Incorporating a Metropolis method in a Distribution Estimation using Markov Random Field Algorithm

Siddhartha K. Shakya, John A.W. McCall, Deryck F. Brown
School of Computing, The Robert Gordon University
St. Andrew Street, Aberdeen, AB25 1HG, Scotland, UK
{ss,jm,db}@comp.rgu.ac.uk

Abstract- Markov Random Field (MRF) modelling techniques have been recently proposed as a novel approach to probabilistic modelling for Estimation of Distribution Algorithms (EDAs)[34, 4]. An EDA using this technique, presented in [34], was called Distribution Estimation using Markov Random Fields (DEUM). DEUM was later extended to DEUM$_d$ [32, 33]. DEUM and DEUM$_d$ use a univariate model of probability distribution, and have been shown to perform better than other univariate EDAs for a range of optimization problems. This paper extends DEUM$_d$ to incorporate a simple Metropolis method and empirically shows that for linear univariate problems the proposed univariate MRF models are very effective. In particular, the proposed DEUM$_d$ algorithm can find the solution in $O(n)$ fitness evaluations. Furthermore, we suggest that the Metropolis method can also be used to extend the DEUM approach to multivariate problems.

1 Introduction

Estimation of Distribution Algorithms (EDAs) [24] 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 [10], and are able to solve problems that are known to be hard for traditional Genetic Algorithms (GA) [27]. 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 [29, 16] according to the type of interaction between allele values that can be represented by the model of the probability distribution.

In [34] an algorithm using a Markov network (also known as a Markov Random Field or an undirected graphical model [25, 17]) approach to probabilistic modelling has been proposed. This was called Distribution Estimation Using Markov Random Field (DEUM). DEUM was later extended to DEUM$_d$, which is Distribution Estimation Using Markov Random Field with direct sampling [32, 33]. DEUM and DEUM$_d$ were presented as novel univariate EDAs using a univariate model of probability distribution. They were shown to perform better than other EDAs of their type over a wide range of optimization problems [34, 32, 33].

This paper extends DEUM$_d$ to incorporate a simple *Metropolis method* [18] and shows that for linear univariate problems the proposed univariate MRF models are very effective. In particular, the proposed DEUM$_d$ can find a solution in $O(n)$ fitness evaluations. Furthermore, we suggest that the Metropolis method can also be used to extend the DEUM algorithms to multivariate problems.

The outline of the paper is as follows. Section 2 presents the background on DEUM$_d$ and also presents brief experimental results on its performance in comparison to other univariate EDAs. Section 3 describes a modification to DEUM$_d$ incorporating a simple Metropolis method. Section 4 presents experimental results on a linear univariate problem. Section 5 discusses our immediate future work, which is to extend the DEUM approach to multivariate problems by using a Metropolis method. Section 6 presents a summary and concludes the paper.

2 DEUM with direct sampling

DEUM$_d$, as for other EDAs, 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.

Apart from DEUM$_d$, Population Based Incremental Learning (PBIL) [1], the Univariate Marginal Distribution Algorithm (UMDA) [24], and the Compact Genetic Algorithm (cGA) [13] all use a univariate model of the probability distribution. The main difference between DEUM$_d$ and these other approaches is the method used to estimate the marginal distribution $p(x_i)$. PBIL, UMDA and cGA estimate the marginal probability $p(x_i = 1)$ by taking the frequency of solutions with $x_i = 1$ in a selected set of solutions divided by the size of the selected set, i.e.:

$$p(x_i = 1) = \sum_{x \in S} p(x) \ \vdots \ p(x) = \frac{1}{N} \tag{2}$$

Here, N is the number of selected solutions and S is a subset of the selected set of solutions having $x_i = 1$.

This approach of estimating the marginal probability is very simple and fully depends on the pattern of 1's and 0's in the selected set of solution. This contrasts with the technique used in DEUM$_d$, which we describe in the next section.

2.1 MRF approach to probabilistic modelling

DEUM$_d$ uses Markov Random Field models as its probabilistic model. MRFs are also known as *Undirected Graphical Models* or *Markov Networks* [17, 25]. A previously proposed EDA, known as *Factorization of the Distribution Algorithm* (FDA) [20] 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 its restriction to a certain class of fitness function. Moreover, FDA is a multivariate EDA (see [21, 20] for more details on FDA). More recently, another algorithm using a Markov network has been proposed by [31] and was called MN-EDA. MN-EDA has strong similarities with DEUM$_d$ but has its differences as well. Particularly, in its use of *Kikuchi approximations of the probability distribution*. Again, MN-EDA is also a multivariate EDA.

In [4], 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:

$$p(x) = \frac{f(x)}{\sum_y f(y)} \equiv \frac{e^{-U(x)/T}}{\sum_y e^{-U(y)/T}} \tag{3}$$

from which an equation for each solution x can be derived (see [4] for detailed information):

$$-\ln(f(x)) = U(x)/T \tag{4}$$

Here, $f(x)$ is the fitness of an individual x, $U(x)$ is an energy function derived from the allele values, and T is a temperature coefficient, which in [4] 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 [34], a *Univariate MRF model* was used that assumes a simple form of energy function with no interactions. To be precise,

$$U(x) = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \tag{5}$$

Here, the α_i are known as the MRF parameters, and completely determine the probability distribution. Each variable x_i provides a contribution $\alpha_i x_i$ to the 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:

$$A\alpha^T = F \tag{6}$$

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 [34], α is used to formulate an updating rule to update a *probability vector*. The probability vector is then sampled to generate a child population. In [32], this approach has been extended to use the 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 2^n 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 (3), we obtain:

$$p(x_i = 1) = \sum_{x \in A} p(x) = \sum_{x \in A} \frac{e^{-U(x)/T}}{Z} \tag{7}$$

Here, $Z = \sum_y e^{-U(y)/T}$ is a (very large) normalising constant. Substituting for $U(x)$ from (5), and noting that $x_i = 1$ for all $x \in A$, we obtain:

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

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:

$$p(x_i = -1) = 1 - p(x_i = 1) = e^{\alpha_i/T} \frac{K}{Z} \tag{9}$$

Here, K is the same constant as in (8), because the chromosomes in A and B agree pair-wise at allele positions other than i. Combining (8) and (9), the constants K and Z drop out, and we get the following expression as an estimate of the marginal probability for $x_i = 1$:

$$p(x_i = 1) = \frac{1}{1 + e^{\beta\alpha_i}} \tag{10}$$

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 the estimation of distribution technique for DEUM$_d$, 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.

The use of the temperature for EDA has been first proposed in *Boltzmann Estimated Distribution Algorithm* [1] (BEDA) [23], where, a Boltzmann selection has been used to estimate the Boltzmann distribution. A cooling schedule for Boltzmann selection for BEDA (and also for FDA) has been later proposed in [22]. These approach 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$_d$, an approximation to the fitness function is used which is done by building a model of fitness function and fitting it to the population.

2.2 Workflow of DEUM

DEUM$_d$ consists of a five-step procedure as follows:

1. Generate an initial population, P, of size M with a 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 solutions.

4. Generate M new solutions using the following distribution:

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

where, $p(x_i = 1) = 1/(1 + e^{\beta\alpha_i})$ and $p(x_i = -1) = 1/(1 + e^{-\beta\alpha_i})$. 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.

[1]BEDA is a conceptual algorithm as requires sum over exponentially many terms to calculate the distribution [22]

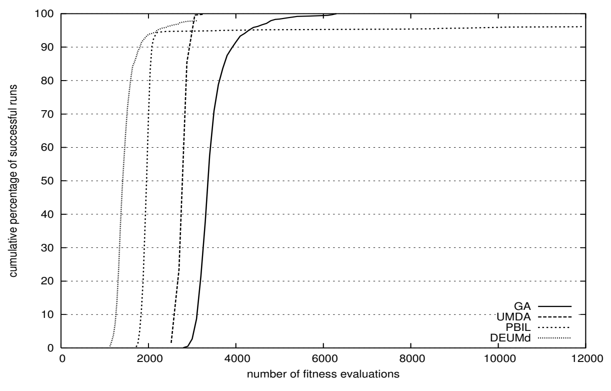

Figure 1: RLD for OneMax

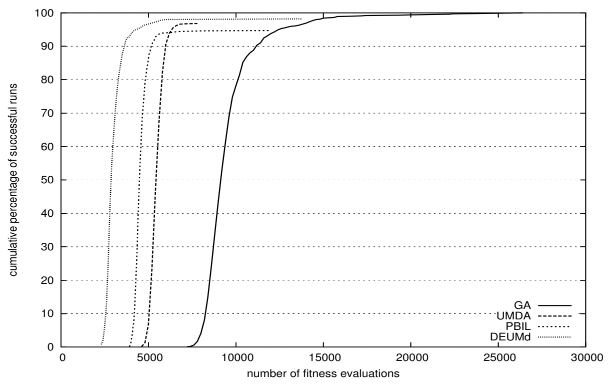

Figure 2: RLD for Plateau

DEUM$_d$ uses the singular value decomposition (SVD) [30, 11] 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 [30].

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.

2.3 Experimental results

Here we briefly review the experimental results on the performance of DEUM$_d$ on a range of optimisation problems. For more details on these experiments, see [33]. The performance of DEUM$_d$ was compared with a GA, PBIL and UMDA.

For the problems where optimum fitness could be found, the number of fitness evaluations taken by each algorithm to

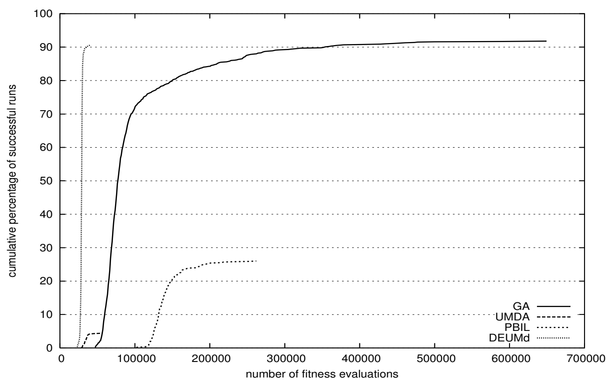

Figure 3: RLD for CheckerBoard

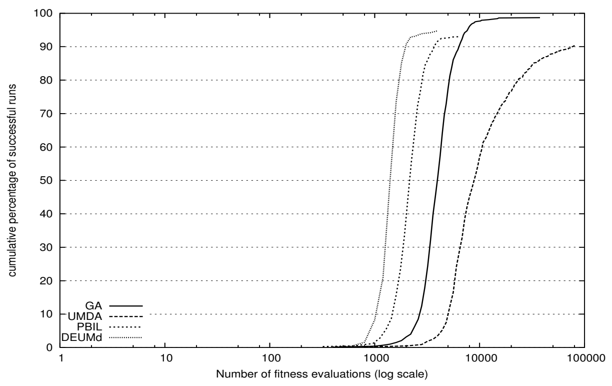

Figure 4: RLD for F6 function

find the optimum was taken as a measure for performance evaluation. Run length distribution (RLD)[14] curves were plotted to measure the performance. The RLD shows, for each algorithm, the cumulative percentage of successful runs that terminated within a certain number of function evaluations. Figure 1 shows the RLD for a 180-bit *OneMax* [24] problem over 1,000 runs for each algorithm. It can be seen that, for DEUM$_d$ 80% of the runs found the optimum solution within 1,600 function evaluations in comparison to 2,000, 2,800 and 3,700 of PBIL, UMDA and the GA respectively. Similarly, figures 2, 3, 4, and 5 show the RLD over 1,000 runs of each algorithm for a 180-bit *Plateau* [19, 16] problem, a 100-bit *Checkerboard* [2, 16] problem, a 20-bit *Schaffer F6 function* [5] and a 180-bit *Trap function of order 5* [26] respectively.

For the problems where the optimum was not known or could not be found, the algorithms were evaluated by the average fitness of solution they could find, and the average number of fitness evaluations taken to find it [15, 7]. The problems addressed were a 50-bit *Equal products function* [2, 7], a 60-bit *Colville function* [7] and a 100-bit *SixPeaks function* [2, 16]. We do not present these results because of the limited space. See [33] for more detail. However, here we highlight the conclusions made from the above experiments. They are as follows:

For the univariate problems(such as OneMax) and also for problems with a low order of dependency between the variables (such as plateau and checker board) the perfor-

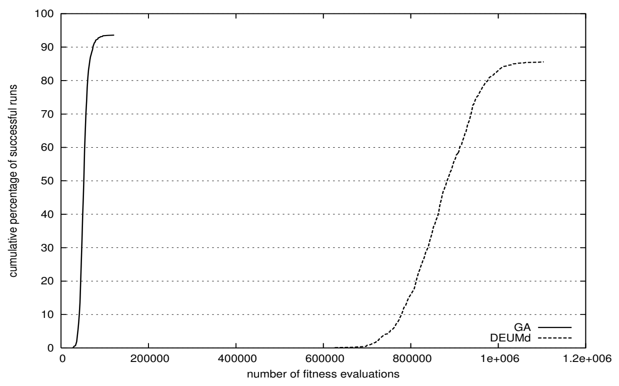

Figure 5: RLD for Trap5

mance 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.

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 the fitness landscape. For the SixPeaks function, none of the algorithms could find the optimum solution. For the trap function, UMDA and PBIL could not find the optimum, even using a very large population size. However, a simple GA with one-point crossover could find the solution after an average of 62,000 fitness evaluations. Interestingly, DEUM$_d$ with a population size of 2,000 could also find the solution, however, with a very large average fitness evaluation, 868,000. It shows that, although DEUM$_d$ is misled by the trap function, by slowing the cooling rate and choosing the correct population size, it still could overcome a trap of order 5.

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 the GA and other univariate EDAs, and was better in some cases.

In general, DEUM$_d$ gave satisfactory results for most of the problems that have been tested, and failed where it was expected to. Again, for most of the problems, the performance of DEUM$_d$ was better than that of other univariate EDAs. These empirical results suggested the effectiveness of MRF models used by DEUM$_d$ over the marginal probability models used by other univariate EDAs. In the next section, we further strengthen this suggestion by showing that, for a linear univariate problem, a simple extension to DEUM$_d$ can find the optimum in $1.5n + 1$ fitness evaluations.

3 Extending DEUM$_d$ to incorporate a Metropolis method

In the MRF modelling literature, *Metropolis methods* [18] are widely used to determine the optimum value for random fields/variables [17]. Here we present a simple variant of it, known as *Zero-Temperature Metropolis method*. Given a set of MRF parameters, α, calculated from a population of chromosomes, it is then possible to approximate the *optimum chromosome*, $x^o = \{x_1^o, x_2^o, ..., x_n^o\}$ (or more precisely *optimum chromosome for the current population*). We call this chromosome, the *Metropolis Population Optimum Chromosome* (MPOC). x^o can be approximated by using the following zero temperature Metropolis method:

1. Generate a chromosome $x^o = \{x_1^o, x_2^o, .., x_n^o\}$ at random.
2. for L iterations, repeat:
   (a) Mutate a variable x_i^o chosen at random to obtain the mutated chromosome $x^{o\prime}$.
   (b) Set $\Delta U = U(x^{o\prime}) - U(x^o)$.
   (c) if $\Delta U < 0$ set $x^o = x^{o\prime}$.
3. Terminate with answer x^o.

For univariate MRF models (5), ΔU can be determined explicitly from the following formula:

$$\Delta U = \alpha_i(x_i^{o\prime} - x_i^o) \tag{11}$$

From (11), we can see that if $\alpha_i < 0$, then (i) $\Delta U < 0$, if $x_i^o = -1$: this suggests accepting the mutation, and (ii) $\Delta U > 0$, if $x_i^o = 1$: this suggests rejecting the mutation. In another words, to make $\Delta U > 0$ (to minimise $U(x^o)$), formula (11) suggests that if $\alpha_i < 0$, x_i^o should be 1 and if $\alpha_i > 0$, x_i^o should be -1. Thus, for a univariate MRF model the MPOC, x^o, can be easily obtained just by looking at the sign of the α_i.

Now let us incorporate this method in DEUM$_d$. This is done by adding two substeps 3.1 and 3.2 in original DEUM$_d$ algorithm. So the redefined DEUM$_d$ will be as follows:

1. Generate an initial population, P, of size M with a uniform distribution.
2. Select the N fittest solutions from P, where $N \leq M$.
3. Calculate the MRF parameters $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ by making a maximum likelihood estimation from the selected solution.
   3.1. Generate MPOC, $x^o = \{x_1^o, x_2^o, ..., x_n^o\}$ where,
   $$x_i^o = \begin{cases} 1 & \text{if } \alpha_i < 0 \\ -1 & \text{if } \alpha_i > 0 \end{cases}$$
   3.2. Terminate, if $f(x^o)$ is optimal/good enough.
4. Generate M new solutions using the following distribution:
   $$p(x) = \prod_{i=1}^{n} p(x_i)$$
   where, $p(x_i = 1) = 1/(1 + e^{\beta\alpha_i})$ and $p(x_i = -1) = 1/(1 + e^{-\beta\alpha_i})$. 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.

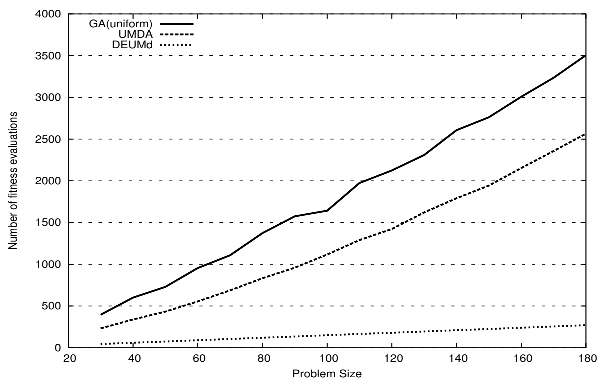

Figure 6: Scalability of DEUM$_d$ on OneMax, showing average number of fitness evaluations for 30- to 180-bit OneMax problems where the population size was 40 to 100 for the GA, 50 to 170 for UMDA, and exactly $1.5n$ for DEUM$_d$.

We observe that the above presented algorithm only uses x^o to check for an optimum and does not use it for further evolution [2]. However, in practice, x^o can be used in various ways to asses the evolution. For example, one could generate one or more x^o and seed them to the next generation. For the purpose of this paper, we do not discuss this topic in detail.

4 Experimental results

The aim of our experiment is to measure the effect of our extension to DEUM$_d$. We show the scalability of DEUM$_d$ on univariate problems, and compare it with that of other univariate EDAs. The obvious test function for this purpose is the OneMax problem [24].

The OneMax problem 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 [2]. We compare the performance of DEUM$_d$ against UMDA and a GA. 100 runs of each algorithm were executed for a series of OneMax problems with chromosomes ranging in size between 30 and 180 bits. The number of fitness evaluations taken to find the optimal solution was recorded for each run. Uniform crossover with exchange probability of 0.5 was used for the GA, crossover was applied all the time and mutation was not applied. The population size, M, ranged from 40 to 100 for the GA, 50 to 170 for UMDA, and was exactly $1.5n$ for DEUM$_d$. τ for DEUM$_d$ was from 5 to 4.

Truncation selection was used where selection size N was $0.5M$ for the GA and $0.3M$ for UMDA. For DEUM$_d$, the selection size was again $1.5n$, i.e., the whole population was selected. No elitism was used and new populations were generated with complete replacement. Figure 6 shows the average number of fitness evaluations for each algorithm over the range of OneMax problems.

The success ratio for converging to the optimum was 100% for DEUM$_d$, 98% for UMDA and 100% for the GA.

[2]For the purpose of our experiment (see section 4), doing so was not necessary

As we can see from Figure 6, UMDA, as it has selection size less than that of GA, has an expected performance better than that of the GA but has less success rate. DEUM$_d$ with a Metropolis method has very stable and efficient performance that is equivalent to $1.5n + 1$ fitness evaluations, i.e., given a randomly-generated initial population of $1.5n$ chromosomes, the x^o generated by the Metropolis method was found to be optimum (Here, +1 is for the fitness evaluation of x^o itself). This result can be compared with the result of a (1+1) EA (also known as *stochastic hillclimber*), an algorithm described by [9] to study the theory of evolutionary algorithms. For the OneMax problem, it has been proved that a (1+1) EA will find the optimum in $O(n \log(n))$ fitness evaluations. Our empirical results show an $O(n)$ performance for DEUM$_d$.

5 Extending DEUM to multivariate EDAs

This section gives an overview of our immediate future work on extending the DEUM algorithms to multivariate EDAs by using the Metropolis method. The work presented here should be seen as work in progress.

To do extend DEUM, let us consider a bivariate MRF model:

$$\begin{aligned} U(x) = \quad & \alpha_1 x_1 + \beta_{1,2} x_1 x_2 + \alpha_2 x_2 + \beta_{2,3} x_2 x_3 + \\ & \ldots + \alpha_n x_n + \beta_{n,1} x_n x_1 \end{aligned} \tag{12}$$

This model (known as the Ising Model [8]) can be interpreted as a *chain* model of dependency proposed in [6]. Here, α and β are the MRF parameters associated with univariate and bivariate interactions between variables respectively. This model completely encapsulates the univariate MRF model, and also incorporates any bivariate relationships between neighbouring variables.

The approximation of both α and β can be done by doing a maximum likelihood estimation over the population of solutions as suggested in Section 2. Given α and β, it is then possible to approximate the MPOC, x^o, using the zero temperature Metropolis method defined in Section 3. For (12) ΔU can be determined explicitly from following formula:

$$\Delta U = (x_i^{o\prime} - x_i^o)(\alpha_i + \beta_{i-1,i} x_{i-1}^o + \beta_{i,i+1} x_{i+1}^o) \tag{13}$$

This formula for calculating ΔU can be easily obtained for any MRF models using following formulation

$$\Delta U = U(x^{o\prime}) - U(x^o)$$

Once we have found x^o, we can then use it to estimate the distribution for the next population. A simple heuristic can be applied for this purpose, i.e., if $x_i^o = 1$, we should increase the probability $p(x_i = 1)$ and if $x_i^o = 0$, we should decrease the probability $p(x_i = 1)$. This heuristic forms the basis for our proposed extension to DEUM for multivariate problems.

A possible workflow for the multivariate DEUM would be as follows:

```
1. Initialize a 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. Find the dependency network and construct an MRF model.
5. Calculate the MRF parameters.
6. Approximate MPOC, $x^0$, using the Metropolis method.
     6.1. if $f(x^o)$ is good enough, terminate.
7. Use $x^0$ to update $p$ using following updating rule
     For $i = 1..n$ do
        If $x_i^o = 1$ then $p_i = p_i(1-\lambda) + \lambda$;
        If $x_i^o = 0$ then $p_i = p_i(1-\lambda)$;
8. Go to step 2 until the termination criteria is satisfied.
```

Step (4) of the algorithm, which is to find a dependency network, is still an open question, and is not addressed in this paper. However, there are various computational techniques that we assume can be used successfully to address this problem. Most of the previously proposed multivariate EDAs already make use of one of these techniques [15, 28, 12, 20]. However, most of them are focused in finding a directed graphical model. Some recent work in EDAs (in particular [31]) has proposed a method for finding an undirected graphical model. However, this topic remains part of our further research.

6 Conclusion

In this paper, we have shown that the univariate MRF models are very effective at addressing linear univariate problems. This is shown by the experimental results carried out with DEUM$_d$ that incorporates a Metropolis method. Furthermore, we show that the Metropolis method can also be used to extend the DEUM algorithm to multivariate EDAs.

The computational cost of approximating MRF parameters for DEUM$_d$ has a polynomial complexity of $O(nN^2)$ (or $O(n^2 N)$ depending upon relationship between n and N) in comparison to the linear complexity, $O(nN)$, of counting the bit frequency for other univariate EDAs [33]. Therefore, the results presented here with OneMax problem can only be seen as of theoretical importance. However, the significantly low number of fitness evaluations needed by DEUM$_d$ suggests that the DEUM$_d$ should be applied for the problems where fitness evaluation is costly and can be trade off against the computation cost of approximating MRF parameters.

Our research so far has focused on the binary representation for the random variables. However, most of the earlier works on the use of MRF modelling has been in integer case [17, 3]. Particularly, in image analysis case [17], each pixel in the Image is a random variable where, even for the simple greyscale image, a pixel can have 256 different value. This shows that the MRF technique can be naturally extended to integer representation of variables. However, more works needed to be done in order to get a full functioning DEUM algorithm for integer representation.

The immediate future work in this area is to implement a multivariate DEUM algorithm. To do this, an effective method of finding the dependencies between variables must be identified. This work is under way, and we expect to find some interesting results in the near future.

Acknowledgments

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

Bibliography

[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] C. A. Bouman. Markov random fields and stochastic image models. Tutorial presented at IEEE International Conference on Image Processing, Washington, 23-25 Oct, 1995.

[4] 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.

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

[6] J. S. de Bonet, C. L. Isbell, Jr., and P. Viola. MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 424. The MIT Press, 1997.

[7] 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.

[8] H. Derin and P. A. Kelly. Discrete-index Markov-type random fields. *Proceedings of the IEEE*, 77:1485–1510, 1989.

[9] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

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

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

[12] G. Harik. Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, 1999.

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

[14] 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.

[15] 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.

[16] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

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

[18] N. Metropolis. Equations of state calculations by fast computational machine. *Journal of Chemical Physics*, 21:1087–1091, 1953.

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

[20] H. Mühlenbein. Evolutionary algorithms and the boltzmann distribution. In *Foundations of Genetic Algorithms (FOGA2002)*, 2002.

[21] 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)*, 2004.

[22] H. Mühlenbein and T. Mahnig. A new adaptive boltzmann selection schedule sds. In *Proceedings of the 2001 Congress on Evolutionary Computation*, 2001.

[23] H. Mühlenbein, T. Mahnig, and A. R. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247, 1999.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] R. Santana. Estimation of Distribution Algorithms with Kikuchi Approximation. *Evolutonary Computation*, 13:67–98, 2005.

[32] 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, *Proceedings of the International Conference on Adaptive and Natural computiNG Algorithms (ICANNGA 2005)*, pages 202–205, Coimbra, Portugal, 2005. Springer-Verlag, Wien.

[33] S. Shakya, J. McCall, and D. Brown. Using a Markov Network Model in a Univariate EDA: An Emperical Cost-Benefit Analysis. In *Proceedings of Genetic and Evolutionary Computation COnference (GECCO 2005)*, Washington, D.C., USA, 2005. ACM.

[34] 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.