Using a Markov Network Model in a Univariate EDA: An Empirical Cost-Benefit Analysis

Siddhartha Shakya
School of Computing
The Robert Gordon University
Aberdeen, UK
ss@comp.rgu.ac.uk

John McCall
School of Computing
The Robert Gordon University
Aberdeen, UK
jm@comp.rgu.ac.uk

Deryck Brown
School of Computing
The Robert Gordon University
Aberdeen, UK
db@comp.rgu.ac.uk

ABSTRACT

This paper presents an empirical cost-benefit analysis of an algorithm called *Distribution Estimation Using MRF with direct sampling* (DEUM$_d$). DEUM$_d$ belongs to the family of Estimation of Distribution Algorithm (EDA). Particularly it is a univariate EDA. DEUM$_d$ uses a computationally more expensive model to estimate the probability distribution than other univariate EDAs. We investigate the performance of DEUM$_d$ in a range of optimization problem. Our experiments shows a better performance (in terms of the number of fitness evaluation needed by the algorithm to find a solution and the quality of the solution) of DEUM$_d$ on most of the problems analysed in this paper in comparison to that of other univariate EDAs. We conclude that use of a Markov Network in a univariate EDA can be of net benefit in defined set of circumstances.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

; G.3 [**Probability and statistics**]: Probabilistic algorithms, Stochastic processes

General Terms

Algorithms, Performance, Theory

Keywords

Estimation of Distribution Algorithms, Evolutionary Computation, Probabilistic Modelling

1. INTRODUCTION

Estimation of Distribution Algorithms (EDAs) [17] is a well-established topic in the field of evolutionary algorithms. EDAs are motivated by the idea of identifying and preserving important patterns or building blocks [6] and are able to solve problems that are known to be hard for traditional Genetic Algorithms (GA) [20]. An EDA maintains the *selection* and *variation* concepts of evolution. However, it replaces the crossover and mutation approach to variation in a traditional GA by building and sampling a probabilistic model of promising solutions. The processing of the building blocks in an EDA is explicitly biased towards the significant patterns identified by a probabilistic model. This contrasts with the implicit processing of building blocks in a traditional GA. EDAs are classified as univariate, bivariate or multivariate [21, 11] according to the type of interaction between allele values that is allowed in the model of the probability distribution.

For the purpose of this paper, we will concentrate on univariate EDAs. Particularly, on an algorithm, described in [23], which we call **D**istribution **E**stimation **U**sing **M**arkov Random Field with **d**irect sampling (DEUM$_d$). DEUM$_d$ is a modification to the algorithm called *Distribution Estimation Using MRF* (DEUM) proposed in [24]. DEUM$_d$ uses a Markov Random Field (MRF) model [3, 12] to estimate the probability distribution. Markov Random Field models are a class of *Undirected Graphical Models* (also known as *Markov Networks*) [12, 18]. A previously proposed EDA, known as *Factorization of the Distribution Algorithm* (FDA) [13] also uses an Undirected Graphical Model to estimate the probability distribution. However, FDA is distinct from DEUM$_d$ in significant ways. Particularly, in its use of a Triangular model of the distribution and it's restriction to a certain class of fitness function. And also, FDA is a multivariate EDA. (see [15, 13] for more details on FDA).

In most univariate EDAs, the probability distribution is estimated using the marginal frequency of particular allele values in a selected subset of the population (e.g., see [17], [21]). In particular, the univariate marginal distribution algorithm (UMDA) proposed in [17] uses marginal frequencies that are sampled to generate successive populations. In DEUM$_d$, these marginal frequencies are replaced with a MRF model also built from a selected subset of the population. This model gives a maximum likelihood estimation of the optimal solution for the selected set, and it is sampled to generate a successive population. The MRF approach to the estimation of distribution is computationally more expensive than the marginal frequency approach. However, there could be a trade-off between computational cost and the quality of results.

The purpose of this paper is to empirically explore the cost and benefit of applying the MRF models used by DEUM$_d$

GECCO'05, June 25–29, 2005, Washington, DC, USA.

over the marginal probability model used by other univariate EDAs. We present an analysis of the performance of DEUM$_d$ on a range of optimization problems and compare this with the performance of other univariate EDAs. We also find it instructive to compare the performance of GA on the same problems.

The rest of the paper is structured as follows. Section 2 gives some background on Univariate EDAs. Section 3 describes the probabilistic model used by DEUM$_d$ and its workflow. Section 4 presents experimental results on the performance of DEUM$_d$ on range of different optimization problems. We analyse our experimental findings in section 5 and present wider conclusions in section 6.

2. BACKGROUND

An EDA regards a solution (chromosome) 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 (an instance of the random field) as $x = \{x_1, x_2, \ldots, x_n\}$ where each x_i is the value taken by the i-th random variable. Here, we consider problems where solutions are encoded as bit-string chromosomes, and so n is the chromosome length, and the x_i represent the allele values in the obvious way (so each x_i is either 0 or 1).

Univariate EDAs do not consider dependencies between variables, i.e., they only model building blocks of order one. In this case, the joint probability distribution, $p(x)$, is simply the product of the univariate marginal probabilities of all variables in a chromosome x:

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

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

As in a traditional GA, an EDA begins by generating an initial population of M solutions. N promising solutions are then selected according to chosen selection criteria (usually fittest) for some $N \leq M$. An estimation of the probability distribution of allele values is then made from the selected set of solutions [17]. Offspring are generated by sampling the probability distribution to replace the current population with a new one. This process continues until a termination criterion is satisfied.

Population Based Incremental Learning (PBIL) [1], the Univariate Marginal Distribution Algorithm (UMDA) [17], and the Compact Genetic Algorithm (cGA) [8] all use a univariate model of the probability distribution.

3. DISTRIBUTION ESTIMATION USING MRF WITH DIRECT SAMPLING

In [3], MRF theory was used to provide a formulation of the joint probability distribution that relates solution fitness, $f(x)$, to an *energy function*, $U(x)$, calculated from the values of the solution variables. To be precise:

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

from which, an equation for each solution x can be derived (see [3] for detail information):

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

Here, $f(x)$ is the fitness of an individual x, $U(x)$ is an energy function derived from allele values and T is a temperature coefficient, which in [3] has a constant value of 1. The summations are over all possible solutions y. $U(x)$ gives the full specification of the joint probability distribution, so it can be regarded 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 interactions between the variables x_i. In [24], a *Univariate MRF model* was used that assumes simple form of energy function with no such interactions. To be precise,

$$U(x) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \tag{4}$$

Here, α_i are called MRF parameters and completely determine the probability distribution. Each variable x_i provides a contribution $\alpha_i x_i$ to overall fitness.

For mathematical reasons, $\{-1, 1\}$ are used as the values of x_i in the model, rather than $\{0, 1\}$. This ensures arithmetical symmetry between the possible allele values.

Each solution in a given population provides an equation satisfying the model. Selecting N promising solutions from a population therefore allows us to estimate the distribution by solving the system of equations:

$$A\alpha^T = F \tag{5}$$

Here, A is the $N \times n$-dimensional matrix of allele values in the selected set, α is the vector of MRF parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, and F is the N-dimensional vector containing $-\ln(f(x))$ of the selected set of solutions x. Depending on the relationship between N and n, the system will be under-, over-, or precisely-specified. A standard fitting algorithm can be used to give a maximum likelihood estimation of the α_i. The α_i can then be used to provide an estimate of the probability of the value of x_i.

In [24], α is used to formulate an updating rule to update a *probability vector*. The probability vector is then sampled to generate a child population. Here, we use α_i to directly estimate the marginal probability $p(x_i)$.

Fixing the value of a particular allele x_i divides the set Ω of all chromosomes into two disjoint sets, which we denote by A and B. More precisely, $A = \{x \in \Omega : x_i = 1\}$ and $B = \{x \in \Omega : x_i = -1\}$. We denote the probability that the allele value in position i is equal to 1 by $p(x_i = 1)$. Clearly, the probability that the allele value in position i is equal to -1 is $1 - p(x_i = 1)$. Applying this to (2), we obtain:

$$p(x_i = 1) = \sum_{x \in A} p(x) = \sum_{x \in A} \frac{e^{-U(x)/T}}{Z} \tag{6}$$

Here, $Z = \sum_y e^{-U(y)/T}$ is a (very large) normalising constant. Substituting for $U(x)$ from (4), and noting that $x_i = 1$ for all $x \in A$, we obtain:

$$p(x_i = 1) = e^{-\alpha_i/T} \frac{K}{Z} \tag{7}$$

where K is a large constant representing the sum over all chromosomes in A of contributions from alleles in positions other than i.

Similarly, summing over B we obtain the probability that the allele value in position i is equal to -1:

$$p(x_i = -1) = 1 - p(x_i = 1) = e^{\alpha_i/T}\frac{K}{Z} \quad (8)$$

Here, K is the same constant as in (7), because the chromosomes in A and B agree pairwise at allele positions other than i. Combining (7) and (8), the constants K and Z drop out, and we get the following expression as an estimate of the marginal probability for $x_i = 1$:

$$p(x_i = 1) = \frac{1}{1 + e^{\beta\alpha_i}} \quad (9)$$

where, $\beta = 2/T$.

Note that, as $T \to 0$, the value of β increases, and the value of $p(x_i = 1)$ tends to limit depending on the sign of 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 our estimation of distribution technique, which combines the univariate MRF model 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.

3.1 Workflow of DEUMd

DEUM$_d$ consists of a five step procedure as follows:
1. Generate an initial population, P, of size M with uniform distribution.
2. Select the N fittest solutions from P, where $N \leq M$.
3. Calculate the MRF parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ by making a maximum likelihood estimation from the selected solution.
4. Generate M new solutions using the following distribution:

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

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.

DEUM$_d$ uses the singular value decomposition (SVD) [22, 7] technique to make the maximum likelihood estimation. SVD proves to be the most stable technique, returning useful estimations from systems of linear equations that are either under- or over-specified [22].

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 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.

4. EXPERIMENTAL RESULTS

The aim of our experiment is to investigate how effective the model of the distribution used in DEUM$_d$ is in comparison to those used in other univariate EDAs. For this purpose, a range of optimization problems from literature has been chosen. Each of these problems has been used in the literature to evaluate different EDAs (see [2, 10, 16, 19, 5]). Some problems are known to be better solved by EDAs and some by GAs. We compare the performance of DEUM$_d$ with two other well known univariate EDAs, 1. PBIL 2. UMDA. We also compare the performance of DEUM$_d$ with GA.

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 fitness could be found, the number of fitness evaluations taken by the algorithm to find the optimum was taken as a measure for performance evaluation. Run Length Distribution (RLD)[9] 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 example, it can be seen in Figure 1 that, for DEUM$_d$ 80% of the runs found optimum solution within 1600 function evaluation in comparision to 2000, 2800 and 3700 of PBIL, UMDA and GA respectively.

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 [10, 5] . This is shown as a table (eg. see Table 4) where the average $\pm$ 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 in the paper. 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 chromosome to be transferred to the child population. The parameters for each algorithm were chosen empirically. For UMDA, PBIL and DEUMd, *truncation selection* was used, i.e. the best N solutions were selected. For GA, different selection methods were tried for each of problem and the best performing method was chosen.

4.1 Onemax Problem

The Onemax problem [17] is a simple linear problem decomposable into building blocks 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 [17]. The Onemax Problem can be defined as:

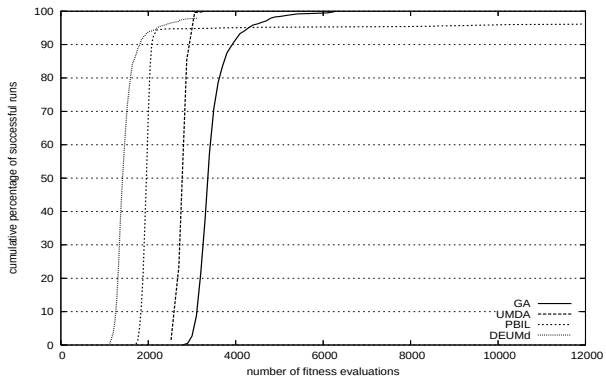

Figure 1: RLD for OneMax

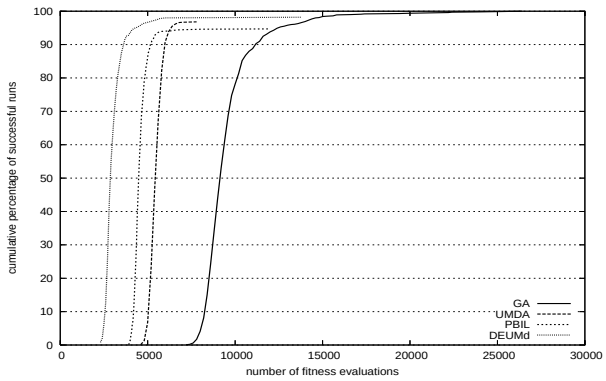

Figure 2: RLD for Plateau

$$F_{om}(x) = \sum_{i=1}^{n} x_i$$

The objective is to maximize the function F_{om} with $x_i \in \{0, 1\}$. The global optimum is located at the point (1,1,.....,1). We take the problem dimension 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 1:

Table 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	-	-	-	-
$DEUM_d$	40	10	-	4	-	-	-

For GA, the truncation selection and the uniform crossover were used. The results in the form of RLD are shown in Figure 1.

4.2 Plateau problem

This problem was proposed in [14] and is used by [11] 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,

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

The goal is to maximaise 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 algorithm 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 2:

Table 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 2.

4.3 Checkerboard problem

In Checkerboard problem [2, 11], 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 Kronecker delta function. Then the checkerboard function can be written as:

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

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

We follow the approach taken by [11, 5] 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 3:

Table 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

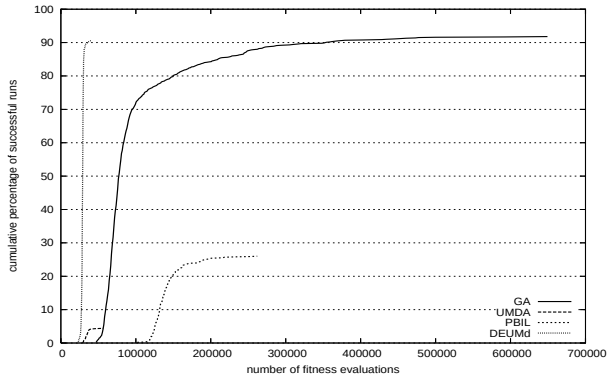

Figure 3: RLD for CheckerBoard

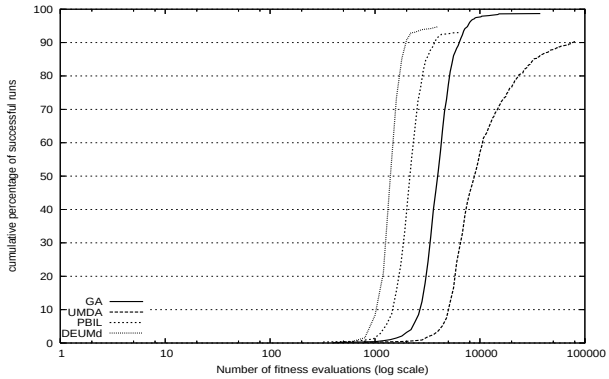

Figure 4: RLD for F6 function

Figure 3. As the percentage of successful runs was low, the mean and standard deviation for fitness and the number of evaluation are also shown, in Table 4.

Table 4: mean ± stdev of fitness and number of fitness evaluation for each algorithm on Checker board problem

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

4.4 Schaffer F6 function

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

$$F_6(x) = 1 + \left(\frac{\cos(x)}{1 + 0.001x^2} \right)$$

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

An interesting feature of this function is that it has many local optima, but a single global optimal solution. 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 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 5:

Table 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 4.

4.5 Equal products function

This problem is presented in [2, 5]. 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:

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

where,

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

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 (chromosome length) to be 50. Following [5], 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:

Table 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 7: mean ± stdev of fitness and number of fitness evaluation for each algorithm on Equal products problem

GA	UMDA	PBIL	DEUM$_d$
211.59 ± (1058.47)	5.03 ± (18.29)	9.35 ± (43.36)	2.14 ± (6.56)
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 instances of the problem, the RLD could not be shown. Results are shown in Table 7.

4.6 Colville function

This is a minimization problem [5]. 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, $-10 \leq x_i \geq 10$.

We have taken the chromosome length to be 60. The optimum value for F_c is 0. 100 independent runs of each algorithm was executed for this problem. The parameter setups for each of the algorithm are shown in Table 8:

Table 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 9) shows the mean ± standard deviation for fitness and number of evaluation for each of the algorithm.

Table 9: mean ± stdev of fitness and number of fitness evaluation for each algorithm on Colville problem

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

4.7 SixPeaks function

The SixPeaks function [2, 11] 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 tends 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 10:

Table 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 11.

Table 11: mean ± stdev of fitness and number of fitness evaluation for each algorithm on SixPeaks problem

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

4.8 Trap function of order 5

A Trap function of order k [19] can be defined as

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

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 12:

For the GA, tournament selection and onepoint crossover were used. The results in the form of RLD are shown in Figure 5.

Table 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

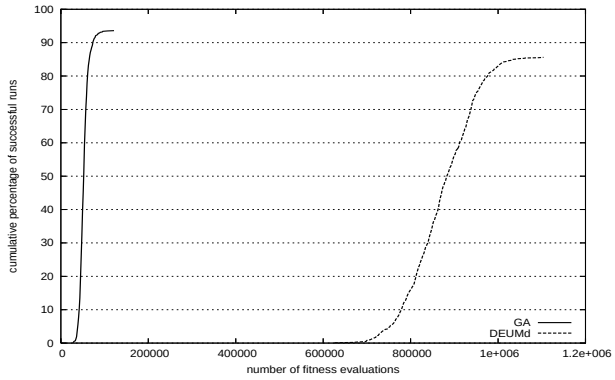

Figure 5: RLD for Trap5

5. ANALYSIS OF RESULTS

Our experimental result 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) and also 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 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 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 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 11 for SixPeaks and Figure 5 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 very hard 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 Table 7 and 9). These results can be verified from the t-test comparison (on quality of fitness) shown in Table 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 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 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. CONCLUSION

The aim of this paper was to empirically explore the cost and benefit of applying MRF models used by DEUM$_d$ over the marginal probability model used by other univariate EDAs. We have presented an empirical analysis on the performance of DEUM$_d$ on a range of optimization problems compared with the performance of other univariate EDAs and GAs.

The computational cost of Estimation of Distribution using MRF model 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 used to make the maximum likelihood estimation of 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 $O(nN^2)$) [25, 7], whereas computational complexity to compute the univariate marginal frequency is $O(nN)$. However, our experiments shows that there is a case to be made for a more sophisticated estimation of distribution in certain circumstances.

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.

The advantage of DEUM$_d$ is that the maximum likelihood estimation used is more sensitive to the distribution than the simpler histogramming method used in other univariate EDAs. Active research is under way to extend this approach to multivariate EDAs. The success of multivariate EDAs on problems with higher order difficulty suggests that further benefits can be gained in this area.

7. ACKNOWLEDGEMENT

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

8. REFERENCES

[1] S. 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] S. Baluja and S. 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. *Lecture Notes in Computer Science*, 2310:65–78, January 2002.

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

[5] L. delaOssa, J. A. Gámez, and J. M. Puerta. Migration of Probability Models Instead of Individuals: An Alternative When Applying the Island Model to EDAs. In *Parallel Problem Solving from Nature VIII*, pages 242–252. Springer, 2004.

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

[7] G. Golub and C. Van Loan. *Matrix Computations*. Baltimore, MD, second edition, 1989.

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

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

[10] P. Larrañaga, R. Etxeberria, J. Lozano, and J. Peña. Optimization by learning and simulation of bayesian and gaussian networks. Technical Report EHU-KZAA-IK-4/99, University of the Basque Country, 1999.

[11] P. Larrañaga and J. 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] H. Mühlenbein. Evolutionary algorithms and the boltzmann distribution. In *Foundations of Genetic Algorithms (FOGA2002)*.

[14] H. Mühlenbein. The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1:pp. 335–360., 1994.

[15] H. Mühlenbein and R. Höns. A theoretical and experimental investigation of estimation of distribution algorithms. In *Optimization by Building and Using Probabilistic Models (OBUPM-2004)*.

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

[17] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions: I. binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 178–187, Berlin, 1996. Springer.

[18] I. Murray and Z. Ghahramani. Bayesian Learning in Undirected Graphical Models: Approximate MCMC algorithms. In *Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, Banff, Canada, 8-11 July 2004.

[19] M. 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.

[20] M. Pelikan and D. 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.

[21] 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.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 2nd edition, 1993.

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

[24] S. K. Shakya, J. A. W. McCall, and D. F. Brown. Updating the probability vector using MRF technique for a Univariate EDA. In E. Onaindia and S. Staab, editors, *Proceedings of the Second Starting AI Researchers' Symposium*, pages 15–25, Valencia, Spain, 2004. IOS press.

[25] R. Suda and S. Kuriyama. Another preprocessing algorithm for generalized one-dimensional fast multipole method. *Journal of Computational Physics*, 195:790–803, 2004.