the convergence speed of DEUM$_d$. As the number of iterations, g, grows, the marginal probability, $p(x_i)$, gradually cools to either 0 or 1. However, depending upon the type of problem, different cooling rates may be required. In particular, there is a trade-off between convergence speed of the algorithm

Distribution Estimation using MRF with direct sampling (DEUM$_d$)

1. Generate an initial population, P, of size M.
2. Select set D from P consisting of N fittest solutions, where $N \leq M$.
3. Calculate the MRF parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ by fitting univariate MFM to D.
4. Generate M new solutions using the following distribution:

$$p(x) = \frac{e^{-\sum_{i=1}^{n} \alpha_i x_i / T}}{Z}$$

   where, $p(x_i = 1) = \frac{1}{1+e^{\beta\alpha_i}}$ and $p(x_i = -1) = \frac{1}{1+e^{-\beta\alpha_i}}$. Here, β is defined as $\beta = g\tau$ where, g is the number of the current iteration and $\tau > 0$ is a *cooling rate* parameter chosen by the user.
5. Replace P by the new population, and go to Step 2 until the termination criterion is satisfied.

Figure 6.1: The pseudo-code of the Distribution Estimation Using MRF with direct sampling (DEUM$_d$) algorithm

and the exploration of the search space. Therefore, the cooling rate parameter, τ, has been introduced. τ gives explicit control over the convergence speed of DEUM$_d$. Decreasing τ slows the cooling, resulting in better exploration of the search space. However, it also slows the convergence of the algorithm. Increasing τ, on the other hand, makes the algorithm converge faster. However, the exploration of the search space will be reduced.

6.2.1 Related works

Before going further and describe the experimental setups, we find it informative to describe some related works. The use of the temperature for EDAs has been first proposed in *Boltzmann Estimated Distribution Algorithm* (BEDA) (Mühlenbein *et al.*, 1999), where, a Boltzmann selection has been used to estimate the Boltzmann distribution. Note that, BEDA is a conceptual algorithm as requires sum over exponentially many terms to calculate the distribution (Mühlenbein & Mahnig, 2001b). A cooling schedule for Boltzmann selection for BEDA has been later proposed in Mühlenbein & Mahnig (2001b). This ap-

proach has strong similarities with our approach, however has its differences as well. In BEDA, the fitness has been directly taken as the energy for the Boltzmann distribution, however in DEUM$_d$, a model of fitness function, MFM, is built and fitted to the population.

6.3 Experiments and Results

The aim of our experiment is to compare the performance of DEUM$_d$ to that of other univariate EDAs. For this purpose, a range of optimisation problems from literature has been chosen. Each of these problems has been used in the literature to evaluate different EDAs (see (Baluja & Davies, 1997; Larrañaga *et al.*, 1999; Mühlenbein & Mahnig, 1999b; Pelikan, 2002; de la Ossa *et al.*, 2004)). Some problems are known to be better solved by EDAs and some by GAs. We compare the performance of DEUM$_d$ with that of PBIL and UMDA. We also compare the performance of DEUM$_d$ with GA.

6.3.1 Methodology

Each algorithm was executed for a fixed number of runs and stopped if it matched one of the following three criteria. 1) the optimal solution is found. 2) population converged. 3) maximum number of fitness evaluations performed.

For the problems where optimum solution could be found, the number of fitness evaluations needed by the algorithm was taken as a measure for performance evaluation. Run Length Distribution (RLD)(Hoos & Stutzle, 1999) curves were plotted to measure the performance. RLD shows, for each algorithm, the cumulative percentage of successful runs that terminated within a certain number of function evaluations.

For the problems where the optimum was not known or could not be found, the algorithms were evaluated by the average quality/fitness of solution they could find and the average number of fitness evaluations taken to find it (Larrañaga *et al.*, 1999; de la Ossa *et al.*, 2004). This is shown as a table (eg. see Table 6.4) where the average ± standard

deviation is shown in the first row for fitness and in the second row for the number of fitness evaluation.

The following abbreviations are used hereafter for the parameters of the algorithms involved in the experiments. They are, **PS** for population size M, **SS** for selection size N, **LR** for learning rate, **CR** for cooling rate τ, **CP** for crossover probability, **MP** for mutation probability and **EL** for number of elitist solution to be transferred to the child population. For UMDA, PBIL and DEUM$_d$, *truncation selection* was used, i.e. the best N solutions were selected. For GA, different selection methods were tried for each of the problem and the best performing method was chosen. The rest of the parameters for each algorithm were chosen empirically.

6.3.2 Test problems

Onemax Problem

The Onemax problem, (5.2), described in previous chapter is an ideal problem for univariate EDAs. We take the problem dimension for Onemax to be 180 so the optimum fitness is 180. Each algorithm was executed for total of 1000 runs. The parameter setups for each of the algorithm are shown in Table 6.1:

Table 6.1: Parameter setup for Onemax

	PS	SS	LR	CR	CP	MP	EL
GA	100	-	-	-	1	0.0025	-
UMDA	180	60	1	-	-	-	-
PBIL	40	10	0.3	-	-	-	-
DEUM$_d$	270	270	-	100000	-	-	-

For GA, the truncation selection and the uniform crossover were used. Notice that, a very high cooling rate (CR) is given for the DEUM$_d$. This is because, for Onemax problem, given population size of $1.5n$, MRF parameters, α, estimated from initial population was accurate enough to predict the optimum. Therefore, in order to converge the marginals, $p(x_i)$, to their limit towards the direction pointed by α_i, the temperature had to be near

zero in initial generation, i.e, $T- > 0$ or equivalently $\beta- > \infty$. This is approximated by giving a very high value for cooling rate, CR. The results for each algorithm in the form of RLD curves are shown in Figure 6.2.

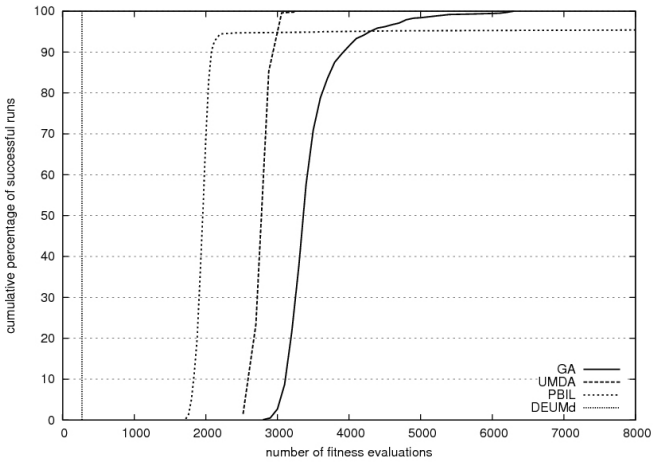


Figure 6.2: Experimental results in the form of RLD showing, for each algorithm running on the 180 bit Onemax problem, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

We can see that, as expected, the performance of DEUM$_d$, both in terms of fitness evaluation and success rate was significantly better than that of other algorithms. For example, Figure 6.2 shows that, with DEUM$_d$ 90% of the runs found solution within around 270 function evaluation in comparison to 20×10^2, 28×10^2 and 40×10^2 fitness evaluation of PBIL, UMDA and GA respectively.

Plateau problem

This problem was proposed in Mühlenbein (1994) and is used by (Larrañaga & Lozano, 2002) to evaluate the performance of EDAs. The individuals of this function consist of a n-dimensional vector, such that $n = m \times 3$ i.e. the genes are divided into groups of three. The plateau function can be defined as:

$$f_p(x) = \sum_{i=1}^{m} g(x_{3i-2}, x_{3i-1}, x_{3i})$$

where,

$$g(x_1, x_2, x_3) = \begin{cases} 1 & \text{if} \quad x_1 = 1 \quad \text{and} \quad x_2 = 1 \quad \text{and} \quad x_3 = 1 \\ 0 & \text{otherwise} \end{cases}$$

The goal is to maximise the function f_p. The global optimum is located at the point (1,1,.....,1). We take the problem dimension n to be 180 so the optimum fitness is 60. Each of the algorithms was executed for 1000 runs and the number of fitness evaluation taken to find the optimum was recorded. The parameter setups are shown in Table 6.2.

Table 6.2: Parameter setup for Plateau

	PS	SS	LR	CR	CP	MP	EL
GA	200	-	-	-	1	0.005	-
UMDA	200	100	-	-	-	-	-
PBIL	40	15	0.2	-	-	-	-
DEUM$_d$	100	20	-	6	-	-	-

For the GA, truncation selection and uniform crossover were used. The results in the form of RLD are shown in Figure 6.3, which shows that for most of the runs the performance of DEUM$_d$, both in terms of fitness evaluation and success rate was better than that of other algorithms. For example, with DEUM$_d$ 80% of the runs found solution within around 33×10^2 fitness evaluation in comparison to 48×10^2, 57×10^2 and 100×10^2 fitness evaluation of PBIL, UMDA and GA respectively.

Checkerboard problem

In Checkerboard problem (Baluja & Davies, 1997; Larrañaga & Lozano, 2002), a $s \times s$ grid is given where each grid can take value 0 or 1. The goal is to create a checkerboard pattern of 0's and 1's on the grid. i.e. each grid with a value 1 should be surrounded in all four basic directions by a value of 0, and vice versa. The fitness function is the number of bits with the correct neighbours. Let, $x = [x_{ij}]_{i,j=1,...s}$ be the grid and $\delta(a, b)$ be the

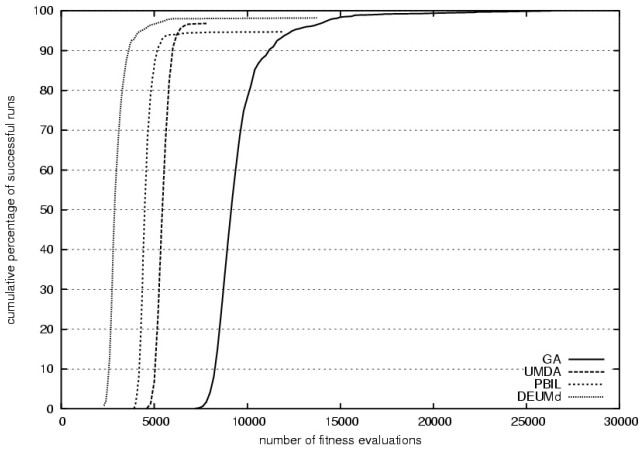


Figure 6.3: Experimental results in the form of RLD showing, for each algorithm running on the 180 bit Plateau problem, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

Kronecker delta function. Then the checkerboard function can be written as:

$$f_{cb}(x) = 4(s-2)^2 - \sum_{i=2}^{s-1}\sum_{j=2}^{s-1}\{\delta(x_{ij}, x_{i-1,j}) +$$

$$\delta(x_{ij}, x_{i+1,j}) + \delta(x_{ij}, x_{i,j-1}) + \delta(x_{ij}, x_{i,j+1})\}$$

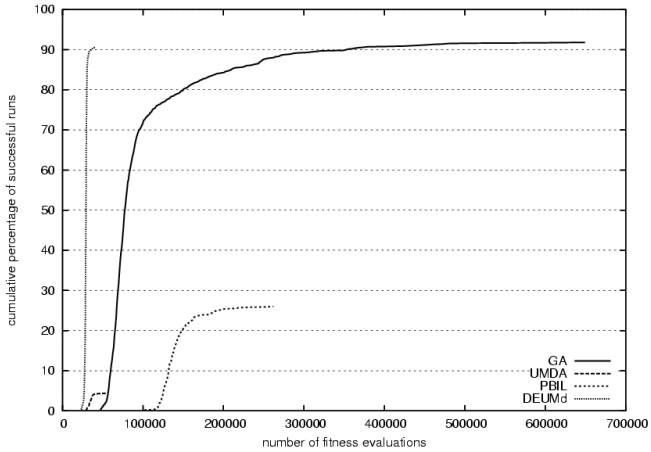


Figure 6.4: Experimental results in the form of RLD showing, for each algorithm running on the 100 bit CheckerBoard problem, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

We follow the approach taken by Larrañaga & Lozano (2002); de la Ossa *et al.* (2004) and

use $s = 10$ so the dimension is 100. The optimum fitness in this case will be 256. Each algorithm was run for total of 1000 runs. The parameter setups for each of the algorithm are shown in Table 6.3:

Table 6.3: Parameter setup for Checkerboard

	PS	SS	LR	CR	CP	MP	EL
GA	1024	-	-	-	0.6	0.01	2
UMDA	1024	500	-	-	-	-	-
PBIL	100	10	0.01	-	-	-	-
DEUM$_d$	100	10	-	0.4	-	-	-

For the GA, truncation selection and onepoint crossover were used. The results in the form of RLD are shown in Figure 6.4. As we can see, the percentage of successful runs was low ($< 95\%$), therefore the mean and standard deviation for fitness and the number of evaluation are also shown in Table 6.4.

Table 6.4: mean ± stdev of fitness and number of fitness evaluation for each algorithm on Checkerboard problem

	GA	UMDA	PBIL	DEUM$_d$
fitness	254.68 ± (4.39)	233.79 ± (9.2)	243.5 ± (8.7)	254.1 ± (5.17)
evaluation	427702.2 ± (1098959.3)	50228.2 ± (9127)	191476.8 ± (37866.95)	33994 ± (13966.75)

Figure 6.4 shows that the success rate for PBIL and UMDA was very poor, with 5% and 25% respectively, in comparison to over 90% of both DEUM$_d$ and GA. Figure also shows that, with DEUM$_d$, 90% of the runs found the optimum within 41×10^3 fitness evaluation in comparison to 350×10^3 fitness evaluation of GA.

Schaffer f6 function

The Schaffer f6 function, described in Davis (1991), is an interesting function for optimisation that has been frequently used to evaluate the performance of GAs. Let us recall

the simplified version of it from (5.3):

$$f_6(x) = 1 + \left(\frac{\cos(x)}{1 + 0.001x^2} \right)$$

where $-300 \leq x \leq 300$.

The optimal solution is $f_6(x) = 2$ when $x = 0$. We performed experiments with a 20-bit representation of the f6 function. Within the limits of representational accuracy, the termination criterion was effectively $f_6(x) > 1.99999988079071$. Each algorithm was run for total of 1000 runs. The parameter setups are shown in Table 6.5:

Table 6.5: Parameter setup for F6 function

	PS	SS	LR	CR	CP	MP	EL
GA	200	-	-	-	1	0.01	2
UMDA	400	120	-	-	-	-	-
PBIL	160	2	0.15	-	-	-	-
DEUM$_d$	200	2	-	8	-	-	-

For the GA, truncation selection and uniform crossover were used. The experimental results in the form of RLD are shown in Figure 6.5, which shows that for most of the runs the performance of DEUM$_d$, both in terms of fitness evaluation and success rate was better than that of other algorithms. For example, Figure 6.3 shows that, with DEUM$_d$ 80% of the runs found solution within around 17×10^2 fitness evaluation in comparison to 27×10^2, 230×10^2 and 50×10^2 fitness evaluation of PBIL, UMDA and GA respectively.

Equal products function

This problem is presented in Baluja & Davies (1997); de la Ossa *et al.* (2004). Given a set of n random real numbers $\{a_1, a_2, ..., a_n\}$ from an interval $[0, k]$, a subset of them is selected. The aim of the problem is to minimise the difference between the products of the selected and unselected numbers. This can be written as:

$$f_{ep}(x) = \left| \prod_{i=1}^{n} h(x_i, a_i) - \prod_{i=1}^{n} h(1 - x_i, a_i) \right|$$

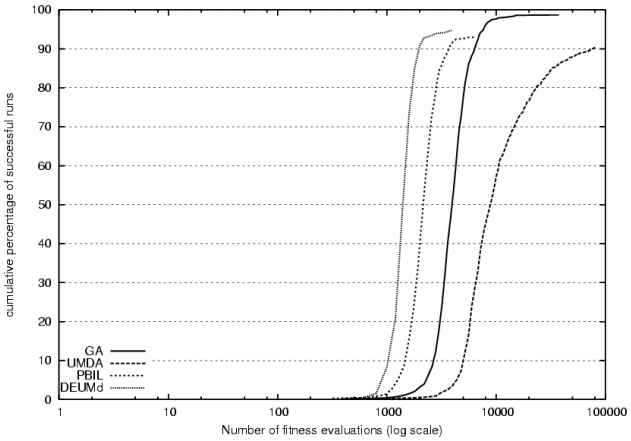


Figure 6.5: Experimental results in the form of RLD showing, for each algorithm running on the 20 bit binary code representation of Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

where,

$$h(x,a) = \begin{cases} 1 & \text{if} \quad x = 0 \\ a & \text{if} \quad x = 1 \end{cases}$$

The optimum value is unknown as the real numbers a_i are generated randomly. However the optimum should be close to zero. We take the problem dimension to be 50. Following de la Ossa *et al.* (2004), the random numbers are taken from the interval [0,4]. Each algorithm was run for total of 100 runs (each time with a random instance of a). The parameter setups for each of the algorithm are shown in Table 6.6:

Table 6.6: Parameter setup for Equal products

	PS	SS	LR	CR	CP	MP	EL
GA	500	-	-	-	0.6	0.01	100
UMDA	500	250	-	-	-	-	100
PBIL	500	250	0.5	-	-	-	100
DEUM$_d$	1000	12	-	0.01	-	-	1

Table 6.7: mean ± stdev of fitness and number of fitness evaluation for each algorithm on Equal products problem

	GA	UMDA	PBIL	DEUM$_d$
fitness	211.59 ± (1058.47)	5.03 ± (18.29)	9.35 ± (43.36)	2.14 ± (6.56)
evaluation	1000000 ± (0)	1000000 ± (0)	1000000 ± (0)	1000000 ± (0)

For the GA, truncation selection and uniform crossover were used. Since, the optimum for this problem was not known and was different for each instance of the problem, the RLD could not be shown. Results are shown in Table 6.7. We can see that the average fitness found by DEUM$_d$ was better than that found by rest of the algorithms. The standard deviation for DEUM$_d$ was also less than that for other algorithms. This shows that the DEUM$_d$ is more predictable and has consistent performance.

Colville function

This is a minimisation problem defined in Michalewicz (1996), and also used by de la Ossa *et al.* (2004). The function can be defined as

$$f_c(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 +$$

$$10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$$

where, $-10 \leq x_i \leq 10$.

We have taken the solution length to be 60 where, each x_i is represented as a 20 bit binary string. The optimum value for f_c is 0. 100 independent runs of each algorithm were executed for this problem. The parameter setups for each of the algorithm are shown in Table 6.8:

Table 6.8: Parameter setup for Colville

	PS	SS	LR	CR	CP	MP	EL
GA	500	-	-	-	0.8	0.01	-
UMDA	1024	512	-	-	-	-	-
PBIL	500	1	0.005	-	-	-	-
DEUM$_d$	1000	1	-	0.01	-	-	-

For the GA, tournament selection and onepoint crossover were used. The table below (Table 6.9) shows the mean $\pm$ standard deviation for fitness and number of evaluation for each of the algorithms. We can see that, DEUM$_d$ and GA had similar performance with

both finding solutions closer to the global optima. However, the standard deviation for DEUM$_d$ was slightly better than that of GA.

Table 6.9: mean ± stdev of fitness and number of fitness evaluation for each algorithm on Colville problem

	GA	UMDA	PBIL	DEUM$_d$
fitness	0.61 ± (1.02)	40.62 ± (102.26)	2.69 ± (2.54)	0.61 ± (0.77)
evaluation	1000000 ± (0)	62914.56 ± (6394.58)	1000000 ± (0)	1000000 ± (0)

SixPeaks function

The SixPeaks function (Baluja & Davies, 1997; Larrañaga & Lozano, 2002) can be mathematically defined as

$$F_{sp}(x,t) = max\{tail(0,x), head(1,x), tail(1,x), head(0,x)\} + R(x,t)$$

where,

$$tail(b,x) = \text{number of tailing } b's \text{ in } x$$

$$head(b,x) = \text{number of leading } b's \text{ in } x$$

$$R(x,t) = \begin{cases} n & \text{if } \; tail(0,x) > t \;\text{ and }\; head(1,x) > t \;\text{ or} \\ & \quad tail(1,x) > t \;\text{ and }\; head(0,x) > t \\ 0 & \text{otherwise} \end{cases}$$

The goal is to maximise the function. This function has 4 global optima which are isolated and therefore are difficult to find. It also has two local optima which are easy to get and therefore the search algorithms tend to converge on local optima. We have taken the dimension to be 100 and t to be 30, thus the optimum fitness value is 169. Each algorithm was run for total of 100 runs. The parameter setups for each of the algorithm are shown in Table 6.10:

Table 6.10: Parameter setup for SixPeakes

	PS	SS	LR	CR	CP	MP	EL
GA	50	-	-	-	0.6	0.01	2
UMDA	1024	512	-	-	-	-	-
PBIL	100	30	0.1	-	-	-	-
DEUM$_d$	40	4	-	0.3	-	-	2

For the GA, truncation selection and uniform crossover were used. As expected the univariate EDAs were not able to find the global optima as they were deceived towards the local optima. This result applies to DEUM$_d$ as well. Mean ± standard deviation of fitness value and number of evaluation for each algorithm are shown in Table 6.11.

Table 6.11: mean ± stdev of fitness and number of fitness evaluation for each algorithm on SixPeaks problem

	GA	UMDA	PBIL	DEUM$_d$
fitness	99.1 ± (9)	98.58 ± (3.37)	99.81 ± (1.06)	100 ± (0)
evaluation	49506 ± (4940)	121333.76 ± (14313.44)	58210 ± (3659.15)	26539 ± (1096.45)

Trap function of order 5

Let use recall the Trap function of order k (Pelikan, 2002) defined in previous chapter (5.4).

$$f_{trap,k}(x) = \sum_{i=1}^{\frac{n}{k}} trap_k(x_{b_i,1} + ... + x_{b_i,k})$$

Each block $(x_{b_i,1} + ... + x_{b_i,k})$ gives a fitness which can be calculated through general trap function of order k

$$trap_k(u) = \begin{cases} f_{high}, & if \quad u = k \\ f_{low} - u\frac{f_{low}}{k-1}, & otherwise \end{cases}$$

Here, u is the number of ones in the input block of k bits. The trap function of order 5 is an instance of the general trap function where $k = 5$, $f_{high} = 5$ and $f_{low} = 4$. The important feature of a trap function is that the block of bits with $u < k$ has decreasing fitness as u increases and so misleads the algorithm away from the global optimum. We take the problem dimension to be 60. Each algorithm was run for total of 1000 runs. The

parameter setups for each of the algorithms are shown in Table 6.12:

Table 6.12: Parameter setup for trap5

	PS	SS	LR	CR	CP	MP	EL
GA	1500	-	-	-	1	0.01	2
UMDA	30000	15000	-	-	-	-	-
PBIL	30000	1	0.1	-	-	-	-
DEUM$_d$	2000	20	-	0.1	-	-	2

For the GA, tournament selection and onepoint crossover were used. The results in the form of RLD are shown in Figure 6.6.

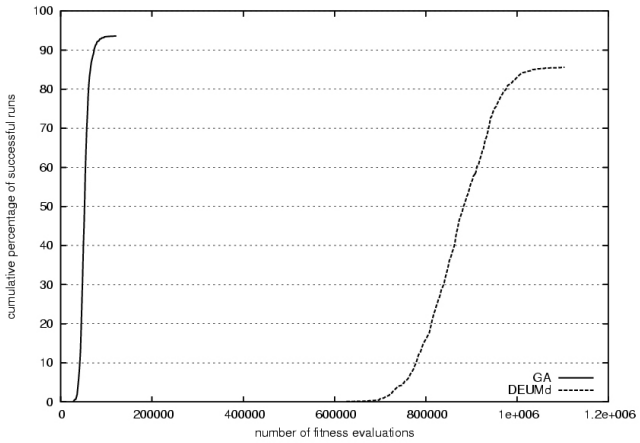


Figure 6.6: Experimental results in the form of RLD showing, for each algorithm running on the 60 bit Trap function of order 5, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

Both, PBIL and UMDA, were not able to find the solution and therefore their results are not present in the Figure 6.6. As expected, GA with onepoint crossover was able to find the solution. Interestingly, DEUM$_d$ was also able to find the solution, however, with very high fitness evaluation.

6.4 Analysis of Results

Our experimental results show that, DEUM$_d$ gives satisfactory results for most of the problems that we have tested and fails where we would expect it to. We perform a statistical analyse on the significance of the results presented above by using a t-test.

For the univariate problems (such as Onemax), by setting the population size to $1.5n$ and by setting $D = P$, DEUM$_d$, as with DEUM$_{pv}$, was able to find the optimum solution in a single generation.

For problems with low order of dependency between variables (such as plateau and checker board) the performance of DEUM$_d$ (in terms of number of fitness evaluations taken to terminate) was significantly better than that of other univariate EDAs and also of the GAs tested. This can be verified from the t-test comparison (on number of fitness evaluations) shown in the Table 6.13. Here, the p-values are shown for each comparison on each problem. All p-values are $<< 0.05$. This indicates that the difference in algorithms' performance originates from their respective effectiveness rather than from random noise.

Table 6.13: Results of the t-test comparison of number of fitness evaluation on problems with lower order dependency

	DEUM$_d$ vs. PBIL	DEUM$_d$ vs. UMDA	DEUM$_d$ vs. GA
Onemax	0.000	0.000	0.000
Plateau	0.000	0.000	0.000
Checkerboard	0.000	0.000	0.000

For the problems with higher order and deceptive dependency (such as SixPeaks and Trap of order 5), DEUM$_d$, as with other univariate EDAs was deceived by the structure of fitness landscape. This can be clearly seen from the Table 6.11 for SixPeaks and Figure 6.6 for Trap function. For the SixPeaks function, none of the algorithm could find optimum solution. For the trap function, UMDA and PBIL could not find the optimum, even using a population size of 30000. However, a simple GA with one-point crossover could find the solution after an average of 62000 fitness evaluations. Interestingly, DEUM$_d$ with population size of 2000 could also find the solution, however, with a very large average fitness evaluation, 868000. It shows that, although DEUM$_d$ is misled by trap function, by slowing the cooling rate and choosing the correct population size, it still could overcome a trap of order 5. Because of the low quality of results, the t-test was not applied to test the significance.

For those problems where the optimum was not known (or was difficult to get) (Colville

and Equal products), the performance of DEUM$_d$ was comparable to that of GA and other univariate EDAs and was better in some cases (see Tables 6.7 and 6.9). These results can be verified from the t-test comparison (on quality of fitness) shown in Table 6.14. Here, the p-values are shown for each comparison on each problem. Although the mean fitness for the DEUM$_d$ was better than that for rest of the algorithms on the Equal products function, the p-values shows that this result is not significant in comparison to PBIL and UMDA (as p-values are > 0.05), but it is significant in comparison to the GA. Similarly, for Colville function, the results for DEUM$_d$ are not significant in comparison to GA but are highly significant in comparison to PBIL and UMDA.

Table 6.14: Results of the t-test comparison of quality of fitness for Colville and Equal products function

	DEUM$_d$ vs. PBIL	DEUM$_d$ vs. UMDA	DEUM$_d$ vs. GA
Equal products	0.103	0.139	0.05
Colville	0.000	0.00016	0.974

6.5 Comparing DEUM$_d$ with DEUM$_{pv}$

So far in this chapter, we have given the extensive experimental analysis comparing the performance of DEUM$_d$ with that of other univariate EDAs and GA. However, we find it informative to give results comparing the performance of DEUM$_d$ with its previous version, DEUM$_{pv}$. In the experimental results section (Section 5.3) of chapter 5, we have shown the graphs comparing the performance of DEUM$_{pv}$ to that of other univariate EDAs on three different problems. Here, we re-plot those graphs together with the performance of DEUM$_d$ added to it.

Onemax Problem

For Onemax problem, the performance of DEUM$_{pv}$ and DEUM$_d$ was exactely the same. For both algorithms, the first solution sampled from MRF parameters, α, calculated from the initial population of $1.5n$ solutions, and with $D = P$, was found to be the optimum

solution. Therefore, the number of fitness evaluation for both algoriothm was exactely $1.5n + 1$. Figure 5.3 in previous chapter shows the scalibility for DEUM$_{pv}$ on 30 to 180 sized Onemax problems, which can also be seen as the scalability for DEUM$_d$.

Schaffer f6 function

Figure 6.7 is the update to the Figure 5.6 showing the added RLD for DEUM$_d$ on 20-bit gray coded Schaffer f6 function. Here, DEUM$_d$, as did DEUM$_{pv}$, also used the population size of 500 and the selection size of 2. The cooling rate τ was 1.5. For the detail on the parameter setting of rest of the algorithm, please see section 5.3.1. We can see that, although the success rate for finding a solution for DEUM$_d$ was 92% compared to 94% of DEUM$_{pv}$, the number of function evaluations needed to find the solution for DEUM$_d$ was significantly less than that of DEUM$_{pv}$ and other EDAs (shown in log scale in figure 6.7).

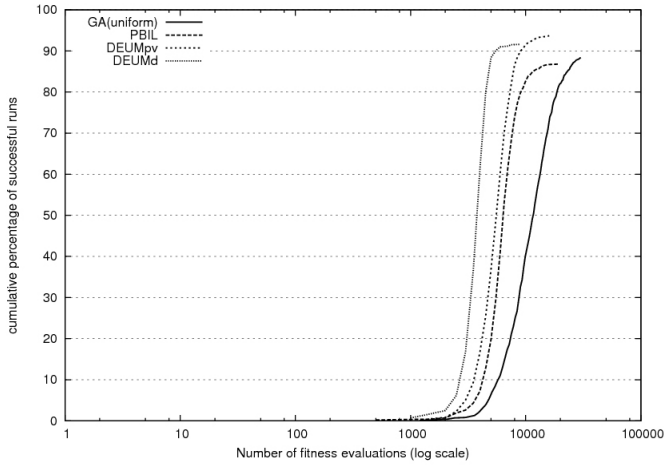


Figure 6.7: RLD for DEUM$_d$ in comparison to DEUM$_{pv}$ and other univariate EDAs, showing, for each algorithm running on the 20-bit gray code representation of Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

For the comparison of DEUM$_d$ and DEUM$_{pv}$ on binary coded Schaffer f6 function, see Figure 6.5 and Figure 5.5, which show that the performance of DEUM$_d$ was better than that of DEUM$_{pv}$ in term of both percentage of successful runs and the number of fitness evaluations.

Trap function of order 5

For the Trap function of order 5, we found that DEUM was not able to find the solution even with the very high number of fitness evaluation [1]. As other univariate EDA, it was deceived by the fitness landscape of the problem resulting in premature convergence to the local optima. In contrast, by slowing the cooling schedule, DEUM$_d$ was able to find the optimum for Trap function of order 5 (see Figure 6.6).

To summarise this section, we state that the results comparing DEUM$_{pv}$ and DEUM$_d$ showed that, for the univariate problems, performance of DEUM$_d$ as expected was similar to the performance of DEUM$_{pv}$. For the rest of the problems tested, DEUM$_d$ outperformed DEUM$_{pv}$. These results indicate that the approach of sampling the Gibbs distribution using a cooling schedule in DEUM$_d$ has an advantage over the approach of updating and sampling a probability vector using a learning rate in DEUM$_{pv}$.

6.6 Cost benefit analysis of using fitness modelling approach to estimating MRF parameters in EDAs

The computational cost of Estimation of Distribution using fitness approximation approach to MRF is of polynomial complexity in comparison to the linear complexity of other univariate EDAs. The reason behind such a high computational cost is mainly because of the SVD technique (Press *et al.*, 1993) used to make the least square approximation of MRF parameters, α, (computational cost of other techniques may vary and are most likely to be cheaper). Assuming $N = n$, computational complexity to compute SVD is $O(n^3)$ (For $N < n$, it is $O(n^2N)$ and for $N > n$, it is $O(nN^2)$) (Golub & Van Loan, 1989; Suda & Kuriyama, 2004), whereas computational complexity to compute the univariate marginal frequency is $O(nN)$. However, our experiments show that there is a case to be made for a more sophisticated estimation of distribution in certain circumstances.

[1] see experimental results section in chapter 5

1. DEUM$_d$ can significantly reduce the number of fitness evaluations required to solve a problem. This will be of particular benefit when fitness evaluation is costly and can be traded off against the computational cost of estimating the distribution.

2. On the problems where only the near optimum solutions could be found, DEUM$_d$ outperformed the other EDAs on quality of solution, often significantly. This suggests that DEUM$_d$ should be tried on problems where the benefit of increased solution quality is likely to outweigh computational cost.

6.7 DEUM algorithms can work even without using an explicit selection operator

So far we have seen that DEUM algorithms, both based on probability vector approach to sampling and direct sampling of Gibbs distribution, can significantly reduce the number of fitness evaluation needed to find the solution. In this section we describe one of the important properties of the DEUM algorithms. That is, their ability to perform optimisation even without using an *explicit* selection process. In other words, DEUM algorithms can exclude the selection operators, such as truncation selection or tournament selection, from their work process, and can use the entire population to estimate and sample the MRF (and still perform the optimisation task). This has been first noted in Shakya *et al.* (2004a). This is a distinct property of DEUM algorithms that differs it from other EDAs, and needs to be understood in order to completely understand the success of these algorithms.

There are various ways to explain this property. However, the key point is as follows: in order to estimate the distribution, DEUM builds a model of fitness function over the entire set of solutions. This contrasts with other EDAs, which build a model of good solutions by only observing the values of each variable in the solution. This allows DEUM to exclude the use of fitness as an explicit way to select solutions. Another way to see this is that, DEUM uses the fitness in the variation process, and therefore does not require

good solutions to be explicitly selected. Let us describe this concept in more detail.

6.7.1 Excluding selection operator from other EDAs

In a typical EDA (or a GA), the fitness is used in the selection phase of evolution in order to explicitly select the set of solutions that takes part in the variation process. Whereas, the variation phase completely depends on the allele of these selected set of solution and does not use (or need) fitness for estimating and sampling probability distribution. This also applies to the crossover and mutation approach to variation in GAs. Without a selection process variation would also fail, i.e., the model of distribution build from the entire population would not be better than the initial random model used to sample the initial population. Therefore, the evolution would not progress towards optimum.

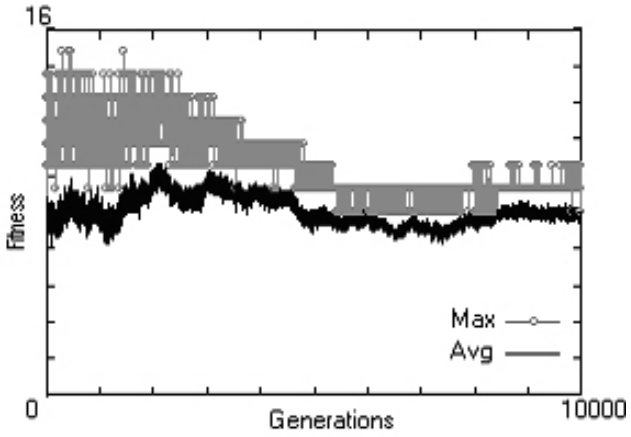


Figure 6.8: A typical run of PBIL without selection pressure on 16 bit Onemax problem showing maximum and average fitness of the population over 10000 generations. Population size is 30, learning rate is 0.1

Let us give as example. In Figure 6.8, the typical evolution process of PBIL over 10000 generation on Onemax problem without using any selection operator is shown. Here, the population, P, consist of 30 solution and all 30 solutions was selected for estimating the distribution, i.e., here $D = P$. The maximum and average fitness of the population is plotted against the number of generation. We can notice that the population is converging to a distribution that is pointing towards equal number of 1 and 0 in the solution. This result is obvious, as the frequency of 0 and 1 in a particular position of a random population

is likely to be equal. However, there will be a small difference in the frequency due to the random noise and therefore each element p_i of the probability vector, p, will be increased or decreased according to such noise. Sampling from such p gives some increase or decrease to number of 1 and 0 in child population. This process continues and over long term, p converges to some random distribution where number of p_i pointing towards 1 and 0 will be almost equal (subject to some random noise). The rate of convergence depends upon the learning rate parameter λ [2]. As bigger the λ will be, as faster the convergence will be. For example, in case of UMDA, which can be seen as an instance of PBIL with learning rate $\lambda = 1$, the convergence will be much faster. This affect is shown in Figure (6.9), where the maximum and average fitness of the population over 150 generation for UMDA is plotted.

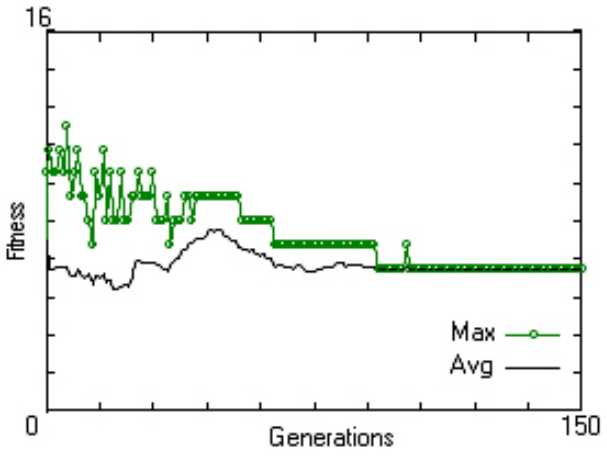


Figure 6.9: A typical run of UMDA without selection pressure on 16 bit Onemax problem showing maximum and average fitness of the population over 10000 generations. Population size is 30

6.7.2 Excluding selection operator from DEUM

Now let us consider the DEUM algorithm and explain what happens if we use entire population to estimate and sample the MRF. In other words what happens if we fit the univariate MFM (or any MFM) to the entire population of solution. The answer is, DEUM will still perform the optimisation. An example is shown in Figure (6.10) where the typical evolution process of DEUM$_d$ over 100 generation on Onemax problem without using any

[2] see Figure 2.11 in chapter 2 for PBIL workflow

selection pressure is shown. Here, the population, P, consist of 30 solution and all 30 solutions was selected for estimating the distribution, i.e., here $D = P$. We see that the population is converging to the optimum solution [3].

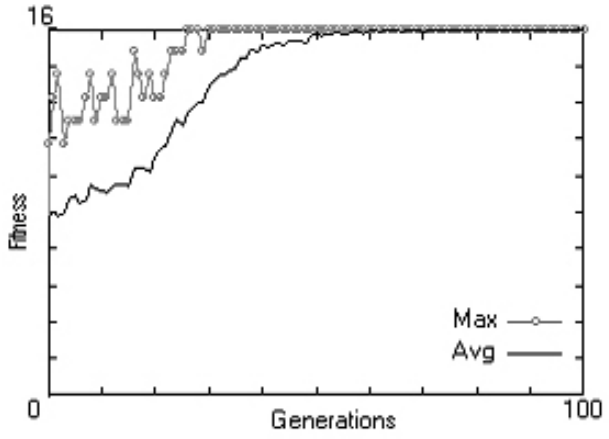


Figure 6.10: A typical run of DEUM$_d$ without selection pressure on 16 bit Onemax problem showing maximum and average fitness of the population over 100 generations. Population size is 30, cooling rate is 1

In order to explain this effect, we first need to recall the MRF parameters α. As we said earlier α completely defines the Gibbs distribution for any MRF, where each α_i are the value associated with x_i. Regardless of whether α is calculated by fitting univariate MFM over entire population of solution or over the fraction of it, we could always suggest the value for x_i that maximises (or minimises) the fitness $f(x)$ by looking on corresponding α_i. Now, the only thing we need to do is to come up with the sampling method that will use this information.

In DEUM$_{pv}$, we have derived the updating rule that updates the probability vector, $p = \{p_1, p_2, .., p_n\}$, depending on α, where, negative α_i increased p_i and therefore increased the probability of $x_i = 1$ in next generation. This increased the probability that combination $\alpha_i x_i$ will be negative, by such increasing the probability of $U(x)$ to be minimum. Similarly, positive α_i decreased p_i and therefore decreased the probability of $x_i = 1$ in next generation. This increased the probability that combination $\alpha_i x_i$ will be negative, by such again increasing the probability of $U(x)$ to be minimum. Therefore, sampling from such p has higher probability of generating x, that will result in minimum $U(x)$.

[3] This graph is also typical for DEUM$_{pv}$

Similarly, in DEUM$_d$, we have assumed jpd for any solution x (as shown in (4.6)) to be

$$p(x) = \frac{f(x)}{Z}$$

It can be noticed that, as bigger the $f(x)$ will be, as higher its probability $p(x)$ will be. In terms of Gibbs distribution, this formulation can be equivalently written as

$$p(x) = \frac{e^{-U(x)/T}}{Z}$$

using which, we get the marginals

$$p(x_i = 1) = \frac{1}{1 + e^{\beta\alpha_i}}$$

(see section 5.2 in this chapter for more detail). As stated earlier, regardless of whether α are estimated from P or D, in order to maximise fitness, a negative α_i indicates that the value of x_i should be 1 and a positive α_i indicates that the value of x_i should be -1. This is reflected in above formulation for marginal probability of $p(x_i = 1)$. Here, $p(x_i = 1) > 0.5$ if $\alpha_i < 0$ and $p(x_i = 1) < 0.5$ if $\alpha_i > 0$. Thus, by sampling from such marginals, we have higher probability of moving towards fitness maximisation.

6.7.3 So is there any need of explicit selection in DEUM ?

The answer is both *yes* and *no* as heavily depends on the problems applied.

On the one hand, using selection may reduce the unnecessary information to be included in the estimation of MRF parameters, therefore may increase the efficiency of the algorithm. This is also proved by the experiment we have presented so far. In most of those experiments, a set D consisting of good solutions from parent P was selected to estimate the MRF parameters. This gave better performance than using $D = P$.

However, the estimation of MRF parameters may get better as the number of dataset grows. In other words, it is better to have as many solutions as possible for estimating

the MRF parameters as each solution contains some unique information about the fitness. In fact, as shown in Figure 6.2 and Figure 5.3, for Onemax problem (5.2), the univariate MFM (4.22) being an exact model of the fitness gives a very efficient result even without using any explicit selection operator. This suggests that, sometimes, it may be better to use entire population for estimation of MRF.

6.8 Summary

In this chapter, we have presented DEUM$_d$: a DEUM algorithm with a direct Gibbs distribution sampling technique. DEUM$_d$ is an update to the DEUM$_{pv}$ algorithm, presented in previous chapter, that used probability vector approach to sampling. We described the use of temperature in Gibbs distribution in order to balance the exploration and exploitation in DEUM$_d$. We presented the extensive experimental analysis on the performance of DEUM$_d$ in a wide range of optimisation problems. Further, we described the cost benefit analysis of using fitness modelling approach to estimation of distribution in DEUM algorithms. Finally, we described an important property of DEUM algorithms, which is their ability to perform optimisation without using any explicit selection operators.

The proposed DEUM$_d$ algorithm is a univariate EDA and is relatively easy to implement. The results show that, it can significantly reduce the number of fitness evaluations needed to find optimum. Together with its better performance in the univariate problems, we also found that it can efficiently solve the problems with simple interactions between variables (such as Plateau and Checkerboard). Furthermore, our results shows that it can also overcome the problem with deceptive interactions, such as trap 5. These results suggest the following conclusion: for any optimisation problem, the first EDA to try is the univariate EDA (Mühlenbein, 1998). There are two main reasons for this, 1) They are easy to implement, and 2) Set of problems that has been shown to be solved by univariate EDA is surprisingly large. Among univariate EDAs, however, DEUM$_d$ shows to be the promising one and therefore could serve as a preferred choice.

Chapter 7

Is-DEUM: A step towards multivariate DEUM

So far in this thesis, we have proposed two variants of DEUM algorithm using a univariate model of probability distribution. They were shown to perform better than other EDAs of their type over a wide range of optimisation problems. This chapter extends DEUM algorithm to use a bivariate model of probability distribution. Therefore, this chapter can be seen as an step forward towards the multivariate DEUM. We present two variants of DEUM that use different sampling technique, 1) Metropolis sampling and 2) Gibbs sampling, and apply them to a well known Ising spin glass problem (Kindermann & Snell, 1980).

Ising spin glass problem has been introduced in early 1920s to model the spin glass system. They have range of practical applications in both statistical physics and AI. Furthermore, the introduction of Ising spin glass problem has played a significant role in the development of the MRF theory itself (Brush, 1967; Preston, 1976). Due to their interesting properties, such as symmetry and a large number of plateaus, they have also been widely studied by the GA (and EDA) community (Pelikan & Goldberg, 2003; Pelikan, 2002; Santana, 2003b, 2005).

The motivation behind choosing the Ising spin glass problem for the bivariate implementation of DEUM algorithm is twofold:

1. Ising spin glass problems have an undirected structure and therefore can naturally be incorporated in the form of cliques and potentials. This eliminates the task of structure learning.

2. Ising spin glass problems have been widely studied by both GA and EDA community and therefore we find it useful to compare the performance of our approach with that of previously proposed ones.

The outline of the chapter is as follows. We start by describing the Ising spin glass problem and give some background on previous applications of EDAs to it. We then present the MRF approach to modelling the Ising spin glass problem and describe how to estimate the model parameters from the population of solutions. We then incorporate this approach in a DEUM, called DEUM with Ising model (Is-DEUM) (Shakya *et al.*, 2006). We then present the experimental results with Is-DEUM using a Metropolis sampling method. This is followed by the experimental results with Is-DEUM using a Gibbs sampling method. Finally we present some discussion of the results and summarise the chapter.

7.1 The Ising spin glass problem and EDAs

The general Ising spin glass problem can be described by an energy function, H, defined over a set of spin variables $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_l\}$ and a set of coupling constants h and J as

$$H(\sigma) = -\sum_{i \in L} h_i \sigma_i - \sum_{i<j \in L} J_{ij} \sigma_i \sigma_j \tag{7.1}$$

Here, each coupling constant $h_i \in h$ and $J_{ij} \in J$ relate to a single spin σ_i and a pair of spins σ_i and σ_j respectively. Each spin variable σ_i can be either +1 or -1. One specific choice of value for the spin variable is called a configuration. L is a lattice of n sites.

Given coupling constants h_i and J_{ij}, the task in the Ising spin glass problem is to find the value for each σ_i that minimises the energy, H. For the purpose of this thesis, we only consider the coupling constants relating pairs of spin variables and therefore set $h_i = 0,\ \forall i \in L$. Additionally, we restrict J_{ij} to take only two values $J_{ij} \in \{+1, -1\}$.

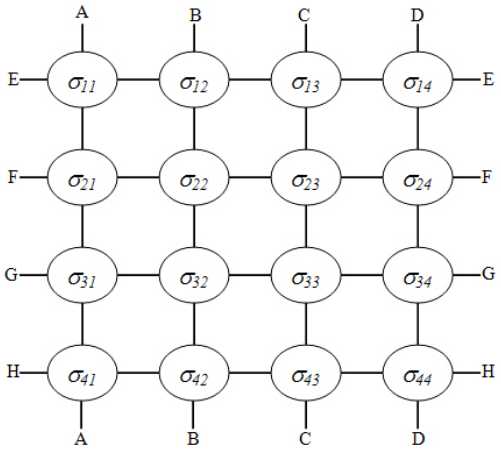


Figure 7.1: A structure showing the interaction between spins for a two dimensional Ising spin glass system with 4×4 spins

Here, we consider the spin glass system on a two dimensional lattice consist of $n = l \times l$ sites, where each spin variable interacts only with its nearest neighbouring variables on a *toroidal* lattice (Figure 7.1). The Hamiltonian specifying the energy for this system can be written as

$$H(\sigma) = -\sum_{i=1}^{l}\sum_{j=1}^{l}\left(J_{ij,(i+1)j}\sigma_{ij}\sigma_{(i+1)j} + J_{ij,i(j+1)}\sigma_{ij}\sigma_{i(j+1)}\right) \tag{7.2}$$

where, $i + 1 = 1$ if $i = l$ and $j + 1 = 1$ if $j = l$.

Here, each $J_{ij,i'j'}$ is the coupling constant in two dimensional lattice relating to spin σ_{ij} and $\sigma_{i'j'}$.

For convenience, we reformulate this as a maximisation problem so seek to maximise

$$-H(\sigma) = \sum_{i=1}^{l}\sum_{j=1}^{l}\left(J_{ij,(i+1)j}\sigma_{ij}\sigma_{(i+1)j} + J_{ij,i(j+1)}\sigma_{ij}\sigma_{i(j+1)}\right) \tag{7.3}$$

In the context of EDAs, spin glass systems on a two dimensional lattice have been of par-

ticular interest to researchers. In particular, Pelikan & Goldberg (2003); Pelikan (2002); Pelikan *et al.* (2004) showed that hBOA could efficiently solve these problems outperforming other algorithms. Santana (2003b) used the Ising spin glass problem as a test problem for the two algorithms MN-EDA and MN-FDA and showed that their performance is better then that of other EDAs based on Bayesian networks. Also (Mühlenbein *et al.*, 1999) stated that, although the two dimensional Ising spin glass problem is in the class of *Additively Decomposable Functions* (ADF), it cannot be efficiently represented as a *junction tree* [1]. This is because, the junction tree based EDA has a *triangular structure* of dependency and therefore requires interaction between variables of order at least 3. However, the two dimensional Ising spin glass problem has a bivariate structure and therefore has a maximum clique of order 2. Santana (2005) argues that the Kikuchi approximation approach to estimate the distribution used by MN-EDA can accurately represent the bivariate dependency, and therefore has an advantage over junction tree based EDAs. This argument applies to the DEUM algorithms as well, as they also can accurately represent the distribution encoded by the structure in the form of potential functions. This is described in detail in next section.

7.2 MRF approach to modelling Ising spin glass problem

Here, we identify the cliques and define the potential function for the Ising spin glass problem. This is then used to define the MFM.

In Figure 7.1, each spin variable, $\sigma_{ij} \in \sigma$, can be seen as a random variable, X_{ij}, in a set, X. Therefore, each solution $X = x$ can be seen as the string representation of the matrix

$$\begin{aligned} x \quad = \quad & \{x_{11}, x_{12}, ..., x_{1l}, \\ & x_{21}, x_{22}, ..., x_{2l}, \\ & \vdots \\ & x_{l1}, x_{l2}, ..., x_{ll}\} \end{aligned} \tag{7.4}$$

[1] Junction tree is described in chapter 3

Here, the total number of variables in X is $n = l^2$. For such x, the fitness function to be maximised is

$$f(x) = \sum_{i=1}^{l}\sum_{j=1}^{l}\left(J_{ij,(i+1)j}x_{ij}x_{(i+1)j} + J_{ij,i(j+1)}x_{ij}x_{i(j+1)}\right) \tag{7.5}$$

Each variable $X_{ij} \in X$ interacts with four of its immediate neighbours. Figure 7.1 can be seen as an undirected graphical structure, G, for X. There are total of $2n$ order 2 cliques in G. For each clique $\{X_{ij}, X_{i'j'}\}$, we assign a potential function $\beta_{ij,i'j'}x_{ij}x_{i'j'}$ and therefore the energy, $U(x)$ in MFM (4.8) for such X will be

$$-ln(f(x)) = U(x) = \sum_{i=1}^{l}\sum_{j=1}^{l}\left(\beta_{ij,(i+1)j}x_{ij}x_{(i+1)j} + \beta_{ij,i(j+1)}x_{ij}x_{i(j+1)}\right) \tag{7.6}$$

In terms of Gibbs distribution it can also be written as

$$p(x) = \frac{e^{-\sum_{i=1}^{l}\sum_{j=1}^{l}\left(\beta_{ij,(i+1)j}x_{ij}x_{(i+1)j}+\beta_{ij,i(j+1)}x_{ij}x_{i(j+1)}\right)/T}}{Z} \tag{7.7}$$

Here, each $\beta_{ij,i'j'}$ is the MRF parameter associated with bivariate clique $\{X_{ij}, X_{i'j'}\}$. It is important to distinguish between $\beta_{ij,i'j'}$ in $U(x)$ with $J_{ij,i'j'}$ in $f(x)$. $\beta_{ij,i'j'}$ is a real valued parameter of the model and will be estimated from a set of solutions. This contrasts with the coupling constants $J_{ij,i'j'} \in \{-1, 1\}$. We use β to denote the set of all $2n$ bivariate MRF parameters $\beta_{ij,i'j'}$. Equation (7.6) is the minimal MFM for the two dimensional Ising spin glass problem.

As stated in chapter 4, depending upon the number and order of cliques considered, we may construct different MFMs from a single graph G. For example, in addition to potential functions $\beta_{ij,i'j'}x_{ij}x_{i'j'}$ for order 2 cliques $\{X_{ij}, X_{i'j'}\}$, we can also assign a potential function, $\alpha_{ij}x_{ij}$, to each singleton clique $\{X_{ij}\}$. The energy for the resulting MFM can be written as

$$-ln(f(x)) = U(x) = \sum_{i=1}^{l}\sum_{j=1}^{l}\left(\alpha_{ij}x_{ij} + \beta_{ij,(i+1)j}x_{ij}x_{(i+1)j} + \beta_{ij,i(j+1)}x_{ij}x_{i(j+1)}\right) \tag{7.8}$$

We use α to denote the set of all n univariate parameters α_{ij}. Equation (7.8) is the complete MFM for the two dimensional Ising spin glass problem.

We use θ to denote the full set of parameters for either MFM.

7.3 Learning MRF parameters for Ising spin glass problem

Once we construct the MFM, finding the parameter of the model is a straight forward task. As described in chapter 3, (and also implemented in the previous two chapters), the basic idea here is to use a set of solutions D to approximate the parameters, θ, of the MRF. Each solution in a given population provides an equation satisfying the MFM (4.8). Selecting a set of solutions D consisting of N promising solutions from a population P therefore allows us to estimate the distribution by solving the system of equations (4.24).

For the minimal MFM, (7.6), F will be an N dimensional column vector containing $-ln(f(x))$ of the solutions in D, θ will be a $2n$ dimensional vector of all MRF parameters β and A will be an $N \times 2n$ dimensional matrix, where each element a_{rs} of A is the product of the alleles from r^{th} solution associated with s^{th} parameter of the model. For the complete MFM, (7.8), $\theta = \{\alpha, \beta\}$ will be a vector of $3n$ MRF parameters, as there will be $2n$ parameters in set β and n parameters in set α, and A will be an $N \times 3n$ dimensional matrix accordingly.

As mentioned earlier in (chapter 5, section 5.2.2), solving a system of linear equation is a computationally expensive process. It grows polynomially to the size of the matrix A. As number of parameter in the model grows, the size of the matrix A grows and therefore the cost of estimating the MRF parameter grows.

7.4 Using a Metropolis method to sample MRF

So far we have shown how to construct a MFM for the Ising spin glass problem and use it to approximate the MRF parameters. Once we get the parameters of the model, the jpd, $p(x)$, is completely specified. Therefore, the next step is to sample $p(x)$. In this section we develop a *zero temperature Metropolis method* for this purpose.

7.4.1 Zero Temperature Metropolis method

Metropolis methods are a class of *Markov Chain Monte Carlo* (MCMC) algorithms (Metropolis, 1953) that have been widely used to sample from a probability distribution. It tries to minimise the energy of the Gibbs distribution. In our case, it results in maximisation of fitness (4.8). Here we present a variant which we call *Bitwise Zero-Temperature Metropolis method* (BZTM). Given a set of MRF parameters, θ, calculated from a set of solutions D, it is then possible to sample a new solution, $x^o = \{x_1^o, x_2^o, ..., x_n^o\}$ using the BZTM as shown in Figure 7.2.

Bitwise Zero-Temperature Metropolis method (BZTM)

1. Generate a solution $x^o = \{x_1^o, x_2^o, .., x_n^o\}$ at random.
2. Repeat:
   (a) Set $x^{tmp} = x^o$.
   (b) For $i = 1$ to n
      i. Mutate variable x_i^o to obtain the mutated solution $x^{o\prime}$.
      ii. Set $\Delta U = U(x^{o\prime}) - U(x^o)$.
      iii. if $\Delta U < 0$ set $x^o = x^{o\prime}$.

   :Until $x^{tmp} = x^o$.
3. Terminate with answer x^o.

Figure 7.2: The pseudo-code of the Bitwise Zero-Temperature Metropolis method

For the complete MFM presented in (7.8), ΔU can be determined explicitly from the

following formula:

$$\begin{aligned}\Delta U &= \left(x_{ij}^{o\prime} - x_{ij}^{o}\right)\Big(\alpha_i + \beta_{(i-1)j,ij}x_{(i-1)j}^{o} + \\ &\quad \beta_{ij,(i+1)j}x_{(i+1)j}^{o} + \beta_{i(j-1),ij}x_{i(j-1)}^{o} + \beta_{i(j+1),ij}x_{i(j+1)}^{o}\Big) \end{aligned} \tag{7.9}$$

Similarly, for the minimum MFM presented in (7.6), ΔU can be determined explicitly from the following formula:

$$\begin{aligned}\Delta U &= \left(x_{ij}^{o\prime} - x_{ij}^{o}\right)\Big(\beta_{(i-1)j,ij}x_{(i-1)j}^{o} + \\ &\quad \beta_{ij,(i+1)j}x_{(i+1)j}^{o} + \beta_{i(j-1),ij}x_{i(j-1)}^{o} + \beta_{i(j+1),ij}x_{i(j+1)}^{o}\Big) \end{aligned} \tag{7.10}$$

This significantly reduces the cost of calculating ΔU.

7.4.2 DEUM with the Metropolis method

Now that we know how to sample the MRF parameters, we can formulate DEUM for the Ising spin glass problem (Is-DEUM). As the Is-DEUM described here implements the Metropolis method as the sampling technique, we denote it as Is-DEUM$_m$. (Figure 7.3) shows the workflow of Is-DEUM$_m$.

Is-DEUM with Metropolis sampler (Is-DEUM$_m$)

1. Generate a population, P, of size M at random.
2. Select a set D consisting of N fittest solutions from P, where $N \leq M$.
3. Calculate the MRF parameters θ by fitting MFM to D.
4. Repeat:
   Sample $x^o = \{x_1^o, x_2^o, ..., x_n^o\}$ using BZTM
   :Until R iteration completes Or $f(x^o)$ is optimal/good enough
5. Terminate with answer x^o.

Figure 7.3: The pseudo-code of the DEUM with Metropolis sampling method

Notice that the Is-DEUM$_m$ only has a single generation. Also in step 4 of the algorithm, we repeatedly use BZTM to sample different x^o. We found that by repeatedly sampling the x^o with different random start, the optimum solution was found in first generation (as we shall show in the next section) [2]. This therefore eliminates the necessity of creating a child population [3].

7.4.3 Experiments and Results

Experiments were performed with three different sizes of two dimensional Ising spin glass systems. They were 4×4 ($n = 16$), 6×6 ($n = 36$) and 8×8 ($n = 64$). Four random instances of each problem size were used for the experiment. Each instance was generated by randomly sampling the coupling constant $J_{ij} \in \{+1, -1\}$. The optimum solution for each instance was verified by using the Spin Glass Ground server, provided by the group of Prof. Michael Juenger [4]. The parameters for Is-DEUM$_m$ were chosen empirically.

These experiments are divided into two parts:

1. A performance comparison with other EDAs
2. A performance comparison between complete and minimal MFM

1. Experiments on the performance comparison with other EDAs

The aim of this experiment is to compare the performance of Is-DEUM with that of other EDAs. Mainly, the comparison is made with the results presented in Santana (2005), where the performance of five different EDAs, both using MRF and Bayesian networks, have been presented for similar instances of Ising spin glass problem. Namely, they were MN-EDA (using Kikuchi approximation approach), MN-EDA^f (using Kikuchi approximation with

[2] This has also been illustrated in (Shakya *et al.*, 2005c) for the Onemax problem, where a Zero-Temperature Metropolis algorithm was able to find the solution in single generation

[3] Though, once we know how to sample from the MRFs, any replacement strategies can be used to form a child population (see (Shakya *et al.*, 2005c) for an example)

[4] http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/sgs.html

fixed structure), MN-FDA (using junction graph approach), EBNA$_{k2}$ (using Bayesian network with $k2$ metrics) and MT-FDA (using a mixture of tree model).

We ran 100 independent run of Is-DEUM$_m$ for each of the 12 instances and recorded the number of fitness evaluations needed to find the optimum. The minimal MFM (7.6) was used in Is-DEUM$_m$ and the whole population was selected for estimation of MRF parameters, i.e, we take $D = P$. Therefore, the selection size N was equal to the population size M. To determine M, we started with the minimum number of M needed to make the system of linear equation specified (in case of (7.6) the minimum M is $2n$). Then we gradually increased it until a success rate of over 95% was achieved (in other words, until more than 95 out of 100 runs found the optimum). The resulting M was taken as the population size for that particular instance. The maximum number of allowed repetitions, R, for BZTM was set to 3000. Is-DEUM$_m$ was terminated if the optimum was found or R repetitions of BZTM were done. As, at the end of each BZTM, the fitness evaluation was done in order to calculate $f(x^o)$, the number of fitness evaluations was calculated as the sum of population size and the total repetitions of the BZTM needed before finding the optimum.

Table 7.1 shows the experimental results on the performance of Is-DEUM$_m$ on all 12 instances of the Ising problems.

Table 7.1: Performance of Is-DEUM$_m$ with minimal MFM for 12 instances of Ising spin glass problem

PI	FE	SD	SR	PS
I-16-1	41.53	1.14	100	40
I-16-2	60.44	13.96	100	50
I-16-3	52.57	1.96	100	50
I-16-4	41.66	1.38	100	40
I-32-1	126.19	48.87	100	90
I-32-2	107.45	21.07	100	90
I-32-3	98.59	9.98	100	90
I-32-4	115.66	28.82	100	90
I-64-1	231.66	35.28	100	200
I-64-2	361.87	170.24	100	200
I-64-3	362.6	177.75	100	200
I-64-4	275.66	92.15	100	200

The first column shows the problem instance (PI). The second and third column show the

average number of fitness evaluation (FE) and the corresponding standard deviation (SD) over the 100 runs, the fourth column shows the success rate (SR) and the fifth column shows the population size (PS) used for the corresponding instances.

The performance of Is-DEUM$_m$ was significantly better than that of other EDAs presented in Santana (2005), both in terms of success rate and the number of fitness evaluations needed to find the optimum. In particular, the best EDA reported in Santana (2005) was MN-FDA^f with average fitness evaluation and success rate of 220.17 and 98.5% respectively for $n = 16$, 1586.02 and 95.25% respectively for $n = 36$ and, 6110.8 and 95% respectively for $n = 64$. Whereas, for Is-DEUM, they were 49.05 and 100% respectively for $n = 16$, 111.96 and 100% respectively for $n = 36$ and 296.69 and 100% respectively for $n = 64$. This is a significant improvement in the performance.

2. Experiment on the performance comparison between complete and minimal MFM

The aim of this experiment is to show that, for the Ising spin glass problem, the use of minimal MFM instead of complete MFM does not decrease the quality of the solution, but does reduce the computational cost needed to find the solution.

Table 7.2 shows the experimental results on the performance of Is-DEUM$_m$ using the complete MFM (7.8) on all 12 instances of the Ising spin glass problems.

The experimental setups were similar to that of Is-DEUM$_m$ using the minimal MFM (7.6) described in previous sub-section. The minimum number of population size (PS) needed to make the system of linear equation specified for (7.8) was $M = 3n$, as compared to $M = 2n$ for (7.6). Therefore as we can see from Table 7.2, the optimum population size needed for all 12 instance for Is-DEUM$_m$ with the complete MFM was greater than that needed by Is-DEUM$_m$ with the minimal MFM (shown in Table 7.1). As a result, the number of fitness evaluations for Is-DEUM$_m$ with the complete MFM was greater than that of Is-DEUM$_m$ with the minimal MFM. Also in sixth column of Table 7.2, the ratio

Table 7.2: Performance of Is-DEUM$_m$ with complete MFM for 12 instances of Ising spin glass problem

PI	FE	SD	SR	PS	tr
I-16-1	61.45	1.27	100	60	1.11
I-16-2	67.43	9.20	100	60	1.02
I-16-3	62.64	3.58	100	60	1.02
I-16-4	61.44	1.38	100	60	1.07
I-32-1	161.52	47.22	100	130	1.33
I-32-2	146.52	17.06	100	130	1.27
I-32-3	137.61	8.22	100	130	1.29
I-32-4	148.64	19.79	100	130	1.34
I-64-1	284.80	49.06	100	250	2.10
I-64-2	376.45	132.26	99	250	1.76
I-64-3	334.99	96.76	100	250	1.55
I-64-4	319.86	115.15	100	250	1.87

of extra time (tr) needed by complete MFM in comparison to minimal MFM is shown for each instance. For each instance, the tr is equal to the average time taken by Is-DEUM$_m$ with complete MFM divided by the average time taken by minimal MFM. For $n = 16$ the difference in time is fairly small, however as n grows the tr grows and for $n = 64$ the time taken by complete MFM is almost double to the time taken by minimal MFM. This result is expected, as the computational time to calculate the MRF parameters grows polynomially with the size of the matrix A in the system of linear equations (4.24). Matrix A grows as selection size $N = M$ grows and N grows as the number of MRF parameters in the MFM grows.

This result shows that, for Ising spin glass problems, using the minimal MFM instead of the complete MFM, results in reduced computational cost without losing the quality of the solutions. We will use the minimal MFM for the rest of the experiments presented in this chapter.

7.5 Using a Gibbs sampler to sample MRF

So far we have shown that by sampling the MRF using a Metropolis method, Is-DEUM was able to solve Ising spin glass problem of size $n = 16$, $n = 32$ and $n = 64$. However, for problem sizes of $n = 100$ and higher, Is-DEUM with the Metropolis method was not able

to find the optimum solution. In this section we describe another sampling method known as *Gibbs Sampler* (GS) and incorporate it in Is-DEUM. The aim here is to solve Ising spin glass problems of larger size. We also present experimental results on the performance of this version of Is-DEUM.

7.5.1 Gibbs sampler

As with the Metropolis method, the Gibbs sampler (GS) (Geman & Geman, 1987) is a class of MCMC algorithm that has been widely used to sample probability distributions. In order to explain GS, we first need to define the formulation of conditional probability $p(x_{ij}|N_{ij})$ for each variable, x_{ij}, from the jpd $p(x)$. Here, N_{ij} is the set of neighbouring variables to x_{ij}. For the two dimenssional Ising spin glass problem on toroidal lattice (Figure 7.1), this set consist of

$$N_{ij} = \{x_{(i+1)j}, x_{i(j+1)}, x_{(i-1)j}, x_{i(j-1)}\} \tag{7.11}$$

Following the notation used in section 6.1, we use x^+ to denote x having a particular $x_{ij} = +1$. Similarly, we use x^- to denote x having $x_{ij} = -1$. The probability that the variable in position ij is equal to 1 given N_{ij}, $p(x_{ij} = 1|N_{ij})$, can then be written as

$$p(x_{ij} = 1|N_{ij}) = \frac{p(x^+)}{p(x^+) + p(x^-)} \tag{7.12}$$

Substituting $p(x)$ from (4.1) and cancelling the Z, we get

$$p(x_{ij} = 1|N_{ij}) = \frac{e^{-U(x^+)/T}}{e^{-U(x^+)/T} + e^{-U(x^-)/T}} \tag{7.13}$$

or,

$$p(x_{ij} = 1|N_{ij}) = \frac{1}{1 + e^{(U(x^+)-U(x^-))/T}} \tag{7.14}$$

As $U(x^+)$ and $U(x^-)$ agree in all terms other than those containing x_{ij}, the common terms

in both $U(x^+)$ and $U(x^-)$ drop out and we get the following expression for the conditional probability for $x_{ij} = 1$.

$$p(x_{ij} = 1|N_{ij}) = \frac{1}{1 + e^{2W_{ij}/T}} \tag{7.15}$$

where, W_{ij} for (7.6) is

$$W_{ij} = \beta_{ij,(i+1)j} x_{(i+1)j} + \beta_{ij,i(j+1)} x_{i(j+1)} +$$

$$\beta_{(i-1)j,ij} x_{(i-1)j} + \beta_{i(j-1),ij} x_{i(j-1)} \tag{7.16}$$

Notice that, for the univariate MFM (4.11), the conditional probability, $p(x_{ij} = 1|N_{ij})$, is generailised to marginal probability, $p(x_{ij} = 1)$, where W_{ij} would be equal to α_{ij}. The Equation (6.5) is therefore the univariate variant of (7.15).

Also note that, in (7.15), as $T \to 0$, the value of $p(x_{ij} = 1|N_{ij})$ tends to a limit depending on the W_{ij}. If $W_{ij} > 0$, then $p(x_{ij} = 1|N_{ij}) \to 0$ as $T \to 0$. Conversely, if $W_{ij} < 0$, then $p(x_{ij} = 1|N_{ij}) \to 1$ as $T \to 0$. If $W_{ij} = 0$, then $p(x_{ij} = 1|N_{ij}) = 0.5$ regardless of the value of T. Therefore, the W_{ij} are indicators of whether the x_{ij} at the position ij should be 1 or -1. This indication becomes stronger as the temperature is cooled towards zero.

Now let us describe a variant of GS, which we call the *Bitwise Gibbs Sampler* (BGS). Pseudo code for BGS is shown in Figure 7.4. It starts by randomly generating a solution, then calculates $p(x_{ij}|N_{ij})$ for a chosen x_{ij} and replaces it by sampling $p(x_{ij}|N_{ij})$. This continues until a termination criterion is satisfied. The temperature coefficient, T, in GS can be used to control the convergence of $p(x_{ij}|N_{ij})$. Here, we starts with high temperature, T, then at each iteration, gradually decrease it using a cooling schedule so as to gradually converge $p(x_{ij}|N_{ij})$ to its limit. The DEUM$_d$ algorithm described in previous chapter also uses temperature to control the convergence of the marginals.

Bitwise Gibbs Sampler (BGS)

1. Generate a solution $x^o = \{x_1^o, x_2^o, .., x_n^o\}$ at random.
2. set $r = 0$ and also set the initial value for T.
3. Repeat:
   (a) Set $x^{tmp} = x^o$.
   (b) For $i = 1$ to n
      i. Increase r by 1
      ii. Decrease T
      iii. Set $x_i^o = 1$ with probability $p(x_i^o = 1|N_i)$

   :Until $x^{tmp} = x^o$.
4. Terminate with answer x^o.

Figure 7.4: The pseudo-code of the Bitwise Gibbs Sampler

7.5.2 DEUM with Gibbs sampler

Now that we know how to sample the MRF using a Gibbs Sampler, we can incorporate it in DEUM. Figure 7.5 shows the workflow of Is-DEUM with a Gibbs Sampler (Is-DEUM$_g$).

Notice that, as with Is-DEUM$_m$ (shown in Figure 7.3), Is-DEUM$_g$ only has a single generation. We found that, for the Ising spin glass problem, by repeatedly sampling the x^o with different random starts, Is-DEUM$_g$ was consistently able to find the optimum solution in the first generation. This, therefore, eliminated the need for having multiple generations. However, once we know how to sample from MRF, any standard parent replacement strategies can be used to incorporate a multiple generation scheme in Is-DEUM$_g$ (if required) (Shakya *et al.*, 2005c).

Is-DEUM with Gibbs Sampler (Is-DEUM$_g$)

1. Generate a population, P, of size M
2. Select the set D consisting of N fittest solutions from P, where $N \leq M$.
3. Calculate the MRF parameters θ by fitting MFM to D.
4. Repeat:
   Generate $x^o = \{x_1^o, x_2^o, ..., x_n^o\}$ using BGS
   :Until R tieration completes Or $f(x^o)$ is optimal/good enough
5. Terminate with answer x^o.

Figure 7.5: The pseudo-code of the DEUM with Gibbs Sampler

7.5.3 Experiments and Results

Experiments were conducted with three different sizes of Ising Spin Glass problem: 10×10 ($n = 100$), 16×16 ($n = 256$) and 20×20 ($n = 400$). Four random instances of each problem size were used for the experiment. Each instance was generated by randomly sampling the coupling constant $J_{ij} \in \{+1, -1\}$. The optimum solution for each instance was verified by using Spin Glass Ground server, provided by the group of Prof. Michael Juenger [5]. The parameters for each algorithm were chosen empirically.

We divide our experiments into three parts:

1. A performance comparison with other EDAs
2. A performance comparison with Repeated Bitwise Gibbs Sampler (RBGS)
3. An effect of population size and selection size on the performance of Is-DEUM$_g$

[5] http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/sgs.html

1. Experiment on the performance comparison with other EDA

We made 30 independent runs of Is-DEUM$_g$ for each of the 12 instances of the Ising spin glass problem and recorded the number of fitness evaluations needed to find the optimum. The minimal MFM (7.6) was used to estimate the energy of the Gibbs distribution. The population size and selection size for Is-DEUM$_g$ were 1000 and 250 respectively for n=100, 3000 and 700 respectively for n=256 and 8000 and 1000 respectively for n=400. The temperature T for the BGS was set to $T = 1/0.0005r$, where r is the current number of x_i^o samplings done in BGS (see Figure 7.4). As r increases, T decreases and the solution x^o will converge to a particular value for each x_i^o. The maximum number of allowed repetitions, R, for BGS was set to 500. Is-DEUM$_g$ was terminated if the optimum was found or R repetitions of BGS were done. As, at the end of each BGS, a fitness evaluation was done in order to calculate $f(x^o)$, the number of fitness evaluations was calculated as the sum of population size and the total repetitions of the BGS needed before finding the optimum.

Table 7.3 shows the experimental results on the performance of Is-DEUM$_g$ on all 12 instances of the Ising spin glass problems.

Table 7.3: Performance of Is-DEUM$_g$ on all 12 instances of Ising problem

PI	FE	SD	FE-PS	IT	SR
I-100-1	1008.90	7.96	8.90	639×10^3	100
I-100-2	1002.73	1.82	2.73	149×10^3	100
I-100-3	1010.97	10.71	10.97	723×10^3	100
I-100-4	1003.20	2.32	3.20	156×10^3	100
I-256-1	3015.27	17.45	15.27	115×10^5	100
I-256-2	3003.23	2.96	3.23	213×10^4	100
I-256-3	3054.28	66.17	54.28	351×10^5	97
I-256-4	3007.50	6.05	7.50	639×10^4	100
I-400-1	8093.62	113.80	93.62	152×10^6	97
I-400-2	8036.47	34.87	36.47	632×10^5	100
I-400-3	8058.77	54.14	58.77	103×10^6	100
I-400-4	8047.07	42.25	47.07	755×10^5	100

The first column shows the problem instances (PI). The second and third column shows the average number of fitness evaluation (FE) and the corresponding standard deviation (SD) over the 100 runs. The fourth column shows the average number of fitness evaluation

without counting the number of evaluation needed to evaluate the population (FE-PS). The fifth column shows the average *internal calculation* (IT) of Is-DEUM$_g$ before finding the solution. Internal calculation is the total number of conditional probability calculations done over all repetitions of BGS. The sixth column shows the success rate (SR) of finding the optimum over 100 runs.

Previously, BOA and hBOA have been applied to Ising problem instances of size $n = 100$, $n = 256$ and $n = 400$ (Pelikan & Goldberg, 2003; Pelikan, 2002; Pelikan *et al.*, 2004). Their performances were reported to be comparable and sometimes better than state of the art algorithms for solving Ising spin glass problems.

Our results show that the number of fitness evaluations needed to find the solution for Is-DEUM$_g$ was significantly less than that reported for BOA and hBOA. For example for $n = 400$, the average fitness evaluation for hBOA (with a hill climber) was about 10^5 (BOA was not able to find the solution for $n = 400$). Whereas, for Is-DEUM$_g$ this was only about 8000.

However, for larger instances of Ising spin glass problem, such as $n = 400$, the computational time for sampling x^o using BGS gets higher and therefore the repeated sampling of x^o dominates the computational time taken by the rest of the process in Is-DEUM$_g$. It even dominates the time taken to solve the system of equations. Therefore, the performance of Is-DEUM$_g$ for Ising spin glass problem should be evaluated in terms of the number of internal calculations, (IT), done by the BGS, rather than by the number of fitness evaluations, (FE). As BOA and hBOA do not have such an internal calculation process, it is difficult to compare them with Is-DEUM$_g$.

2. Experiment on the performance comparison with Repeated Bitwise Gibbs Sampler (RBGS)

In Is-DEUM$_g$, we have used GS to sample the MRF which we estimate from the population of solutions. Here we show how we can directly apply GS to the fitness function and sample

x^o.

Given the fitness function (7.5), the conditional probability $p(x_{ij} = 1|N_{ij})$ for any $x_{ij} = 1$ can also be estimated directly as

$$p(x_{ij} = 1|N_{ij}) = \frac{1}{1 + e^{2\Gamma_{ij}}/T} \tag{7.17}$$

Where,

$$\Gamma_{ij} = J_{(i-1)j,ij}x_{(i-1)j} + J_{ij,(i+1)j}x_{(i+1)j} + J_{i(j-1),ij}x_{i(j-1)} + J_{i(j+1),ij}x_{i(j+1)} \tag{7.18}$$

Repeated Bitwise Gibbs Sampler algorithm (RBGS)

1. Repeat:
   Sample a solution $x^o = \{x_1^o, x_2^o, ..., x_n^o\}$ using BGS with $p(x_i = 1|N_i)$ from (7.17) :Until R iteration completes Or $f(x^o)$ is optimal/good enough
2. Terminate with answer x^o.

Figure 7.6: The pseudo-code of the Repeated Bitwise Gibbs Sampler algorithm

We can use (7.17) in BGS (Figure 7.4) as formulation of $p(x_{ij}|N_{ij})$ and sample x^o. As the performance of BGS heavily depends on the initial solution, we repeatedly ran the BGS with different random starts. Figure 7.6 presents the workflow of the *repeated BGS algorithm* (RBGS).

Table 7.4: Performance of Is-DEUM and RBGS for Ising spin glass problem of size $n = 100$, $n = 256$, and $n = 400$. Each column is the average of all four instances of that particular problem size

		Is-DEUM$_g$			RBGS	
PI	FE-PS	IT	SR	FE	IT	SR
I-100	6.45	417×10^3	100	6.53	330×10^3	100
I-256	20.07	136×10^5	99	37.19	284×10^5	99
I-400	58.98	984×10^5	99	94.67	114×10^6	87

The temperature T for the BGS was set to $T = 1/0.005r$, where r is the current number of x_i^o samplings done in BGS (see Figure 7.4). Note that, the constant associated with r

here is 0.005 which is greater than 0.0005 used in Is-DEUM$_g$. This was simply because, with 0.005 the performance of RBGS was better than that with 0.0005. The maximum number of allowed repetitions R was set to 500.

As at the end of each repetition of BGS, a fitness evaluation (FE) was done to check the quality of sampled x^o, we record the total number of BGS repetitions as the number of fitness evaluations for RBGS. However, in contrast to RBGS, Is-DEUM$_g$ had to evaluate a population of solutions, which, although taking a negligible amount of time in comparison to the time taken for a sampling using BGS, hugely contributed to the number of fitness evaluations for Is-DEUM$_g$. Therefore, for Is-DEUM$_g$, we use the fitness evaluation without counting the evaluation of the population (FE-PS) to compare with FE of RBGS. On the other hand, the internal calculation of BGS (IT), is the most dominant factor for the computational cost in both algorithms and therefore we also record the IT for RBGS and compare it with that of Is-DEUM$_g$.

Table 7.4, shows the fitness evaluations (FE) (FE-PS for Is-DEUM$_g$), internal calculation (IT) and success rate (SR) for three different sizes of Ising spin glass problem. Each column is the average of all four instances for their respective problem size. The result for Is-DEUM$_g$ is the average taken from Table 7.3.

Our results show that in terms of both fitness evaluations and internal calculation taken to find the optimum, the performance of Is-DEUM$_g$ was better than that of RBGS. Also note that the success rate for Is-DEUM$_g$ was 99% in comparison to 87% for RBGS for $n = 400$. These results show that sampling from the MRF estimated from the population instead of the actual fitness function results in better performance of the algorithm.

Let us explain the above results. The fitness landscape of the Ising spin glass problem contains large number of plateaus. This is because the values of coupling constants, J, relating two spin variables are restricted to -1 and +1. As RBGS uses J to estimate $p(x_{ij})$, it takes time to overcome all the plateau. However, Is-DEUM$_g$ uses real-valued MRF parameters β to estimate $p(x_{ij})$. β therefore alters the fitness landscape by introducing some variation to the plateau. The result is more efficient searching of the fitness landscape

by the sampling algorithm.

3. Experiment on the effect of population size and selection size on the performance of Is-DEUM$_g$

Performance of EDAs (Including DEUM algorithms) highly depends on two key parameters, 1) the population size (PS) and 2) the selection size (SS). Most of the time, giving a larger PS provides more information about the search space and therefore results in a better performance. The SS however depends on the type of the problem addressed. For deceptive problems, such as trap function (5.4), giving a larger selected set of solution may result in the better estimation of distribution as provides more informance about the fitness landscape. Whereas, for linear problems, such as Onemax (5.2), providing a small set may be enough and result in better effeiciency of the algorithm. However, use of a larger PS and SS comes with a cost, which is a rapid increase in the computational time taken by the algorithm. Larger PS results in increased number of fitness evaluations (FE) and larger SS results in increased time to estimate a probabilistic model.

Having a *single generation*, the negative effect of having a larger PS or SS is less noticable in Is-DEUM$_g$ than in other EDAs, though the positive effect of better estimation of distribution is still noticable. The experiments presented in this section show how the change in PS and SS affects the performance of Is-DEUM$_g$ for Ising spin glass problem.

Table 7.5: The effect of change in population size and selection size on the performance of the Is-DEUM$_g$

PS,SS	8000,1000	4000,1000	2000,1000	1000,1000	2000,2000	4000,4000	8000,8000
FE	8058.98	4098.40	2113.54	1195.16	2123.35	4117.50	8089.66
FE-PS	58.98	98.40	113.54	195.16	123.35	117.50	89.66
IT	985×10^5	119×10^6	868×10^5	393×10^5	613×10^5	907×10^5	108×10^6
tr	1	1.17	0.94	0.51	0.80	1.42	2.07
tr_IT	1	1.20	0.88	0.43	0.60	0.97	1.07
tr_θ	1	1.18	1.18	0.81	1.56	3.17	5.95
SR	99	99	93	18	84	90	97

Table 7.5 presents the experimental results on the performance of 7 different instances Is-DEUM$_g$ with different combination of PS and SS for $n = 400$ bit Ising spin glass problem. Each column represents the performance for specific setup of PS and SS, (PS,SS).

The central column is for Is-DEUM$_g$ with the smallest (PS,SS) = (1000,1000). The PS gradually increases up to 8000 and leaves the SS constant to 1000 as we move towards the left columns of the table. Whereas, towards the right columns of the table both PS and SS gradually increases to 8000.

Each row of the Table 7.5 shows the effect of change in the PS and SS on different criteria used to evaluate the performance of the algorithm. The first row shows the number of fitness evaluation (FE). The second row shows the number of fitness evaluations without counting the evaluation of the population (FE-PS). The third row shows the internal calculation (IT). The fourth row shows the ratio of time (tr) needed by each instance of Is-DEUM$_g$ to find the solution. tr is calculated as the average time taken by a particular instance of Is-DEUM$_g$ divided by the average time taken by Is-DEUM$_g$(8000,1000). Similarly, fifth and sixth row shows the tr for internal calculation (tr_IT) and tr for calculation of MRF parameters, (tr_θ), respectively. Finally, the seventh column shows the success rate (SR) for each instance of Is-DEUM$_g$. All the figures were the average over 30 runs of the algorithm on all 4 instances of the problem.

We can see that, the worst success rate (SR) of 18% was for the Is-DEUM$_g$(1000,1000). The SR gradually improves as the population size (PS) grows. This can be observed by moving to either left or right columns of the table. This shows that the larger PS improves the performance. We can also notice that, although the SR grows towards the right columns of the table, the computational time needed to calculate the MRF parameters, (tr_θ), also grows, and therefore grows the total computational time needed to find the solution, (tr). This effect is obvious as, with higher SS, the number of equation in (4.24) grows and so do the computational time to solve it. On the other hand, towards the left columns of the table, the SR grows, however leaves the tr_θ almost unchanged. This shows that using a higher population size with lower selection size does not decrease the success rate in comparison to using higher population size and higher selection size, however increases the efficiency of the algorithm.

Due to the low success rate of less than 95%, we do not include Is-DEUM$_g$(2000,1000), Is-DEUM$_g$(1000,1000), Is-DEUM$_g$(2000,2000) and Is-DEUM$_g$(4000,4000) for finding the

best performing algorithm in all criteria. Among the remaining three instances, Is-DEUM$_g$(8000,1000) was the best performing algorithm.

Notice that the time to calculate MRF parameters (tr_θ) increased 5.95 times for Is-DEUM$_g$(8000,8000) in comparison to Is-DEUM$_g$(8000,1000). This obviously increased the time to find the solution, (tr). Also notice that, although the fitness evaluation (FE) for Is-DEUM$_g$(4000,1000) was less than that of Is-DEUM$_g$(8000,1000), the FE-PS was higher. As we said earlier, for larger size of Ising spin glass problem, the FE-PS should be taken as the performance measurer rather than FE as the computation time needed to evaluate a population is negligible in comparison to the time needed to sample a solution using GS. This result is also confirmed by internal iteration (IT), which is less for Is-DEUM$_g$(8000,1000) than that for Is-DEUM$_g$(4000,1000).

7.6 Summary

In this chapter, we have presented a bivariate version of DEUM algorithm and applied it to a well known Ising spin glass problem. We have also presented two sampling algorithms, Metropolis sampling and Gibbs sampling, and incorporated them to DEUM as the sampling methods. Our experimental results show that the performance of DEUM was significantly better than that of other EDAs as well as that of a local Gibbs sampler algorithm.

The MCMC algorithm such as Metroplis sampler and Gibbs sampler are effective ways to sample from a probability distribution. They are particularly useful in DEUM as can exploit the temperature in the Gibbs distribution to achieve a better convergence of the algorithm. These methods may well serve as a basis of sampling technique for future DEUM algorithms.

Chapter 8

Future Work

In this chapter, we outline some of the prospective future work that has been identified from our research study. This can be categorised into four groups:

1. Extension to current DEUM algorithms
2. Research on performance improvement
3. Theoretical works
4. Application

8.1 Extension to current DEUM algorithms

This line of research concerns the exploration of future DEUM algorithms, either by introducing new components to the algorithm, or by improving its existing components.

8.1.1 Incorporate a structure learning Algorithm to DEUM

Let us recall the three major steps of EDAs that are different from the GA:

1. Learn the structure of a probabilistic model
2. Estimate the parameters of a probabilistic model
3. Sample the estimated model

The DEUM algorithms implemented so far only have the last two steps and assume the structure of the model to be fixed. The immediate extension for DEUM algorithms would be to incorporate a structure learning algorithm. This would allow them to be applied to a wider range of multivariate optimisation problems. We believe that, the two structure learning algorithms reviewed in chapter 2: Conditional independence test (Santana, 2005) and Linkage detection algorithm (Heckendorn & Wright, 2004), can be readily adopted for this purpose.

8.1.2 Multi-generation scheme in DEUM

Unlike, the two univariate DEUM, DEUM$_{pv}$ and DEUM$_d$, the bivariate Is-DEUM only had a single generation, as was able to find the solution in first generation. However, once we know how to sample from the distribution, it is always possible to have multi-generation scheme. There may be various ways to do so. We suggest two of the immediate ones.

1. **By direct sampling :** The straight forward way to implement multi-generation scheme in Is-DEUM is to sample M solutions using a Gibbs (or Metropolis) sampler, and create the next population to replace the parent P. However, it may be necessary to explicitly maintain the diversity in the population as Gibbs sampler may generate increasing number of similar solutions. There are various ways to do so. One way is to use different nitching techniques and make sure that the useful information in the parent population is not lost while replacing the parent population with new solutions. Another way is to limit the Gibbs sampler to only iterate for small number of iteration. i.e., to sample solution from some *intermediate* distribution which is, at least for first few generations, neither closer to uniform nor closer to a global optimal. This would also help to maintain the variation in the population.

2. **By maintaining a probability vector:** Another way is to use the probability vector to maintain and sample the distribution. The idea is to generate a solution $x = \{x_1, x_2, .., x_n\}$ using Gibbs sampler and update each element of probability vector $p = \{p_1, p_2, .., p_n\}$ towards the direction pointed by values in the solution, i.e, if $x_i = 1$ then increase p_i towards some fixed learning rate. Similarly if $x_i = 0$ then decrease p_i. There may be various ways to obtain x that update the probability vector. One example is to run the Gibbs sampler on the best solution in the current population. This makes sure that at least as best solution is generated. Another way is to generate a population of solutions and choose the one with the best fitness.

8.1.3 Region based decomposition of Energy in Gibbs Distribution

DEUM builds a model of fitness function (MFM) that approximates the energy of the Gibbs distribution. In chapter 4, we introduced two different MFM, for a single MRF structure G:

1. **Minimal MFM** considering only the maximal cliques in G
2. **Complete MFM** considering all maximal cliques, their sub cliques including singleton cliques in G

However, various other *intermediate* models can be constructed for a single structure depending on considered cliques and sub-cliques. Deciding between all possible models could become a time consuming process, particularly when a structure contains more of the higher order cliques. This, so far, did not pose any problem to the proposed DEUM algorithms, as they assumed either a univariate structure or a fixed bivariate structure. The immediate solution to this problem is to restrict the DEUM algorithms to use either minimal MFM or complete MFM.

Alternatively, the *region based decomposition* approach can be used to decide on which cliques to consider in MFM. The Kikuchi approximation approach to estimate and sample

the MRF reviewed in chapter 3 are based on the region based decomposition of cliques. The general idea behind region based decomposition is to find the set of maximal cliques and sub cliques that *validly* factorises the jpd [1]. Once we find the set of cliques, a potential function can be assigned to each of them to construct an MFM. MFM is then fitted to a population in order to approximate the parameters of the MRF.

8.1.4 Selection in DEUM

One of the distinct properties of DEUM algorithm is that, they use fitness to model the distribution and threfore can perform optimisation even without using an explicit selection operator [2]. In other words, a complete parent population, $D = P$, can be used to estimate the probability distribution, and then sampled to evolve the better solutions. However, it is yet to be clarified when selection should be applied and when it should not be applied. For all the experiments presented in this thesis, we empirically determined the size of D. Further research should be carried out in this area. This may lead to an important result for EDA, i.e. the elimination of selection size parameter from the algorithm.

8.1.5 Metropolis sampler with temperature

The zero Temperature Metropolis Method (ZTM) used in Is-DEUM is one of the simplest variant of the metropolis sampling algorithm. ZTM mutates a variable in the solution x to get a mutated solution x'. Then, accepts the mutated solution if energy for x' is less than that of x. In other words, zero temperature metropolis sampler sets $x \rightarrow x'$ if $\Delta U = U(x') - U(x) \leq 0$.

It would be interesting to see the performance of Is-DEUM with a Metropolis method that maintains a temperature coefficient. The mutated solution could then be accepted with a

[1] More detail is given in chapter 3

[2] This property has been described in chapter 5, section 5.3.1

probability $p(x')$, where

$$p(x') = \begin{cases} 1 & \text{if} \quad \Delta U \leq 0 \\ e^{-\Delta U/T} & \text{if} \quad \Delta U > 0 \end{cases}$$

Here, temperature T, as in Gibbs sampler, could be varied in order to control the convergence of the algorithm.

8.1.6 Research on different ways to numerically define the clique potential functions

So far, we have defined the clique potential functions as the product of variables in the clique and a coefficient associated with them (for examples, see (4.10) and (4.14)). However, it is possible to define the potential functions in any desired way. For example, it may be beneficial to define the potential functions, such that they models some aspect of the fitness function, which may result in better approximation of distribution. Further research can be done in this area.

8.1.7 Clique based mutation to minimise Energy in Gibbs distribution

The Zero Temperature Metropolis method (Figure 7.2) presented in chapter 7, and used by Is-DEUM, only had a single bit mutation scheme, i.e., in each iteration it chooses the configuration of a bit that minimises the energy $U(x)$. However, we could use a *multi bit* mutation scheme for metropolis sampling. The idea is to check the effect of all 2^k configuration of clique (where k is the clique size) in the $U(x)$ and choose one that minimises it. Although the computational cost of this task is exponential to the size of the maximum clique in structure G, it is still an attainable goal.

In context of EDAs, a similar approach, known as *Building Blocks-wise mutation*, has been previously introduced by (Sastry *et al.*, 2004; Sastry & Goldberg, 2004). They use the structure of the model to identify the configuration of each clique that maximises the

fitness function $f(x)$. The clique based metropolis sampling to minimise energy, however, will be computationally less expensive than their approach as the energy, $U(x)$, here is the linear approximation of the fitness, i.e. $-ln(f(x)) = U(x) = \sum \psi_i(c_i)$, and therefore does not require the costly fitness evaluation. Rather, it will only require the calculation of the difference in two energies, which involves calculation of the potential functions containing only the mutated variables. This has been demonstrated in equations (7.9) and (7.10).

8.2 Research on performance improvement

This line of research concerns the implementation of various performance improvement techniques that have been either previously exploited in different EA, or are specific to the DEUM algorithm.

8.2.1 Research in more efficient way to estimate MRF parameters

The Singular Value Decomposition (SVD) technique used in DEUM algorithms to solve the system of linear equations is a computationally expensive process. Here, the time grows polynomially to the number of MRF parameters in the model. Therefore, further research should be done in order to find the efficient way to estimate the MRF parameters.

One solution to this may be the use of some non numeric techniques such as *heuristic based optimisations* to solve the system of equations.

Another solution may be to decompose the matrix A in system of linear equations, (4.24) into smaller matrixes and solve them individually. It can be possible for the case, where the problem can be decomposed into a number of sub problems. Estimating a separate model for each of the sub problem would significantly reduce the number of parameters to compute at once, and therefore would significantly reduce the computational time to solve the system of equations.

8.2.2 Different cooling schedules

Due to the use of Gibbs distribution approach to factorise jpd, temperature plays an important role in DEUM algorithms. We use temperature to balance between exploration and exploitation of the search space in two DEUM algorithms: DEUM$_d$ and Is-DEUM$_g$. Both of these algorithms use a *cooling scheme* for this propose. However, the cooling scheme used so far is of proportionate nature. i.e., in each iteration, the temperature is decreased with a fixed ratio specified by the user, which is proportional to the number of current iteration (see DEUM$_d$ workflow in Figure 6.1). However, there may be various other possible cooling schemes, such as linear scheme, quadratic scheme, or exponential scheme, that may be similarly applied in these algorithms. Further research can be done to find the effect of different cooling schedule in the performance of these algorithms.

8.2.3 Performance enhancement using different GA techniques

There are several well established performance enhancement techniques in GA that can be readily adopted in DEUM algorithms. They include:

1. **Replacement strategy**: There are various replacement strategy in GA that is used to improve the efficiency of the algorithm. They include various elitism strategies (Davis, 1991), restricted tournament replacements (Pelikan & Goldberg, 2000), and so on. They can be easily adopted in DEUM algorithms.

2. **Hybridisation**: Hybridisation of GAs with other search techniques have been found to be very effective in improving the optimisation quality. Specifically, the local search algorithms, such as various hill climbers, greedy algorithms, tabu search has been widely incorporated in GA. Similar hybridisation techniques can be implemented in DEUM.

3. **Parallelisation**: Another highly explored area in EA research is their parallelisation. Because EA are a population based optimisation techniques, parallelisation can be naturally implemented and are shown to work extremely well. Parallel version of several EDAs

has also been proposed (Lozano *et al.*, 2001a; Mendiburu *et al.*, 2005; Lobo *et al.*, 2005; Ocenasek & Schwarz, 2000; Ocenasek, 2002; Ocenasek *et al.*, 2003). DEUM could also benefit from such approach.

4. **Prior knowledge utilisation**: One of the advantages of EDA is that the prior knowledge about the problem decomposition can be easily incorporated in their model of distribution. This may result in better estimation of distribution and thus in better performance of the algorithm. We have shown this in DEUM by applying it to the Ising spin glass problem, where the information about the variable interaction was copied to the model of distribution. A similar approach may be applied for the problems where the variable dependency is known in advance.

8.3 Theoretical works

One of the motivation behind the emergence of EDA was to achieve better and more rigorous theoretical analysis of the evolutionary process (Larrañaga & Lozano, 2002). The work on DEUM so far is based on empirical analysis of its performance in various situations. The theoretical analysis of DEUM would be an interesting area to follow. Amongst other, they could include analysis of DEUM convergence in different situations, analysis of population size requirements and their scalability.

8.4 Application

Although recent years have seen growing interest in EDA research, the application of EDAs in real world optimisation problems is still an emerging area of research. Our experimental results show that, for the wide range of tested problems, DEUM can significantly reduce the number of fitness evaluations needed to find the solution and often results in better quality of the found solution. It gives a strong motivation to apply DEUM to real world problems where the less fitness evaluation and improved solution quality is likely to be the

most important requirements.

Chapter 9

Conclusion

This chapter highlights some of the important contributions made by our research and concludes the thesis.

9.1 Important contributions

The following contributions are identified as important ones among the numerous outcomes from our research.

1. **Review of Probabilistic Graphical Models in EDA**: A review of Probabilistic Graphical Models in context of EDA has been presented. Specifically, various techniques to estimate a MRF in EDAs have been described.

2. **MRF Fitness Model**: A general model of fitness function, called MRF Fitness Model (MFM), has been established, which relates the energy of the Gibbs distribution with the fitness of the solution and completely specifies the probability distribution.

3. **MRF parameter estimation**: Least square fitting technique has been proposed to estimate the parameter of the MRF.

4. **DEUM**: A general framework of EDA, called Distribution Estimation using MRF (DEUM), has been introduced using the proposed approach to estimate the MRF.

5. **DEUM$_{pv}$**: A univariate version of DEUM, called DEUM$_{pv}$, has been presented using a probability vector approach to sampling the MRF.

6. **DEUM$_{d}$**: An extension to DEUM$_{pv}$, called DEUM$_{d}$, has been introduced that replaced probability vector approach to sampling in DEUM$_{pv}$ with a direct Gibbs distribution sampling approach.

7. **Is-DEUM**: A bivariate variant of DEUM, called Is-DEUM, has been proposed and applied to the well known Ising spin glass problems. Two versions of Is-DEUM have been proposed using two different sampling techniques: 1) Is-DEUM$_{m}$ with a Metropolis sampling technique and 2) Is-DEUM$_{g}$ with a Gibbs sampling technique.

8. **Experimental Analysis**: Detail experimental analysis has been performed to test the performance of DEUM algorithms in a wide range of optimisation problems. Comparison has been made with other EDAs and validated using different significance testing techniques.

9.2 General conclusion

In this thesis, we have presented DEUM as a general framework for an EDA based on Markov Random Field approach to estimate and sample the distribution. The key to the DEUM success lies on the effective exploitation of two main property of the Gibbs distribution, which also distinguishes them from other EDAs. They follow:

1. **Modelling fitness to estimate MRF parameters:** The behaviour of the system using Gibbs distribution completely depends on the energy function, which further depends on the chosen clique potential functions defined on the structure of the MRF. The described fitness modelling approach to estimate the MRF exploits this property by building a model of fitness function, MFM, over the chosen clique potential functions. The model is then fitted to the population of solutions to estimate the

MRF parameters. The result is that the fitness is now incorporated in the variation part of the evolution, which allows DEUM to perform optimisation even without using a traditional selection operator. This is a distinct characteristic of DEUM, which has a significant contribution to the performance of these algorithms.

2. **Exploitation of Temperature:** The temperature coefficient in Gibbs distribution can be manipulated to observe the desired system behaviour. Particularly, with high temperature, the distribution is closer to be uniform and with lower temperature, it concentrates around some global optima. This is exploited in several DEUM algorithms. Particularly, the DEUM$_d$ starts with a high temperature, i.e., start with a near uniform distribution, and, as iteration continues, gradually cools the temperature down to minimum i.e. gradually converges the distribution to some optima. The result is the effective exploration of the search space and better performance of the algorithm. Is-DEUM$_g$ also takes similar approach by using a cooling schedule in the Gibbs sampling algorithm.

A number of experiments has been preformed to test the performance of DEUM algorithms on range of optimisation problems. Results show that, for most of the tested problems, the DEUM algorithms significantly outperformed other EDAs, both in terms of number of fitness evaluations and in terms of the quality of the found solutions. Although, the use of least square fitting technique to estimate the parameter may make these algorithms computationally more expensive than other EDAs using frequency counting approach, the better quality of solution and the less fitness evaluation may account to be more important for many real world problems, and therefore DEUM algorithms can have significant impact in such problems.

To conclude the thesis, we state that the proposed fitness modelling approach to estimating and sampling the MRF in DEUM has a great potential for solving optimisation problems. The thesis has introduced several algorithms based on this general framework and successfully applied in many optimisation problems. The results obtained are promising and can have important implications in general search and optimisation research. It also paves the way for wider exploration of other promising EDAs based on this framework.

Bibliography

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,. Tech. Rep. CMU-CS-94-163, Pittsburgh, PA.

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

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

Baluja, S. & Davies, S. (1998). Fast probabilistic modeling for combinatorial optimization. In *AAAI-98*.

Bengoetxea, E., Larrañaga, P., Bloch, I., Perchant, A. & Boeres, C. (2000). Inexact graph matching using learning and simulation of Bayesian networks. An empirical comparison between different approaches with synthetic data. In *Workshop Notes of CaNew2000: Workshop on Bayesian and Causal Networks: From Inference to Data Mining*, fourteenth European Conference on Artificial Intelligence, ECAI2000. Berlin.

Bengoetxea, E., Larrañaga, P., Bloch, I. & Perchant, A. (2001a). Image recognition with graph matching using estimation of distribution algorithms. In *Proceedings of the Medical Image Understanding and Analysis 2001*.

Bengoetxea, E., Larrañaga, P., Bloch, I. & Perchant, A. (2001b). Solving graph matching with EDAs using a permutation–based representation. In P. Larrañaga & J.A. Lozano,

eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

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

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems (with discussions). *Journal of the Royal Statistical Society*, **36**, 192–236.

Bethe, H.A. (1935). Statistical theory of superlattices. *Proc. Roy. Soc. London A*, 150–552.

Born, C. & Kerbosch, J. (1973). Algorithms 457 - finding all cliques of an undirected graph. *Communications of the ACM*, **16(6)**, 575–577.

Bosman, P.A. (2003). *Design and Application of Iterated Density-Estimation Evolutionary Algorithms*. Ph.D. thesis, Universiteit Utrecht, Utrecht, The Netherlands.

Brown, D.F., Garmendia-Doval, A.B. & McCall, J.A.W. (2002). Markov Random Field Modelling of Royal Road Genetic Algorithms. *Lecture Notes in Computer Science*, **2310**, 65–78.

Brush, S.G. (1967). History of the lenz-ising model. *Rev. Mod. Phys.*, **39**.

Buntine, W.L. (1991). Theory refinement of Bayesian networks. In *Uncertainty in Artificial Intelligence*, 52–60.

Chickering, D., Heckerman, D. & Meek, C. (1997). A bayesian approach to learning bayesian networks with local structure. *In Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*, 80–89, also appears as Technical Report MSR-TR-97-07, Microsoft Research, August, 1997.

Chickering, D.M., Geiger, D. & Heckerman, D. (1994). Learning Bayesian Networks is NP-Hard. Tech. Rep. MSR-TR-94-17.

Chow, C. & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, **14**, 462–467.

Cooper, G.F. & Herskovits, E.A. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, **9**, 309–347.

Davis, L., ed. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

de Bonet, J.S., Isbell, C.L., Jr. & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In M.C. Mozer, M.I. Jordan & T. Petsche, eds., *Advances in Neural Information Processing Systems*, vol. 9, The MIT Press.

de Campos, L.M., Gámez, J.A., Larrañaga, P., Moral, S. & Romero, T. (2001). Partial abductive inference in Bayesian networks: an empirical comparison between GAs and EDAs. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

de la Maza, M. & Tidor, B. (1993). An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection. In S. Forrest, ed., *Proc. of the Fifth International Conference on Genetic Algorithm*, 124–131, Morgan Kaufmann, Urbana-Champaign, IL.

de la Ossa, L., Gámez, J.A. & Puerta, J.M. (2004). Migration of Probability Models Instead of Individuals: An Alternative When Applying the Island Model to EDAs. In *Parallel Problem Solving from Nature VIII*, 242–252, Springer.

Droste, S. (2005). Not all linear functions are equally difficult for the compact genetic algorithm. In *In proceedings of Genetic and Evolutionary Computation COnference (GECCO 2005)*, 679–686.

Etxeberria, R. & Larrañaga, P. (1999). optimization with bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99.*, 332339, Cuba.

Fogel, L.J. (1962). Autonomous automata. *Industrial Research*, **4**, 14–19.

Galić, E. & Höhfeld, M. (1996). Improving the generalization performance of multi-layer-perceptrons with population-based incremental learning. In *Parallel Problem Solving from Nature. PPSN-IV*, 740–750.

Geman, S. & Geman, D. (1987). Stochastic relaxation, gibbs distributions and the bayesian restoration of images. In M.A. Fischler & O. Firschein, eds., *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, 564–584, Kaufmann, Los Altos, CA.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Goldberg, D.E., Korb, B. & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems*, **3**, 493–530.

Golub, G. & Van Loan, C. (1989). *Matrix Computations*. Baltimore, MD, 2nd edn.

González, C., Lozano, J. & Larrañaga, P. (2001). Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, **12**.

González, C., Rodríguez, J.D., Lozano, J. & Larrañaga, P. (2003). Analysis of the Univariate Marginal Distribution Algorithm modeled by Markov chains. *Lectur Notes in Computer Science*, **2686**, 510–517.

Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, **16**, 122–128.

Hammersley, J.M. & Clifford, P. (1971). Markov fields on finite graphs and lattices. *Unpublished*.

Harik, Cantu-Paz, Goldberg & Miller (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In *IEEECEP: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*.

Harik, G. (1994). Finding multiple solutions in problems of bounded difficulty. Tech. Rep. IlliGAL Report No. 94002, University of Illinois at Urbana-Champaign, Urbana, IL.

Harik, G. (1997). *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Ph.D. thesis.

Harik, G. (1999). Linkage learning via probabilistic modeling in the ECGA. Tech. Rep. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign.

Harik, G.R., Lobo, F.G. & Goldberg, D.E. (1999). The compact genetic algorithm. *IEEE-EC*, **3**, 287.

Heckendorn, R.E. & Wright, A.H. (2004). Efficient linkage discovery by limited probing. *Evolutonary Computation*, **12**, 517–545.

Heckerman, D., Geiger, D. & Chickering, D.M. (1994). Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, 85–96.

Henrion, M. (1988). Propagating uncertainty in bayesian networks by probabilistic logic sampling. In J.F. Lemmer & L.N. Kanal, eds., *Uncertainty in Artificial Intelligence 2*, 149–163, North-Holland, Amsterdam.

Höhfeld, M. & Rudolph, G. (1997). Towards a theory of population-based incremental learning. In *Proceedings of the 4th International Conference on Evolutionary Computation*, 1–5, IEEE Press.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Hoos, H.H. & Stutzle, T. (1999). Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence*, **112**, 213–232.

Inza, I., Larrañaga, P. & R. Etxeberria, B.S. (2000). Feature subset selection by Bayesian networks based optimization. *Artificial Intelligence*, **123**, 157–184.

Inza, I., Larrañaga, P. & Sierra, B. (2001a). Estimation of Distribution Algorithms for feature subset selection in large dimensionality domains. In H. Abbass, R. Sarker & C. Newton, eds., *Data Mining: A Heuristic Approach*, Idea Groups Publishing.

Inza, I., Larrañaga, P. & Sierra, B. (2001b). Feature subset selection by Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

Inza, I., Merino, M., Larrañaga, P., Quiroga, J., Sierra, B. & Girala, M. (2001c). Feature subset selection by population-based incremental learning. A case study in the survival of cirrhotic patients with TIPS. *Artificial Intelligence in Medicine*.

Jensen, F.V. (1996). *Introduction to Bayesian Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Jensen, F.V. (2001). *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Jensen, F.V. & Jensen, F. (1994). Optimal junction trees. In *Uncertainty and Artificial Intelligence: Proceedings of the Tenth Conference*, Morgan Kaufmann, San Mateo, CA.

Jordan, M.I., ed. (1998). *Learning in Graphical Models*. NATO Science Series, Kluwer Academic Publishers, Dordrecht.

Jordan, M.I. (2004). Graphical models. *Statistical Science (Special Issue on Bayesian Statistics)*, **19**, 140–155.

Jordan, M.I., Ghahramani, Z., Jaakkola, T. & Saul, L.K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, **37**, 183–233.

Kargupta, H. (1996). The gene expression messy genetic algorithm. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, 631–636, IEEE Press.

Kikuchi, R. (1951). A Theory of Cooperative Phenomena. *Physical Review*, **81**, 988–1003.

Kindermann, R. & Snell, J.L. (1980). *Markov Random Fields and Their Applications*. AMS.

Kullback, S. & Leibler, R.A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, **22**, 79–86.

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

Larrañaga, P., Etxeberria, R., Lozano, J. & Peña, J. (1999). Optimization by learning and simulation of bayesian and gaussian networks. Tech. Rep. EHU-KZAA-IK-4/99, University of the Basque Country.

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

Lauritzen, S.L. (1996). *Graphical Models*. Oxford University Press.

Lauritzen, S.L. & Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, **50**, 157–224.

Li, S.Z. (1995). *Markov Random Field modeling in computer vision*. Springer-Verlag.

Lobo, F.G., Lima, C.F. & Mrtires, H. (2005). Massive parallelization of the compact genetic algorithm. In B. Ribeiro, R.F. Albrechet, A. Dobnikar, D.W. Pearson & N.C. Steele, eds., *In proceedings of the International Conference on Adaptive and Natural computiNG Algorithms (ICANNGA 2005)*, Springer-Verlag, Wien, Coimbra, Portugal.

Lozano, J.A., Sagarna, R. & Larrañaga, P. (2001a). Parallel Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, 129–145, Kluwer Academis Publishers.

Lozano, J.A., Sagarna, R. & Larrañaga, P. (2001b). Solving job scheduling with Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, 231–242, Kluwer Academis Publishers.

Marascuilo, L.A. & McSweeney, M. (1977). *Nonparametric and distribution-free methods for the social sciences*. Wadsworth Publishing, Belmont, CA.

Mendiburu, A., Lozano, J.A. & Miguel-Alonso, J. (2005). Parallel implementation of edas based on probabilistic graphical models. *IEEE Transactions onn Evolutionary Computation*, **9**, 406–423.

Metropolis, N. (1953). Equations of state calculations by fast computational machine. *Journal of Chemical Physics*, **21**, 1087–1091.

Michalewicz, Z. (1996). *Genetic Algorithm +Data Structures = Evolution Programs*. Springer-Verlag, New York.

Mitchell, M. (1997). *An Introduction To Genetic Algorithms*. MIT Press, Cambridge, Massachusetts.

Mitchell, M., Holland, J.H. & Forrest, S. (1994). When will a genetic algorithm outperform hillclimbing? In J.D. Cowan, G. Tesauro & J. Alspector, eds., *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann.

Mühlenbein, H. (1994). The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, **1**, pp. 335–360.

Mühlenbein, H. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation*, **5**, 303–346.

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

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

Mühlenbein, H. & Mahnig, T. (2001a). Mathematical analysis of evolutionary algorithms for optimization. In A. Rodriguez, M. Ortiz & R. Santana, eds., *Proceedings of the Third Internatinal Symposium on Adaptive Systems*, 166–185, Institute of Cybernetics, Mathematics and Physics (ICIMAF), Cuba.

Mühlenbein, H. & Mahnig, T. (2001b). A new adaptive boltzmann selection schedule sds. In *Proceedings of the 2001 Congress on Evolutionary Computation*.

Mühlenbein, H. & Paaß, G. (1996). From recombination of genes to the estimation of distributions: I. binary parameters. In H.M. Voigt, W. Ebeling, I. Rechenberg & H.P.

Schwefel, eds., *Parallel Problem Solving from Nature – PPSN IV*, 178–187, Springer, Berlin.

Mühlenbein, H., Mahnig, T. & Ochoa, A.R. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, **5**, 215–247.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.

Ocenasek, J. (2002). *Parallel Estimation of Distribution Algorithms*. Ph.D. thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic.

Ocenasek, J. & Schwarz, J. (2000). The parallel bayesian optimization algorithm. In *Proceedings of the European Symposium on Computational Inteligence, Physica-Verlag*, 61–67, Kosice, Slovak Republic.

Ocenasek, J., Schwarz, J. & Pelikan, M. (2003). Design of multithreaded estimation of distribution algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, 1247–1258.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman Publishers, Palo Alto, CA.

Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy*. Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, also IlliGAL Report No. 2002023.

Pelikan, M. & Goldberg, D.E. (2000). Hierarchical problem solving by the bayesian optimization algorithm. IlliGAL Report No. 2000002, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.

Pelikan, M. & Goldberg, D.E. (2003). Hierarchical BOA solves Ising spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, 1271–1282, also IlliGAL Report No. 2003001.

Pelikan, M. & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In

R. Roy, T. Furuhashi & P.K. Chawdhry, eds., *Advances in Soft Computing - Engineering Design and Manufacturing*, 521–535, Springer-Verlag, London.

Pelikan, M., Goldberg, D.E. & Cant'u-Paz, E. (1999a). BOA: The Bayesian Optimization Algorithm. In W. Banzhaf et al., ed., *Proceedings of the Genetic and Evolutionary Computation Conference GECCO99*, vol. I, 525–532, Morgan Kaufmann Publishers, San Fransisco, CA.

Pelikan, M., Goldberg, D.E. & Lobo, F. (1999b). A survey of optimization by building and using probabilistic models. Tech. Rep. 99018, Illinois Genetic Algorithms Lab, UIUC, Urbana, IL.

Pelikan, M., Ocenasek, J., Trebst, S., Troyer, M. & Alet, F. (2004). Computational complexity and simulation of rare events of ising spin glasses. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 36–47.

Pelikan, M., Sastry, K. & Goldberg, D. (2005a). Multiobjective hboa, clustering, and scalability. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, 663–670, also IlliGAL Report No. 2005005.

Pelikan, M., Sastry, K. & Goldberg, D.E. (2005b). Sporadic model building for efficiency enhancement of hboa. IlliGAL Report No. 2005026, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.

Petrovski, A., Shakya, S. & McCall, J. (2006). Optimising cancer chemotherapy using an estimation of distribution algorithm and genetic algorithms. In *proceedings of Genetic and Evolutionary Computation COnference (GECCO 2006) (in press)*, ACM, seattle, USA.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1993). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2nd edn.

Preston, C. (1976). Random fields. *Lecture Notes in Mathematics*, **534**.

Rechenberg, I. (1973). *Evolutionstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Formman-Holzboog, Stuttgart.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica*, **14**, 465–471.

Robles, V., de Miguel, P. & Larrañaga, P. (2001). Solving the travelling salesman problem with Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

Roure, J., Sangüesa, R. & Larrañaga, P. (2001). Partitional clustering by means of Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

Sagarna, R. & Larrañaga, P. (2001). Solving the knapsack problem with Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academis Publishers.

Sagarna, R. & Lozano, J.A. (2005). On the performance of Estimation of Distribution Algorithms applied to software testing. *Applied Artificial Intelligence*, **19**, 457–489.

Sagarna, R. & Lozano, J.A. (2006). Scatter search in software testing, comparision and collaboration with Estimation of Distribution Algorithms. *Europian Journal of Operational Research*, **169**, 392–412.

Santana, R. (2003a). A markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, vol. 2837, 337–348, Springer-Verlag, Dubrovnik, Croatia.

Santana, R. (2003b). *Probabilistic modeling based on undirected graphs in Estimation Distribution Algorithms*. Ph.D. thesis, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba.

Santana, R. (2005). Estimation of Distribution Algorithms with Kikuchi Approximation. *Evolutonary Computation*, **13**, 67–98.

Santana, R., Larraaga, P. & Lozano, J.A. (2005). Interactions and dependencies in estimation of distribution agorithms. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC-2005)*, 1418–1425, IEEE press, Edinburgh, UK.

Sastry, K. & Goldberg, D.E. (2004). Designing competent mutation operators via probabilistic model building of neighborhoods. Tech. Rep. 2004006, IlliGAL.

Sastry, K., Goldberg, D.E. & Pelikan, M. (2004). Efficiency enhancement of probabilistic model building genetic algorithms. Tech. Rep. 2004020, IlliGAL.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **7**, 461–464.

Shakya, S., McCall, J. & Brown, D. (2005a). Estimating the distribution in an EDA. In B. Ribeiro, R.F. Albrechet, A. Dobnikar, D.W. Pearson & N.C. Steele, eds., *In proceedings of the International Conference on Adaptive and Natural computiNG Algorithms (ICANNGA 2005)*, 202–205, Springer-Verlag, Wien, Coimbra, Portugal.

Shakya, S., McCall, J. & Brown, D. (2005b). Using a Markov Network Model in a Univariate EDA: An Emperical Cost-Benefit Analysis. In *proceedings of Genetic and Evolutionary Computation COnference (GECCO2005)*, 727–734, ACM, Washington, D.C., USA.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2004a). Preliminary results on Evolution without Selection. In *Proceedings of Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science (PREP 2004)*, Hertfordshire, UK.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2004b). Updating the probability vector using MRF technique for a Univariate EDA. In E. Onaindia & S. Staab, eds., *Proceedings of the Second Starting AI Researchers' Symposium, volume 109 of Frontiers in artificial Intelligence and Applications*, 15–25, IOS press, Valencia, Spain.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2005c). Incorporating a metropolis method in a distribution estimation using markov random field algorithm. In *proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2005)*, vol. 3, 2576–2583, IEEE press, Edinburgh, UK.

Shakya, S.K., McCall, J.A.W. & Brown, D.F. (2006). Solving the ising spin glass problem using a bivariate eda based on markov random fields. In *proceedings of IEEE Congress*

on Evolutionary Computation (IEEE CEC 2006), IEEE press, Vancouver, Canada (in press).

Sierra, B., Jiménez, E., Inza, I., Larrañaga, P. & Muruzábal, J. (2001). Rule induction using Estimation of Distribution Algorithms. In P. Larrañaga & J.A. Lozano, eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, Kluwer Academic Publishers.

Smyth, P. (1998). Belief networks, hidden markov models, and markov random fields: a unifying view. *Pattern Recognition Letters*.

Spirtes, P., Glymour, C. & Scheines, R. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, **9**, 62–72.

Spirtes, P., Glymour, C. & Scheines, R., eds. (1993). *Causation, Prediction and Search*. Lecture Notes in Statistics 81, Springer Verlag, New York.

Suda, R. & Kuriyama, S. (2004). Another preprocessing algorithm for generalized one-dimensional fast multipole method. *Journal of Computational Physics*, **195**, 790–803.

Whittaker, J. (1990). *Graphical Models in applied multivariate Statistics*. John Wiley.

Wright, A.H. & Pulavarty, S. (2005). Estimation of distribution algorithm based on linkage discovery and factorization. In *In proceedings of Genetic and Evolutionary Computation COnference (GECCO 2005)*, ACM, Washington, D.C., USA.

Yedidia, J.S., Freeman, W.T. & Weiss, Y. (2001). Bethe free energy, kikuchi approximations and belief propagation algorithms. Tech. Rep. TR2000-26, Mitsubishi Electric Research Laboratories.

Yedidia, J.S., Freeman, W.T. & Weiss, Y. (2002). Understanding belief propagation and its generalizations. Tech. Rep. TR2001-22, Mitsubishi Electric Research Laboratories.

Yedidia, J.S., Freeman, W.T. & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, **51**, 2282–2312.

Zhang, Q. & Muehlenbein, H. (2004). On the convergence of a class of estimation of distribution algorithms. *IEEE Trans. on Evolutionary Computation*, **8**.

Zhang, Q. & Mühlenbein, H. (1999). On global convergence of FDA with proportionate selection. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99*, 340–343, la Habana.