Updating the probability vector using MRF technique for a Univariate EDA

S. K. Shakya, J. A. W. McCall, and D. F. Brown
{*ss, jm, db*}@*comp.rgu.ac.uk*
School of Computing, The Robert Gordon University,
St.Andrew Street, Aberdeen,
AB25 1HG, UK.

Abstract. In this paper, we propose a new technique to update a probability vector [1] for Estimation of Distribution Algorithms (EDA)[15]. We present a novel algorithm belonging to the general class of EDA which we call Distribution Estimation using Markov Random Fields (DEUM). In common with other EDAs, DEUM uses a population of chromosomes to build a probabilistic model of good solutions. The model is then used to update the probability vector which is sampled to generate further populations, replacing the function of recombinative operators in a traditional genetic algorithm. DEUM uses Markov Random Field (MRF) modelling [3] to estimate the probabilistic relationship between allele values and fitness and iteratively refine a probability vector to generate better solutions. We present experimental results on the performance of DEUM and make comparisons with other algorithms in the Univariate EDA class. Our experiments show that MRF modelling provides a significant advantage over other approaches on problems where univariate algorithms are typically used.

1 Introduction

Estimation of Distribution Algorithms (EDA) [15] [11], also known as Probabilistic Model Building Genetic Algorithms [PMBGA] [17], are a developing area of research interest in the field of evolutionary and genetic algorithms (GA). The earliest example of this class of algorithm is Population Based Incremental Learning (PBIL), first proposed in [1]. Several different variants of EDA have been proposed to date [18] and are the subject of active research in the GA community. EDAs are reported to solve problems that are known to be the hard problems for traditional GAs [19]. EDAs are motivated by the idea of identifying and preserving the important patterns or Building Blocks [7]. A model of the probability distribution of allele values is constructed from the population of solutions at each cycle. This model is used to generate the next population of solutions in a way that is explicitly biased towards important patterns learnt by the EDA. This contrasts with the implicit processing of Building Blocks in traditional GAs. EDAs are classified as univariate, bivariate or multivariate [17] according to the extent to which interaction between allele values is incorporated in the estimate of the probability distribution.

The aim of this paper is to introduce a new approach to update a probability vector [1] in order to improve the performance of EDAs. We describe an optimization algorithm which we call *Distribution Estimation using Markov Random Fields* (DEUM). DEUM belongs to the class of univariate EDAs. DEUM uses the Markov Random Field (MRF) modelling approach described in [3] to update the probability vector. In most univariate EDAs, the probability distribution is estimated using the marginal frequency of each allele value in a selected subset of the population (see for example [15], [17]). In particular, PBIL uses marginal frequencies to evolve a probability vector that is sampled to generate successive populations. In DEUM, we adapt the PBIL approach by replacing marginal frequencies with an MRF model on a selected set of solutions. The MRF model gives a maximum likelihood estimation of the optimal solution based on the selected set and this is used to update the probability vector. The result of this, as we hypothesise and show by experiments, is a significant improvement in learning on well-known univariate EDA problems.

The rest of the paper is constructed as follows. Section 2 sets out the relevant background of EDAs. Section 3 introduces our univariate MRF model and shows how we determine the model from a population of solutions. Section 4 describes the workflow of DEUM and we present experiments and results in section 5. We summarise the paper and outline further work in section 6.

2 Background

In EDAs, a solution (chromosome/individual) is regarded as a set of random variables (the alleles) each taking a particular value from a set of possible values. In particular we represent a solution as $x = \{x_1, x_2, .., x_n\}$ where each x_i is the value taken by the i^{th} random variable. In this paper, we consider problems where solutions are encoded as bit-string chromosomes and so n is chromosome length, the x_i represent allele values in the obvious way and so each x_i will be 0 or 1.

As in a traditional GA, all EDAs begin by generating an initial population of M solutions. N promising solutions are then selected according to a chosen selection criterion (usually fittest) for some $N \leq M$. An estimation of the probability distribution of allele values is made from the selected set of solutions [15]. Offspring are then generated by sampling the probability distribution to replace the current population with a new population. This process continues until a termination criterion is satisfied.

Univariate EDAs do not consider any dependencies between variables, i.e., they are capable of modelling only building blocks of order one. So the joint probability distribution becomes simply the product of the univariate marginal probability of all variables in an individual, i.e;

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

where, $p(x_i)$ is the marginal probability of i^{th} variable having value x_i.

Population Based Incremental Learning (PBIL) [1], Univariate Marginal Distribution Algorithm (UMDA) [15] and Compact Genetic Algorithm (cGA) [9] all use the univariate model of probability distribution.

Bivariate EDAs consider pair-wise dependencies between variables, i.e., they are capable of modelling building blocks of order two. The probability model in this case becomes more complex than one of univariate model and takes a form of a probabilistic network between variables [20]. Mutual Information Maximization for Input Clustering (MIMIC) [5], Combining Optimizers with Mutual Information Trees (COMIT) [2], Bivariate Marginal Distribution Algorithm (BMDA) [20] all use the bivariate Model of probability Distribution.

Multivariate EDAs consider dependencies between variables of order more than two. The probability network representing such dependencies obviously becomes more complex, and the computation time to search for a good network increases. Extended Compact Genetic Algorithm (ECGA) [8], Factorised Distribution Algorithm (FDA) [13], Bayesian Optimization algorithm (BOA) [16], Learning Factorised Distribution Algorithm (LFDA) [14], and Estimation of Bayesian Network (EBNA) [6] all use the multivariate model of probability distribution.

This paper concentrates on univariate EDAs.

3 Univariate Markov Random Field Modelling of Chromosome Fitness

MRF theory [12] is a branch of probability theory for analysing the spatial or contextual dependencies of physical phenomena. It has been applied for many years to the analysis of images, particularly in the detection of visual patterns or textures. In [3] MRF theory was used to provide a formulation of the joint probability distribution that relates solution fitness to an energy function calculated from the values of the solution variables. To be precise,

$$p(x) = \frac{f(x)}{\sum_y f(y)} = \frac{e^{-U(x)}}{\sum_y e^{-U(y)}} \tag{2}$$

from which can be derived an equation for each solution x,

$$-\ln(f(x)) = U(x) \tag{3}$$

Here, $f(x)$ is the fitness of an individual and $U(x)$ is an energy function. The summations are over all possible solutions y. Specification of $U(x)$ fully specifies the joint probability distribution. $U(x)$ can be thought of as a probabilistic model of the fitness function. In particular, minimising $U(x)$ is equivalent to maximising $f(x)$.

In general, the form of the energy function will involve interaction between the variables x_i. In DEUM, however, we use a simple form which assumes no interaction. Instead each variable provides a contribution $\alpha_i x_i$ to the overall energy. Using the formulae above, we derive an equation for each solution

$$-\ln(f(x)) = \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n \tag{4}$$

We refer to this as the **Univariate MRF model**. The real-valued α_i are called MRF parameters and completely determine the probability distribution.

Each solution in any given population gives an equation satisfying the model. Selecting N promising solutions from a population therefore allows us to estimate the distribution by solving

$$A\alpha = F$$

Where,
A is the $N \times n$ dimensional matrix of allele values in the selected set,
α is the vector of MRF parameters $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$,

F is the N dimensional vector containing $-\ln(f(x))$ of the selected set of solutions x.

Solving this system of linear equations, we get the set of MRF parameters $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. Depending on the relationship between N and n, the system will be under-, over-, or precisely specified. A standard least-squares fitting algorithm can be used to give a maximum likelihood estimation of the α_i.

For mathematical reasons, we use {-1, 1} as the values of x_i in our model rather than {0, 1}. This ensures arithmetical symmetry between the possible allele values. Another way of saying this is that the magnitude of the energy contribution from an order 1 building block x_i is completely determined by the MRF parameter α_i and is independent of the allele value. As a consequence of this, the energy contribution from a particular variable will be positive or negative depending on the product of signs of the MRF parameter and the variable value. More precisely, α_i will indicate whether x_i is likely to be -1 or 1.

In the next section we describe DEUM, an EDA that uses the univariate MRF model to direct the search for better solutions.

4 DEUM: Distribution Estimation using Markov Random Fields

The workflow of DEUM is similar to that of other univariate EDAs such as PBIL, UMDA and cGA. DEUM begins by initializing a probability vector where each element of the vector is assigned the value 0.5. M solutions are then generated by sampling the probability vector, out of which N best solutions are selected. MRF parameters are then calculated by fitting the univariate MRF model on the selected set and solving the system of linear equations. DEUM uses the Singular Value Decomposition (SVD) [21] technique to solve the system of linear equations. SVD proves to be the most stable technique and can solve systems of linear equations that are under-specified or over-specified. The MRF parameters are then used to update the probability vector, which is then sampled M times to create a new population. This process continues until termination criteria are satisfied.

Pseudo-code for DEUM is as follows:

```
1. Initialize probability vector $p = \{p_1, p_2, ..., p_n\}$ by assigning 0.5 to each $p_i$
2. Sample $p$ to generate $M$ number of parent solution
3. Select $N$ fittest solutions from parent where $N \leq M$
4. Calculate MRF parameters $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ by applying univariate MRF model
5. Use $\alpha$ 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
```

λ is a **learning rate** [1] parameter specified by the user which takes a value between 0 and 1. λ has a direct effect on the convergence speed of DEUM 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 updating rule uses the sign of the MRF parameter to direct the search towards favouring a particular value of x_i. This is achieved by updating the probability vector p_i in the appropriate direction by a fixed learning rate λ. Note that when selection size $N = 1$, the DEUM updating rule will always have an identical effect to the PBIL updating rule.

The updating rule should be understood in terms of the energy function. To maximise fitness, we are trying to minimise energy because of the negative log relationship between energy and fitness represented by the univariate MRF model. When all of the terms on the right hand side of our system of equations are non-positive, i.e., on the condition that $-\ln(f(x)) \leq 0$, for all solutions in the selected set, a negative α_i indicates that x_i is more likely to be 1 and a positive α_i indicates that x_i is more likely to be -1. So the updating rule moves the marginal probability in the direction indicated by the α_i. This condition is equivalent to $f(x) \geq 1$ for all solutions x in the selected set. This is easily achieved on the problems we consider in this paper and held true on all runs of each of the experiments presented in the next section. For problems where $f(x) < 1$ is likely to occur in a selected set of promising solutions, DEUM should be applied by adding 1 to the fitness function. This will not affect the selection process of the algorithm as DEUM selects the fittest N solutions out of M.

5 Experimental results

In this section, we compare DEUM with other univariate EDAs and a GA on three different problems. In order to compare best with best, we empirically determined the parameters for DEUM. 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.

5.1 Onemax Problem

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

$$f_{onemax}(x) = \sum_{i=1}^{n} x_i$$

Where, x_i is the value of i^{th} variable of set x and the fitness is the sum of all bits in x. This problem is decomposable of order one 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 [15].

We compare performance of DEUM 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 with chromosomes ranging in size between 30 and 180. The number of fitness evaluations taken to find the solution was recorded for each run. Uniform crossover with exchange probability of 0.5 was used for 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 (40 - 270). Learning rate λ was from 0.5 to 0.6 for DEUM. Truncation selection with selection

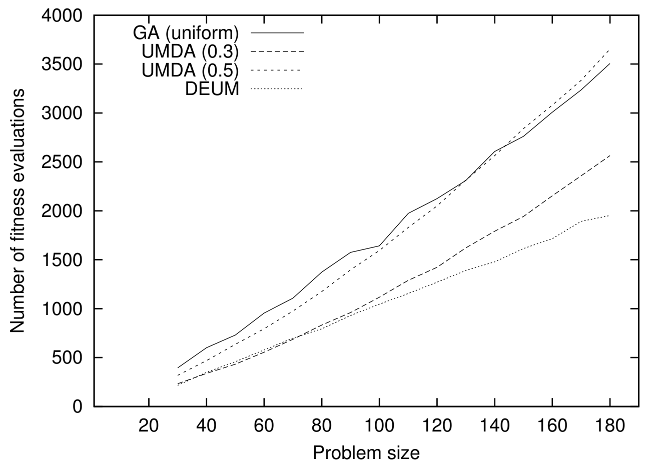


Figure 1: Average number of fitness evaluations for 30 to 180 sized onemax problem where the population size was 40 -100 for GA (uniform), 50 - 170 for both variant of UMDA and $1.5n$ for DEUM which is 40 - 300. λ for DEUM was 0.5-0.6.

size $N = 0.5M$ was used for GA (uniform). For DEUM selection size of $N = 0.85M$ was used. No elitism was used and new populations were generated with complete replacement. Fig. 1 shows the average number of fitness evaluations for each algorithm over the range of onemax problems.

The success ratio of converging to the optimum was 93.5% for DEUM, 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 Fig. 1.

As we can see from Fig. 1, 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 has a comparable performance to UMDA (0.3) for small problems and performs better for large problems.

5.2 Schaffer f6 Function optimization

The Schaffer f6 function, described in [4], is an interesting function for optimization, which has been frequently used to evaluate the performance of GAs. A simplified version of it is presented below.

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

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

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

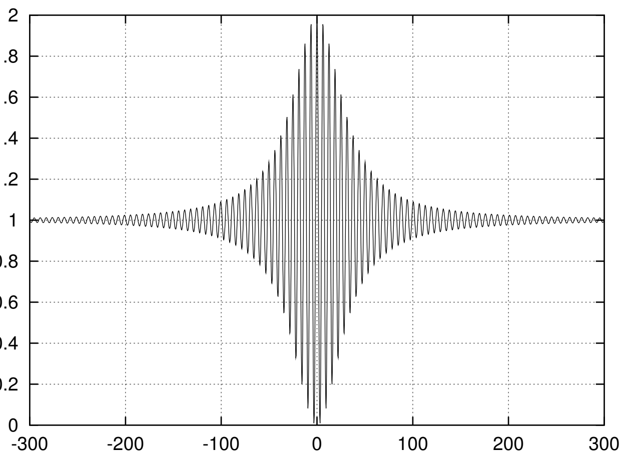


Figure 2: Fitness landscape for simplified version of Schaffer f6 function

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, 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 $f(x) > 1.99999988079071$. The experimental results are shown in Fig. 3.

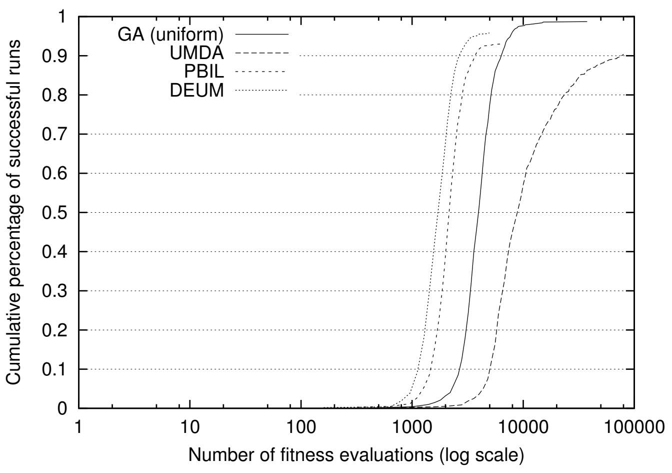


Figure 3: Experimental results in the form of RLD showing, for each algorithm running on the 20-bit binary representation of Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

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, 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. The experimental results are shown in Fig. 4. The result for UMDA is not shown in the figure as its performance was much worst than other three algorithms.

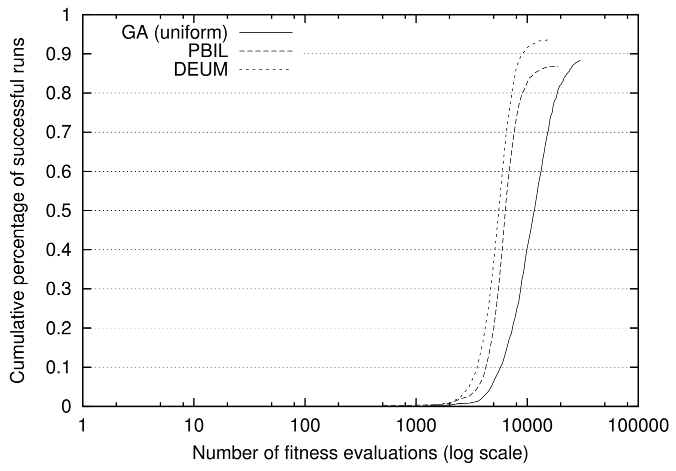


Figure 4: Experimental results in the form of RLD showing, for each algorithm running on the 20-bit gray code representation of Schaffer f6 function, the cumulative percentage of successful runs that terminated within a certain number of function evaluations

Figs. 3 and 4 shows the Run Length Distribution (RLD) [10] for each of the compared algorithms, i.e., for each algorithm, the cumulative percentage of successful runs is plotted against the number of function evaluations needed to achieve success. RLD is a powerful technique used to understand the dynamic behaviour, that is usually observed in stochastic algorithms like GAs and EDAs. For example, Fig. 3 shows that, with DEUM, 50% of runs found the optimum within 1700 function evaluations in comparison to 2200 function evaluations of PBIL. We can see that, for both binary and gray representations, the performance of DEUM was better than other Univariate EDAs.

As we stated earlier, the performance of DEUM was identical to the performance of PBIL when selection size N was set to 1 (Fig. 5a). The updating rule in this case is effectively identical. However, as selection size increases, the performance of DEUM exceeds that of PBIL (Fig. 5b and 5c).

5.3 Composed trap function of order 5

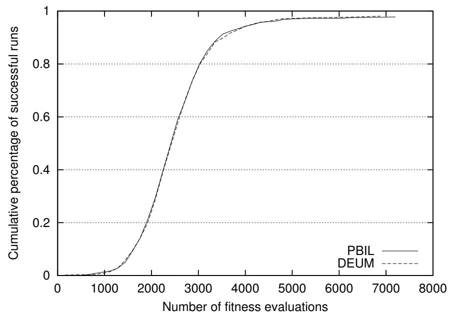
A composed trap function of order k [18] can be defined as

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

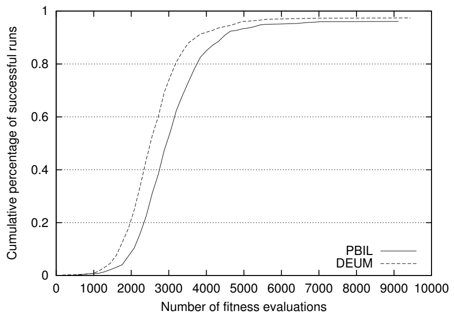
Each block $(x_{b_i,1} + ... + x_{b_i,k})$ gives a fitness which can be calculated through a general trap function of order k

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

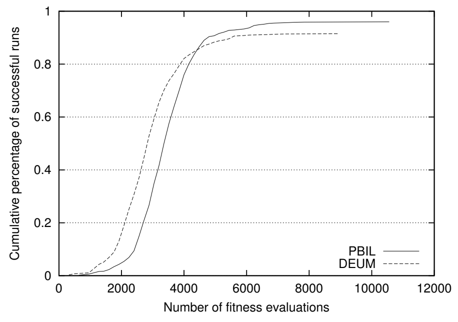
Where, u is the number of ones in the input block of k bits. The trap function of order 5 (Fig. 6) is the instance of the general trap function where $k = 5$, $f_{high} = 5$ and $f_{low} = 4$.



(a) $N = 1$



(b) $N = 2$



(c) $N = 3$

Figure 5: Experimental results comparing DEUM and PBIL for 20-bit binary representation Schaffer f6 function. For both algorithm population size was 160, learning rate was 0.1 and selection size was 1 for (a), 2 for (b) and 3 for (c).

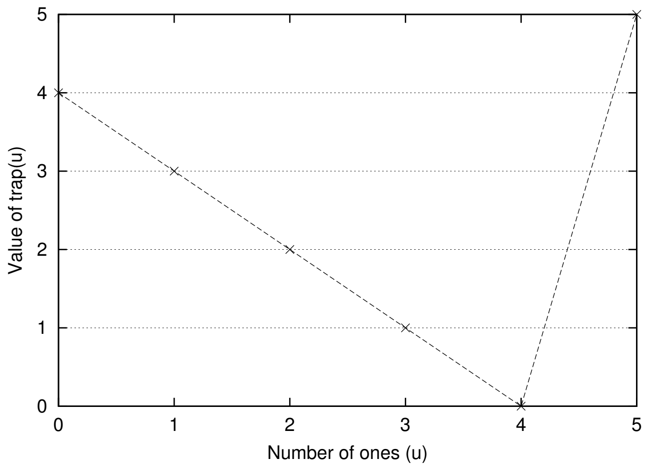


Figure 6: The trap function of order 5, where global optimum is in 11111 and local optimum is in 00000. Any block of bits with $u < 5$ deceives algorithm to the local optimum as u increases.

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, like other univariate EDAs and GA with uniform crossover, is misled by a trap function because it cannot detect the interaction between variables.

Experiments done with problem size $n = 30$ showed that none of GA (uniform), UMDA, PBIL with learning $\lambda = 0.1$ or DEUM 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. 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.

6 Conclusions

In this paper, we have presented DEUM as a novel EDA, which uses MRF modelling of fitness to update the probability vector. The motivation behind DEUM is to use MRF modelling to provide a better estimate of the probability distribution. Our experiments support our hypothesis that, for order one problems, the use of MRF parameters instead of the simple univariate marginal frequency does provide better convergence of the probability vector, leading to better performance in terms of the number of function evaluations required for convergence to the global optimum.

Our experiments with trap function of order 5 showed that DEUM has a similar problem as other univariate EDAs, that is premature convergence to the local optimum. Our future work will focus on developing MRF model on bivariate and multivariate EDAs, so that the problems with higher order variable interactions can also be tackled.

At this stage of our research, DEUM only uses the sign of MRF parameters to update probability vector. However, it would be interesting to use the value of MRF parameters as a weighted energy contributor of a particular allele on the fitness of a chromosome. This means, we need to find the technique to sample the MRF parameter to generate next population. Work on this topic is ongoing and, the encouraging results are expected in near future.

Acknowledgement

We would like to thank the three anonymous reviewers for their useful comments on an earlier version of this paper.

References

[1] Shumeet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,. Technical Report CMU-CS-94-163, Pittsburgh, PA, 1994.

[2] Shumeet Baluja and Scott Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proceedings of the 1997 International Conference on Machine Learning*, 1997.

[3] D. F. Brown, A. B. Garmendia-Doval, and J. A. W. McCall. Markov Random Field Modelling of Royal Road Genetic Algorithms. *Evolution Artificielle 2001*, page 12, 2001.

[4] Lawrence Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[5] Jeremy S. de Bonet, Charles L. Isbell, Jr., and Paul Viola. MIMIC: Finding optima by estimating probability densities. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 424. The MIT Press, 1997.

[6] R. Etxeberria and P. Larrañaga. optimization with bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99.*, page 332339, Cuba, 1999.

[7] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[8] G. Harik. Linkage learning via probabilistic modeling in the ECGA, 1999.

[9] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE-EC*, 3(4):287, November 1999.

[10] Holger H. Hoos and Thomas Stutzle. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence*, 112(1-2):213–232, 1999.

[11] Pedro Larrañaga and Jose A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[12] S. Z. Li. *Markov Random Field modeling in computer vision*. Springer-Verlag, 1995.

[13] Heinz Mühlenbein and Thilo Mahnig. Convergence theory and application of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7(1):19–32, 1999.

[14] Heinz Mühlenbein and Thilo Mahnig. FDA - A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376, 1999.

[15] Heinz Mühlenbein and Gerhard Paass. From recombination of genes to the estimation of distributions: I. binary parameters. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 178–187, Berlin, 1996. Springer.

[16] M. Pelikan, D. E. Goldberg, and E. Cant'u-Paz. BOA: The Bayesian Optimization Algorithm. In W. Banzhaf et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO99*, volume I, pages 525–532, San Fransisco, CA, 1999. Morgan Kaufmann Publishers.

[17] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. Technical Report 99018, Illinois Genetic Algorithms Lab, UIUC, Urbana, IL, 1999.

[18] Martin Pelikan. *Bayesian optimization algorithm: From single level to hierarchy*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002. Also IlliGAL Report No. 2002023.

[19] Martin Pelikan and David E. Goldberg. Hierarchical BOA solves Ising spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 1271–1282, 2003. Also IlliGAL Report No. 2003001.

[20] Martin Pelikan and Heinz Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, pages 521–535, London, 1999. Springer-Verlag.

[21] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2nd edition, 1993.