A Markovianity based Optimisation Algorithm

Siddhartha Shakya
Intelligent Systems Research Centre,
BT Group Chief Technology Office
Adastral Park, Ipswich, UK
sid.shakya@bt.com

Roberto Santana
Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizábal 1, 20018. San Sebastian - Donostia, Spain
roberto.santana@ehu.es

Abstract

Several EDAs based on Markov networks have been recently proposed. Key ideas behind these EDAs were to factorise the joint probability distribution in terms of the cliques in the undirected graph. As such, they make use of the global Markov property of the Markov network in one form or another. This paper presents a Markov Network based EDA that exploits Gibbs sampling to sample from the Local Markov property, the Markovianity, and does not directly model the joint distribution. We call it Markovianity based Optimisation Algorithm. We also present some initial results on the performance of the proposed algorithm which show that it can solve problems with complex interaction between variables and its performance compares well with other Bayesian network based EDAs.

1 Introduction

Estimation of Distribution Algorithms (EDAs) [19][12] are a class of population based optimisation algorithm that extracts statistical information from the population of solutions and uses it to generate new solutions. They are able to solve problems that are known to be hard for traditional Genetic Algorithms (GA) [26]. 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 solutions. The evolution process in an EDA is explicitly biased towards the significant patterns identified by the probabilistic models. This contrasts with the more implicit processing of important patterns in traditional GAs. EDAs are classified as univariate, bivariate or multivariate [28][12] according to the type of interaction between variables in the solution that can be represented by the model of the probability distribution.

Much research in EDAs focuses on different approaches to probabilistic modelling and sampling. Particularly, directed graphical models (Bayesian networks) [23] have been widely studied and are well established as a useful approach for modelling the distribution in EDAs. Some of the well known instances of Bayesian network based EDA includes Bayesian Optimisation Algorithm (BOA) [27], hierarchical Bayesian Optimisation Algorithm [25], Estimation of Bayesian Network Algorithm (EBNA) [4][11] and Learning Factorised Distribution Algorithm (LFDA) [17]. Some research on the use of undirected graphical models (Markov networks) [2][15][21] in EDAs has also been done [30][35][36][31][33][32][34]. Some of the well known instances of Markov network based EDA includes DEUM [33] and MN-EDA [31]. Most of these Markov network based EDAs are based on the idea of factorising the joint probability distribution from an undirected graph. More precisely, they make use of the global Markov property of the Markov network, the joint probability distribution, in one form or another.

This paper describes an EDA based on local Markov property, the Markovianity, that does not explicitly factorise the joint probability distribution. Instead, it estimates conditional probabilities in terms of neighbouring nodes in the undirected graph and samples from them using Gibbs sampler. We call it the Markovianity based Optimisation Algorithm (MOA) [1]. MOA incorporates features that have been independently employed in previous implementations of EDAs based on Markov models, but have not been used together. The resulting algorithm is qualitatively different to its predecessors. It does structural learning of the probabilistic model from the data but it can also take advantage of a priori structural information in a straightforward way. Complex approximations to the joint probability distribution are avoided and the temperature parameter is included to balance the exploration and exploitation of the search space. The use of Gibbs sampling remains as a key component, allowing MOA to deal with interactions represented by cycles.

The two main objectives of this paper are:

1. to introduce a Markov network based EDA using the local Markov property and avoid the complications involved in estimating joint probability distribution for a Markov network
2. to use Gibbs sampling as the method to sample from the undirected model

The outline of the paper is as follows. Section 2 gives background on the use of probabilistic graphical models in EDAs. It also introduces the local and global properties of Markov Network. Section 3 reviews several Markov network based EDAs that make use of the global Markov properties. Section 4 presents a detailed description of the MOA. Section 5 presents the experimental results on the performance of MOA on several test functions. It also compares the performance of MOA with Gibbs sampling with the one without it. Section 6 highlights some future work and concludes the paper.

2 EDAs and Probabilistic graphical models

An EDA regards a solution, $x = \{x_1, x_2, .., x_n\}$, as a set of values taken by a set of variables, $X = \{X_1, X_2, ..., X_n\}$. EDAs begin by initialising a population of solutions, P. A set of promising solutions D is then selected from P, and is used to estimate a probabilistic model of X. The model is then sampled to generate the next population. Figure 1 shows the general EDA workflow.

[1]Initial description of MOA is published in [38]

Estimation of Distribution Algorithm

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

Figure 1: The workflow of the general Estimation of Distribution Algorithm

Estimation of the probability distribution lies in the very heart of an EDA. Its effectiveness largely depends on how well it estimates and samples the probability distribution. This is where probabilistic graphical models [13] can be useful. Probabilistic graphical models provide an efficient and effective tool to represent the probability distribution of the random variables. They can be seen as a merger of two disciplines, probability theory and graph theory [9]. They are mainly categorised into two groups [2]. 1) Directed models (Bayesian networks) and 2) Undirected models (Markov networks / Markov Random Fields).

2.1 Bayesian networks

A Bayesian network can be regarded as a pair (B, Θ), where B is the structure of the model and the Θ is a set of parameters of the model. The structure B is a *Directed Acyclic Graph (DAG)*[3], where each node corresponds to a variable in the modelled data set and each edge corresponds to a conditional dependency. A set of nodes Π_i is said to be the parent of X_i if there are edges from each variable in Π_i pointing to X_i. The parameter $\Theta = \{p(x_1|\Pi_1), p(x_2|\Pi_2), ..., p(x_n|\Pi_n)\}$ of the model is the set of conditional probabilities, where each $p(x_i|\Pi_i)$ is the set of probabilities associated with a variable $X_i = x_i$ given it's parent variables Π_i. A Bayesian network is characterized in terms of the joint probability distribution of the variables in the modelled dataset as

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

2.2 Markov networks

A Markov network is a pair (G, Ψ), where G is the structure and the Ψ is the parameter set of the network. G is an undirected graph where each node corresponds to a random variable in

[2]There are several other categories of probabilistic graphical model such as factor graph and mixture models. However, for the purpose of this paper, we limit them to two categories.

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

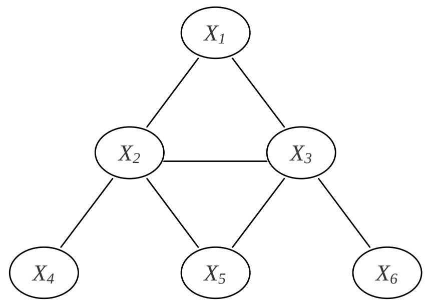


Figure 2: A Markov network structure on 6 random variables

the modelled data set and each edge corresponds to conditional dependencies between variables. However, unlike Bayesian networks, the edges in Markov networks are undirected. Here, the relationship between two nodes should be seen as a *neighbourhood relationship*, rather than a parenthood relationship. We use $N = \{N_1, N_2, ..., N_n\}$ to define a *neighbourhood system* on G, where each N_i is the set of nodes neighbouring to a node X_i. Figure 2 shows an example of a Markov network structure on 6 random variables. Here, variable X_1 has 2 neighbours, $N_1 = \{X_2, X_3\}$. Similarly, variable X_2 has 4 neighbours $N_2 = \{X_1, X_3, X_4, X_5\}$.

A Markov network is characterised in terms of neighbourhood relationship between variables by its *local Markov property* known as *Markovianity* [2][15], which states that the conditional probability of a node X_i given the rest of the variables can be completely defined in terms of the conditional probability of the node given its neighboring states N_i. N_i is sometimes referred to as *Markov Blanket* for X_i [20]. In terms of probability it can be written as

$$p(x_i|x - \{x_i\}) = p(x_i|N_i) \tag{2}$$

A Markov network is also characterised in terms of *cliques*[4] in the undirected graph by its global property, the joint probability distribution, and can be written as

$$p(x) = \frac{1}{Z}\prod_{i=1}^{m}\psi_i(c_i) \tag{3}$$

Where, $\psi_i(c_i)$ (or more precisely $\psi_i(C_i = c_i)$) is a *potential function* on clique $C_i \in X$, m is the number of cliques in the structure G. $Z = \sum_{x\in\Omega}\prod_{i=1}^{m}\psi_i(c_i)$ is the normalising constant known as the *partition function* which ensures that $\sum_{x\in\Omega} p(x) = 1$. Here, Ω is the set of all possible combination of the variables in X.

Equivalently, using Hammersley-Clifford theorem [7], the global Markov property can also be written in terms of Gibbs distribution as

$$p(x) = \frac{e^{-U(x)/T}}{Z} \tag{4}$$

where,

$$Z = \sum_{y\in\Omega} e^{-U(y)/T} \tag{5}$$

[4]Given an undirected graph G, a clique is a fully connected subset of the nodes. For example, in Figure 2, variables $\{X_1, X_2, X_3\}$ define a clique.

is a normalising constant, T is a parameter of the Gibbs distribution known as the *temperature* and $U(x)$ (or more precisely $U(X = x)$) is known as the *energy* of the distribution.

Given an undirected graph, G, on X, energy, $U(x)$, is defined as a sum of *potential functions* over the cliques, C_i, in G.

$$U(x) = \sum_{i=1}^{m} u_i(c_i) \tag{6}$$

Here, $u_i(c_i)$ (or more precisely $u_i(C_i = c_i)$) is a potential function defined over a clique $C_i \in X$. Equation (4), in terms of clique potential function, can also be written as

$$p(x) = \frac{e^{-\sum_{i=1}^{m} u_i(c_i)/T}}{Z} \tag{7}$$

Note that the relationship between $\psi_i(c_i)$ in (3) and $u_i(c_i)$ in (7) is defined as

$$\psi_i(c_i) = e^{-u_i(c_i)/T} \tag{8}$$

It is the clique potential function $u_i(c_i)$, that captures the interaction between variables in the clique c_i, and should be carefully defined in order to get a desired behaviour of the Markov network. We do not go into detail on different ways to defining the clique potential functions, interested readers are advised to see [15], [33].

3 Markov network based EDAs

Most of the EDAs based on Markov network use its global property (3) in one form or another. More precisely, they factorise the joint probability distribution in terms of the cliques in the undirected graph and sample it to generate new solutions.

Three main categories can be distinguished in this class of Markov network based EDAs. They are:

1. Distribution Estimation using Markov network algorithm (DEUM)
2. Markov Network Estimation of Distribution Algorithm (MN-EDA), Markov network Factorised Distribution Algorithm (MN-FDA)
3. Factorised distribution algorithm (FDA)

DEUM [35][33] is a family of Markov network based EDA that builds a model of fitness function in terms of the cliques in the undirected graph and factorises joint probability as a Gibbs distribution. The parameters of the fitness model is then estimated from the population of solutions and Markov chain Monte Carlo simulations, including Gibbs sampler [5] and Metropolis sampler [16], are used to sample new solutions. Several variants of DEUM have been proposed and are found to perform well in comparison to other EDAs of their class in range of different test problems, including Ising Spin Glass and SAT. [35][36][37].

MN-EDA [31] and MN-FDA [30] are based on the idea of making an approximation to the joint probability distribution in terms of cliques in the undirected graph. MN-EDA does so by means of Kikuchi approximation [10] of the joint distribution and uses a Gibbs sampler to sample the new solutions. Similarly, MN-FDA constructs a junction graph [30] from the undirected structure that approximates the joint probability, which is then sampled using Probabilistic logic sampling (PLS) [8] to generate new solutions.

FDA is one of the early EDAs proposed by [18]. Based on the *running intersection property* [13] of an undirected graph, it first identifies *residuals* and *separators* from the undirected structure and constructs a junction tree [14] that completely specifies the joint probability distribution. Junction tree is then sampled using PLS to generate new solution. An FDA able to learn a junction tree from the data was introduced in [22].

4 Markovianity based Optimisation Algorithm (MOA)

All Markov network based EDAs described in previous section approximate and sample global Markov property in one form or another. However, the local Markov property, as defined in (2), can be directly used in EDAs without requiring to define the joint probability distribution. Here we describe an EDA using the local Markov property, the Markovianity. We call it Markovianity based Optimisation Algorithm (MOA). Since, it only exploits the local Markov property, MOA can be seen as the subset of the other global Markov property based EDAs, with a simpler workflow. Furthermore, in addition to gain in efficiency, it avoids the numerical operations associated to the computation of potentials or Kikuchi approximation, which may also represent gains in model accuracy.

Markovianity based Optimisation Algorithm

1. Generate initial (parent) population P of size M
2. Select set D from P consisting of N solutions, where $N <= M$
3. Estimate structure of a Markov network from D
4. Estimate local Markov conditional probabilities, $p(x_i|N_i)$, for each variable X_i as defined by the undirected structure and sample them to generate new population
5. Replace old population by new one and go to step 2 until termination criteria are meet

Figure 3: The workflow of Markovianity based Optimisation Algorithm

Figure 3 shows the workflow of MOA. It starts by generating a population of solutions. A set of solutions is then selected from the population using a selection method, which are then used to estimate the structure of the Markov network. The conditional probabilities defined by the local Markov property (2) are then estimated from the selected set of solutions and sampled to generate the new population.

A number of different approaches can be used in order to estimate an undirected structure. For the purpose of this paper we implement a mutual information based approach [5]. More precisely, we estimate cross entropy of each pair of variables in the solution to create a matrix of mutual information. The pairs with the mutual information higher than a certain threshold are then made neighbours. Also, in order to avoid an overly complex network, we limit the number of neighbours that a variable can have to a certain number. Figure 4 describes the implemented Markov network structure learning algorithm.

Estimating structure - Step 3 of MOA

1. Create a matrix of mutual information, MI, by estimating cross entropy for each pair of variables in the solution. Cross entropy between two random variables, A and B, is given by

$$CE(A,B) = \sum_{a,b} p(a,b) log \left(\frac{p(a,b)}{p(a) \cdot p(b)} \right)$$

   where sum is over all possible combinations of A and B, and $p(a,b)$ is the joint probability of $A = a$ and $B = b$ computed from D

2. Create an edge between two variables, if the mutual information between them is higher than the given threshold. Here we compute the threshold, TR as $TR = avg(MI) * sig$, where $avg(MI)$ is the average of the elements of the MI matrix and sig is the significance parameter, which for the purpose of this paper is set to 1.5.

3. If the number of neighbours to a variable is higher than the maximum number, MN, allowed, only keep MN neighbours that have the highest mutual information.

Figure 4: The workflow of an undirected structure learning algorithm

After estimating the structure of the network, the next step is to estimate the conditional probabilities and sample new population from it. By its definition, an undirected structure may contain cycles. Apart from some restricted set of undirected structures, for example those that satisfy running intersection properties and can be formulated as a directed acyclic graph, most of the Markov networks do not satisfy the ancestral ordering of variables needed by PLS. Alternatively, Markov Chain Monte Carlo (MCMC) [16] methods could be used for sampling. We use Gibbs sampler [5], a class of MCMC method, as the sampling method in MOA. A number of different versions of Gibbs sampler can be implemented for this purpose. Figure 5 describes a version that has been implemented for the purpose of this work. Note that each execution of the Gibbs sampler creates a single solution. Multiple execution of Gibbs sampler should be done in order to create the population of solutions.

[5] Mutual information was originally used by [1] to learn tree-based factorisations in EDAs

Gibbs Sampler - Step 4 of MOA

1. Generate a solution $x = \{x_1, x_2, .., x_n\}$ at random.
2. For r iterations (in this paper we set $r = n \times ln(n) \times IT$, where IT, the *iteration coefficient*, is set to 4), do the following:
   (a) Choose a variable x_i from x at random.
   (b) Using selected set of solutions, D, compute conditional probabilities $p(x_i|N_i)$ for each value of x_i as Gibbs probability,
   $$p(x_i|N_i) = \frac{e^{p(x_i,N_i)/T}}{\sum_{x_i'} e^{p(x_i',N_i)/T}}$$
   where sum is over all possible values of x_i. For example, in binary case, where $x_i = \{0, 1\}$, probability of $x_i = 1$ given the value of its neighbours N_i is written as
   $$p(1|N_i) = \frac{e^{p(1,N_i)/T}}{e^{p(1,N_i)/T} + e^{p(0,N_i)/T}}$$
   Here, T is the *temperature coefficient* that controls the convergence of the Gibbs probability distribution. Increasing T makes the distribution close to being uniform, and decreasing T converges it to an extrema.
   (c) Sample $p(x_i|N_i)$ to get new x_i.
3. Terminate with answer x.

Figure 5: The workflow of implemented Gibbs Sampler algorithm

Here, we set a linear schedule for the temperature as $T = \frac{1}{g \times CR}$, where g is the current generation of MOA and CR is the *cooling rate* parameter. CR can be varied in order to control the convergence of MOA. For instance, setting CR higher will result in quick convergence of the conditional probabilities and therefore a quick convergence to a solution, i.e. gain in efficiency, but with less exploration of the search space. Conversely, smaller CR would result in slower convergence of the conditional probabilities and therefore slower convergence to a solution, i.e. more exploration of the search space, but with higher fitness evaluation. For the purpose of this paper we set CR to 0.5, since large number of different experiments preformed showed this value to be a good compromise between exploration and the exploitation of the search space for the problems tackled.

5 Experimental results

We test MOA on a number deceptive functions with multivariate dependency between variables. They are, deceptive function of order 3 (deceptive3) [6] and the trap function [24]. Both of these functions are widely used in the EDA literature as the benchmark to test the performance of different EDAs [18, 27, 34]. The *deceptive*3 function is defined as

$$deceptive3(x) = \sum_{i=1}^{\frac{n}{3}} f_{Gdec}(x_{3i-2} + x_{3i-1} + x_{3i}) \tag{9}$$

$$f_{Gdec}(u) = \begin{cases} 0.9 & for \quad u = 0 \\ 0.8 & for \quad u = 1 \\ 0.0 & for \quad u = 2 \\ 1.0 & for \quad u = 3 \end{cases}$$

Where, u is the number of ones in the input block of 3 bits.

Similarly, a Trap function of order k can be defined as

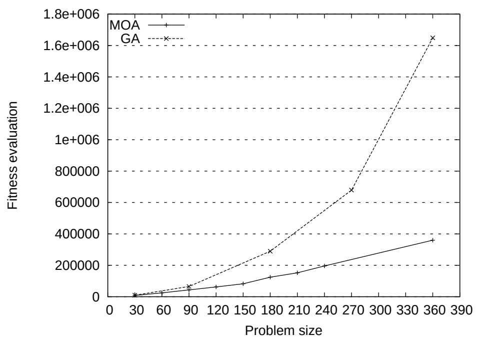
$$f_{trap,k}(x) = \sum_{i=1}^{n/k} trap_k(x_{b_i,1} + ... + x_{b_i,k}) \tag{10}$$

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

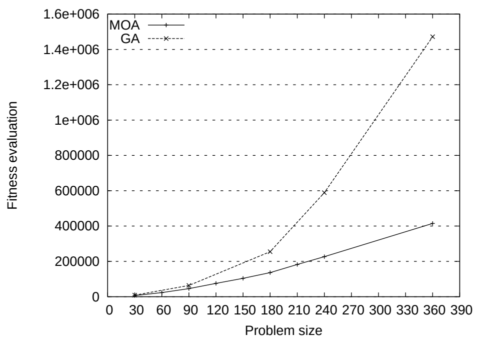
$$trap_k(u) = \begin{cases} f_{high}, & \text{if} \quad u = k \\ f_{low} - u\frac{f_{low}}{k-1}, & \text{otherwise} \end{cases}$$

Where, u is the number of ones in the input block of k bits, and f_{high} and f_{low} are parameters that control the distance between the local and global optima. .

In order to test the different aspects of MOA performance, we divide our experiments into six parts. In the first part, we compare the performance of MOA with that of other Bayesian network based EDAs and GAs on above functions. Next, we slightly modify the problem formulation and introduce a permutation to the ordering of bits in the problem, such that bits in the block are no more tightly linked to each other. This makes the problem very hard to solve for a blind crossover based GA. In contrast, we show that, as expected, this does not make any difference to the MOA performance. In the third part, we further increase the difficulty of the problem by introducing the overlapping dependency between blocks and test the performance of MOA on it. Introducing overlaps makes the problem very difficult, since one (or more) variable can be part of two (or more) blocks. In this case, even if the fitness maximising configuration for a block is found, it can be very easily disrupted due to the incorrect configuration of another block. In the fourth part, we show how we can easily incorporate problem specific knowledge about the interaction between variables in MOA. We do so by testing MOA on different versions of the above functions when prior knowledge about the problem is given as the undirected graph. In the fifth part, we show that the implemented structure learning algorithm in MOA can correctly identify the structure for above problems even when little (or no) knowledge about the problem structure is known. We also suggest other alternative structure learning algorithms that could improve the performance of MOA. Finally in the sixth part, we compare the performance of MOA with Gibbs sampling to that of MOA without Gibbs sampling and show that Gibbs sampling is one of the key components of MOA workflow.



(a) MOA vs GA on deceptive3 function



(b) MOA vs GA on trap5 function

Figure 6: Scalability graph comparing the performance of MOA and GA for both deceptive and trap function of size ranging from 30 bits to 360 bits

5.1 Comparison with other EAs

In this section, we compare the performance of MOA with the performance of its Bayesian network counter part, the BOA. Comparison is made with the BOA results reported [27]. BOA has been shown to significantly outperform GA in both of these functions. Therefore, we also find it interesting to compare the performance of MOA with GA.

For both deceptive3 and trap function, problem size, n, ranged from 30 to 360 bits. The order of interaction in trap function was set to 5. We call it trap5, which is the instance of the general trap function where $k = 5$, $f_{high} = 5$ and $f_{low} = 4$. For all experiments, population size (PS) was gradually increased until all of the 10 runs of the algorithm found the optimum solution.

Parameter setups for MOA were as follows: population size (PS) ranged from 1000 to 32000 for 30 to 360 bit deceptive3 problems and from 600 to 24000 for 30 to 360 bit trap5 problems. Truncation selection with selection size (SS) of 50% of the PS was used. In order to prevent quick diversity loss, the elitism parameter (EL) was set to the 50% of the PS, i.e., best half of the parent population was preserved in the next generation. Also, the number of maximum neighbours (MN) allowed to the structure of the Markov network in MOA was set to 2 for deceptive function and 4 for trap5 function. This is similar to the setup for BOA in [27], where maximum number of parents for each node in a Bayesian network was limited to 2 for deceptive function and 4 for trap5 function.

For GA, population size (PS) ranged from 600 to 18000 for 30 to 360 bit deceptive3 problems and from 600 to 20000 for 30 to 360 bit trap5 problems. Truncation selection with selection size (SS) of 50% of the PS was used. Similar to [27], onepoint crossover with crossover probability of 1 was used. The mutation probability was, however, set to to 0.0001, a lot smaller value than that set in []. Also, no elitism was used, i.e. parent population was completely replaced by the child population. We found that with such smaller mutation rate and complete replacement strategy, GA performance was significantly better than that reported in [27].

Table 1 shows the average ± the standard deviation of the fitness evaluations required by both

MOA and GA to find the optimum solution for deceptive3 over 10 runs. Similarly, Table 2 shows the same statistic for trap5 problem. Also, Figure 6(a) shows the scalability graph comparing the performance of both MOA and GA over different problem sizes for deceptive problem and Figure 6(b) shows the same for trap problem.

Table 1: The average ± standard deviation required by MOA and GA to find the optimum solution for deceptive3 function of size 30 to 360 over 10 runs

Size	MOA-Evaluations	GA-Evaluations
30	8100 ± 459	9300 ± 990
90	43340 ± 1063	65340 ± 3292
180	124490 ± 1866	289500 ± 10659
240	195750 ± 4500	679000 ± 22828
360	360000 ± 11314	1648800 ± 37326

Table 2: The average ± standard deviation required by MOA and GA to find the optimum solution for trap5 function of size 30 to 360 over 10 runs

Size	MOA-Evaluations	GA-Evaluations
30	6660 ± 276	8940 ± 1586
90	45900 ± 1049	63800 ± 9402
180	136420 ± 2157	255000 ± 12019
240	226500 ± 3000	588889 ± 27131
360	415000 ± 7071	1472000 ± 60992

Our results show that, for smaller problems, the performance of MOA and GA is comparable, however once the problem size start to get larger, MOA significantly outperforms GA in terms of number of fitness evaluation required to find the optimum solution. Also, the lower standard deviation for fitness evaluation in MOA suggests that, it is more predictable algorithm than GA.

Also, in comparison to BOA, MOA requires slightly less fitness evaluation. This can be observed by comparing the MOA results with that presented in [27] for BOA. As an example, it is shown in [27] that BOA requires in average around 160000 fitness evaluation to solve 180 bit deceptive function, while MOA only requires around 125000 fitness evaluation. Also, for trap function, BOA in average requires around 220000 fitness evaluations, while MOA only requires around 136000 fitness evolutions. [6]

5.2 Introducing permutation

The purpose of the experiments in this section is to show that different ordering of the bits in the solution does not affect the performance of the MOA. In other words, we show that MOA does not care whether the bits in the blocks are tightly located, closer to each other, or whether they are

[6]We note that in order to make the comparison fair, we compare the performance MOA with the version of BOA presented in [27] which also had a parameter, similar to MN, that restricted the maximum number of parents going to a node. Similar to the later version of BOA, improved structure learning algorithm is likely to remove this parameter from MOA workflow. Doing so remains the part of the future work.

loosely scattered all over the solution. For this purpose, we randomly permute the ordering of the bits in the solution for Trap5 problem. An example of original and modified dependency graph is shown in figure 7 (b) and figure 7 (c) respectively. This makes the bits in a single block further apart from each other. This obviously has a negative effect to the performance of the GA since the crossover operator do not take into account the dependency between variable in the solution and can easily disrupt a correct configuration of the block. We test both GA and MOA to the permuted trap5 function. The parameter setups were same as in previous section. As expected, GA was not able to find the solution, even for the very small problem size of 30 bits and with a very high population size of 20000. However, MOA was able to find the solution in similar fitness evaluation as with non-permuted trap5. The result is shown in figure 8.

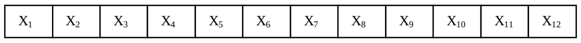
5.3 Introducing overlaps

In this section, we introduce overlap between the blocks in the solution. For example, overlap of order one means a variable in a block is common to the next block. Figure 7 (d) shows an example of the overlap of order one in trap function of order 4 (trap4). Notice that there is a cycle, i.e., the last block overlaps with the first block. These are difficult class of problems, since, not only the deceptiveness and the loose linkage of the variables in block is present, but also there are overlapping dependencies between these blocks. Here, not only finding the correct configuration of values in block is difficult, but once found, preserving them is also very difficult, since it can be easily disrupted by the incorrect configuration of the neighbouring blocks. Note that, with the introduction of overlaps, the number of blocks also increases in comparison to non overlapping problem of the same size. For example, there are 15 blocks of order 4 in 60 bit non-overlapping trap4 function. In contrast, there are 20 blocks of order 4 in order one overlapping trap4 problem.

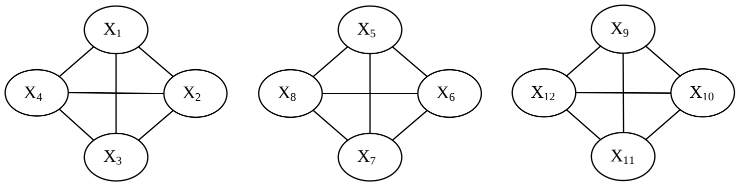
We test MOA on order one overlapping trap problems with $k = 4$ (trap4). The problem size ranged from 30 to 120 bits. As with previous case, GA was not able to find the solution even for the very small sized problem of 30 bits. The parameter setups for MOA were as follows: population size (PS) ranged from 600 to 6000 for 30 to 120 bit problems. Truncation selection with selection size (SS) of 50% of the PS was used. In order to prevent quick diversity loss, the elitism parameter (EL) was set to the 50% of the PS, i.e., best half of the parent population was preserved in the next generation. Also, the number of maximum neighbours (MN) allowed to the structure of the Markov network in MOA was set to 6, since (as can be seen from the Figure 7 (d)) there can be at most 6 neighbours for a variable.

The scalability graph showing the number of fitness evaluations required by MOA to find the solution for 30 to 120 bit permuted trap4 problem with overlap of order 1 is shown in Figure 9. We also plot the same statistic for the MOA on non-permuted trap4 function with overlap of order 1, in order to (again) show that ordering of bits in the solution does not make difference to the performance of MOA. Since, we could not find any literatures testing other EDAs on this function, no comparison could be made to assess the performance of MOA with respect to that of other EDAs [7].

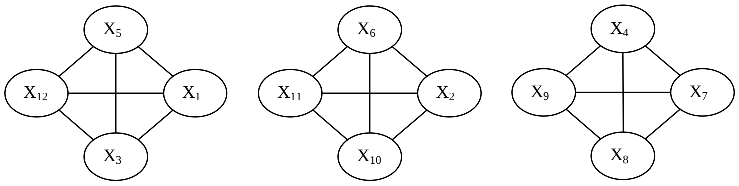
[7] We note that in [29], BOA has been tested on a multivariate function called random decomposable problems (rADPs). While, this function also has overlapping dependency, it does not consider deceptiveness and therefore has different properties. Testing MOA to rADPs remains one of the works for future



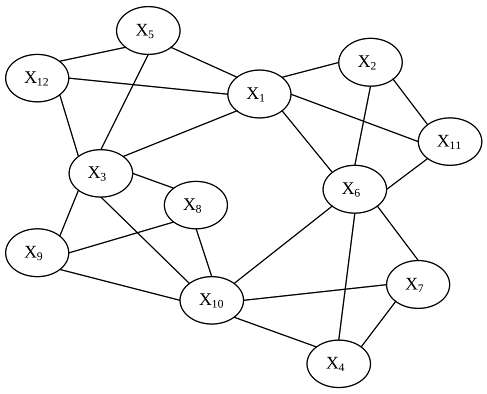
a. The ordering of the variables in the solution



b. The original dependency



c. A permuted dependency



d. An overlapping permuted dependency

Figure 7: An order 4 deceptive problem with different dependency structure

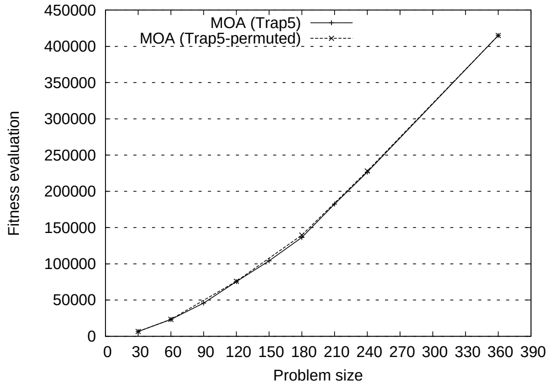


Figure 8: Number of fitness evaluations required by MOA on trap5 function of size ranging from 30 to 360 bits with both permuted ordering and non-permuted ordering of bits

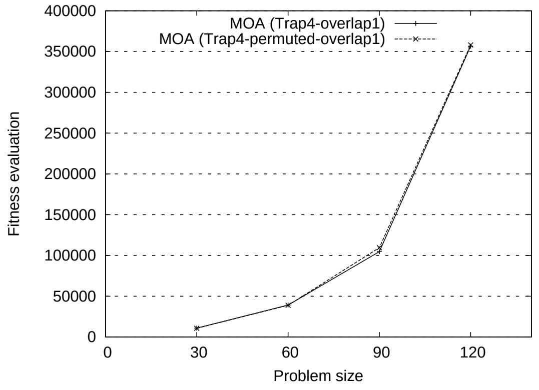


Figure 9: Number of fitness evaluations required by MOA on overlapping trap4 function of order 1 for size ranging from 30 to 120 bits with both permuted ordering and non-permuted ordering of bits

5.4 Incorporating prior information

In this part, we show how we can incorporate the prior information about the problem dependency in MOA and also analyse its effect to the performance of the algorithm. For this, we test MOA on different deceptive functions shown in Figure 7, by giving their dependency as a undirected graph. We note that with a Bayesian network based EDA, finding exact DAG structure for deceptive problems (in particular, to the one with cycles, Figure 7 (d)) can be difficult [3]. While with MOA, such dependency can be directly represented as a Markov network. Results comparing the performance of MOA with pre given structure with that of MOA that required to learn the structure from the data is shown in figure 10(a) for deceptive3 function, in figure 10(b) for trap5 function and in figure 10(c) for overlapping trap4 function.

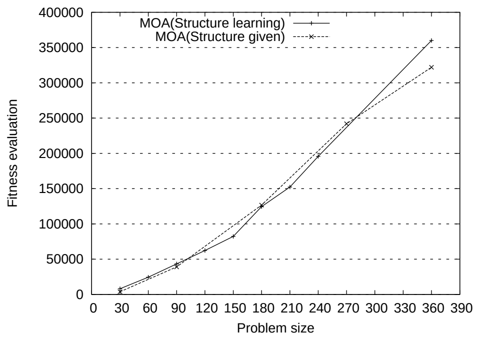
It can be seen that, for deceptive3 problem, there was no improvement in the performance, while for trap5 problem the performance improvement was significant. Again, for overlapping trap4 problem the improvement in performance was marginal. These results suggest that, giving the prior information about the dependency may improve the performance of the algorithm, however, this is not guaranteed and is highly problem dependent. These results are interesting and clearly require further work in order to get the explanation of these effects. These results also link us back to an open question in the EDA, that is whether it is necessary to have an exact problem structure in order to get a better performance?

5.5 Untuned vs tuned MOA

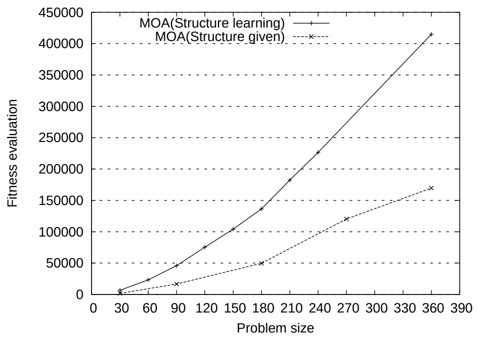
In all of the experiments presented so far, the maximum neighbour (MN) parameter for MOA was tuned in advance according to the problem structure. In this part, we show that MOA can solve the problem even without tuning the maximum neighbour (MN) parameter. For that, we perform two sets of experiments, each with different setup for the number of maximum neighbours (MN) allowed to the structure of the Markov network in MOA. For the first set of experiments, MN was set to 8 for both deceptive3 and trap5 functions. This emulates the situation when the structure of the network is completely unknown and is left to the algorithm to find, i.e., MN is un-tuned. For the second set of experiments, the MN was set to 2 for deceptive3 function and 4 for trap5 function (as with previous experiments). This emulates the situation when some experiments have been done to tune MN.

Parameter setups were as follows: for all experiments, population size (PS) was gradually increased until all of the 20 runs of the algorithm found the optimum solution. For the first set of experiments with un-tuned MN, i,e, when $MN = 8$, PS ranged from 1000 to 80000 for 30 to 180 bit deceptive3 problems and from 1000 to 38000 for 30 to 180 bit trap5 problems. Similarly, for the second set of experiments with tuned MN, i.e. with $MN = 2$ for deceptive3 and $MN = 4$ for trap5, PS ranged from 1000 to 32000 for 30 to 360 bit deceptive3 problems and 600 to 20000 for the trap5 problems of similar size. Truncation selection with selection size (SS) of 50% of the PS was used and elitism (EL) was set to the 50% of the PS, i.e., best half of the parent population was preserved in the next generation.

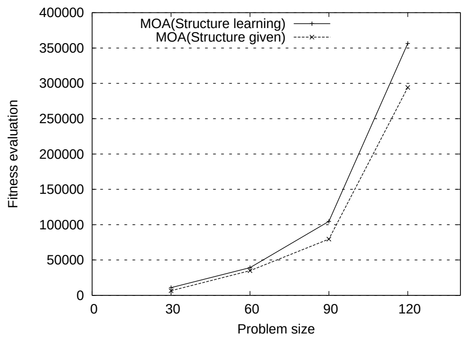
For each problem size, MOA was executed for 20 times and the number of fitness evaluations required to find the optimal solution was recorded. Table 3 shows, for the experiments with tuned MN, the average, the standard deviation and the maximum of the fitness evaluations required by MOA to find the optimum solution for both deceptive3 and trap5 functions of different sizes over 20 runs. Similarly, Table 4 shows same statistic for the experiments with un-tuned MN. Also,



(a) For deceptive3 function



(b) For trap5 function



(c) For overlapping trap4 function of order 1

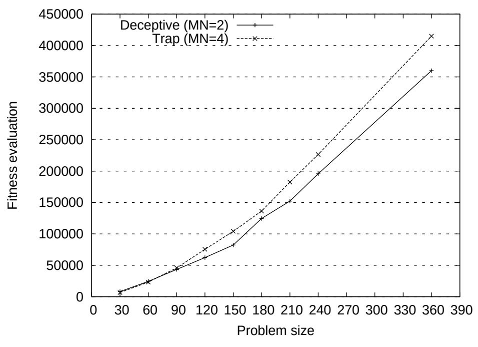
Figure 10: Scalability graph showing the number of fitness evaluations required by MOA when the structure of the problem is learnt from the population and when the structure of the problem is given as the prior information

Table 3: The average, the standard deviation and the maximum fitness evaluation required by MOA to find the optimum solution for both deceptive3 and trap5 function of size 30 to 360 over 20 runs with tuned MN, i.e., with $MN = 2$ for deceptive3 and $MN = 4$ for trap5

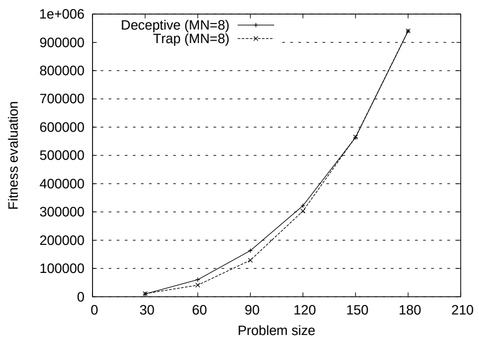
		deceptive3			trap5	
Size	Avg	Stdev	Max	Avg	Stdev	Max
30	8100	459.47	9000	6660	275.68	7200
60	24440	822.19	26000	23280	3187.75	29600
90	43340	1062.70	44000	45900	1048.81	48000
120	62250	2121.32	66000	75440	2611.17	80500
150	82333	1300.00	85800	104100	4701.06	117000
180	124490	1865.74	129800	136420	2157.06	140600
210	152250	3500.00	154000	182500	2886.75	185000
240	195750	4500.00	198000	226500	3000.00	228000
360	360000	11313.71	368000	415000	7071.07	420000

Table 4: The average, the standard deviation and the maximum fitness evaluation required by MOA to find the optimum solution for both deceptive3 and trap5 function of size 30 to 180 over 20 runs with un-tuned MN, i.e. with $MN = 8$

		deceptive3			trap5	
Size	Avg	Stdev	Max	Avg	Stdev	Max
30	9750	1379.41	13500	10650	411.64	11000
60	60250	7115.12	80000	40900	1728.84	43000
90	163200	4732.86	168000	129000	4281.74	135000
120	322000	9033.27	336000	303000	5019.96	312000
150	563333	15055.45	580000	565000	10488.09	580000
180	940000	28284.27	960000	940500	13435.03	950000



(a) With tuned MN (i.e. $MN = 2$ for deceptive3 and $MN = 4$ for trap5)



(b) With untuned MN (i.e. $MN = 8$)

Figure 11: Comparative performance of MOA on deceptive3 and trap5 functions of size ranging from 30 to 360

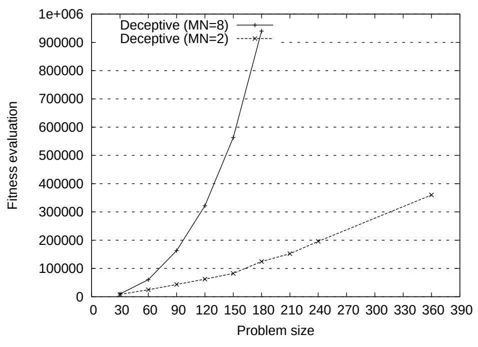
Figure 11(a) shows the scalability of MOA over different problem sizes for the set of experiments with tuned MN, and Figure 11(b) shows the same information for the experiments with un-tuned MN.

It can be noticed that, when MN is tuned (Figure 11(a)), the trap5 function requires slightly higher number of function evaluations in comparison to deceptive3 function. This is expected since the order of interaction in trap5 is higher than that in deceptive3 function, taking longer to find the solution. Also, when MN is not tuned (Figure 11(b)), the deceptive3 function, at the beginning, required slightly higher number of function evaluations. This is due to the larger population required by the structure learning algorithm to correctly discard 6 neighbours out of allowed 8. On the other hand, for trap5 function, only 4 out of 8 allowed neighbours had to be discarded. However, as the problem size grows, the task of tackling the higher order interaction in trap5 outweighs the structure learning task in deceptive3, resulting in almost similar fitness evaluation for both problems.

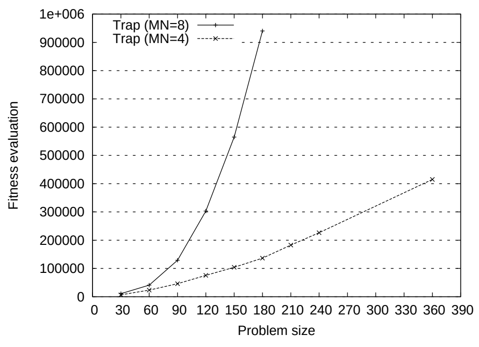
Also, in Figure 12(a), we compare the scalability of MOA on deceptive3 function with both un-tuned and tuned MN, i.e. with $MN = 8$ and $MN = 2$. Similarly, In Figure 12(b), we compare the same for the trap5 function, i.e. with both $MN = 8$ and $MN = 4$. These comparisons show that correctly tuning MN can result in a significant performance improvement. However, we again note that this issue of tuning is specific to the structure learning algorithm used in MOA and may be resolved by using alternative structure learning algorithms.

5.6 Gibbs sampling vs. temperature less sampling

In order to highlight the importance of Gibbs sampling in MOA, in this part, we compare the performance of MOA with Gibbs sampling with that of MOA with *temperature less sampling.* The workflow of temperature less sampling is the same as the one shown in Figure 5, except for



(a) Untuned vs tuned MN on deceptive3 function



(b) Untuned vs tuned MN on trap5 function

Figure 12: Performance of MOA with both un-tuned and tuned maximum neighbour size MN

the conditional probability estimation part, which is done as

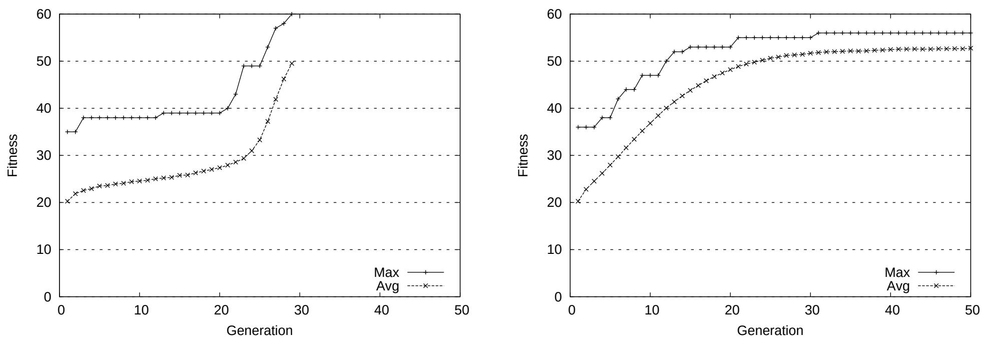
$$p(x_i|N_i) = \frac{p(x_i, N_i)}{\sum_{x_i'} p(x_i', N_i)}$$

We found that temperature less sampling was not able to find the solution for higher sized problems within a reasonable population size. Figures 13(a) shows a typical example of how maximum and average fitness in the population progress in each generation during a typical run of MOA with Gibbs sampling for 60 bit trap function. Similarly, Figure 13(b) shows the same for the temperature less sampling. The parameter set up were as follows, PS was set to 2000, SS and EL were set to 50% of PS, and MN was set to 4. The algorithm was stopped once the optimum was found, or 50 generations were passed.

We can notice that the temperature less sampling could not find the optimal solution in 50 generations and converged to some near optimal solution, where as Gibbs sampling finds the solution in around 30 generations. We can also see that the curves in Gibbs sampling is of sigmoid shape in comparison to (near) linear shape in temperature less sampling. This suggests that temperature less sampling narrows the search space as generation progress. In contrast, Gibbs sampling first (thoroughly) explores the search space and then converges to a solution. Cooling rate (CR) parameter influences the form of the curve in the Gibbs sampling. The lower the CR, the higher the exploration and more sigmoid the curve is.

6 Theoretical analysis of Gibbs sampler

Gibbs sampler plays an important role in the way the information contained in the Markov model is used for exploring the promising areas of search. In some cases, the expected waiting time of the Gibbs Sampler for moves from states of high probability to states of lower probability can be very long. It is important to theoretically analyze how the behaviour of the Gibbs sampler is related to some particular features of the problem.



(a) Gibbs sampling with $CR = 0.5$ (b) Temperature less sampling

Figure 13: An example of how maximum and average fitness in the population progress in each generation during a typical run of MOA on 60 bit trap function

In this section we show some interesting theoretical properties of the Gibbs sampler by means of an example. We compute the Boltzmann distribution associated to the deceptive3 function (Equation 9) and analyze the behaviour of Gibbs sampling for this distribution. We use the smallest possible set of variables $X = X_1, X_2, X_3$, and define the joint probability distribution.

$$p(\mathbf{x}) = \frac{e^{f_{deceptive}}(\mathbf{x})/T}{\sum_{x'} e^{f_{deceptive}(\mathbf{x}')}/T} \tag{11}$$

For $T = 1$, the probabilities computed are $p(0, 0, 0) = 0.16558$, $p(0, 0, 1) = 0.14982$, $p(0, 1, 1) = 0.06732$ and $p(1, 1, 1) = 0.18299$. Since the function values depend on the unitation, we also have: $p(0, 0, 1) = p(0, 1, 0) = p(1, 0, 0)$, $p(0, 1, 1) = p(1, 1, 0) = p(1, 0, 1)$ and the conditional probabilities of each variable given each neighbours will be the same:

$$\begin{array}{cccc} 0.52498 & 0.68997 & 0.68997 & 0.26894 \\ 0.47502 & 0.31003 & 0.31003 & 0.73106 \end{array} \tag{12}$$

From the conditional probabilities, the probabilities $P(x^i, x^j)$ that the Markov chain changes from configuration x^i to x^j in a single step can be computed. These probabilities are represented in the transition matrix whose elements i, j correspond to $P(x^i, x^j)$. Equation 13 represents the transition matrix corresponding to the probability distribution shown in Equation 11.

$$T_m = \begin{pmatrix} 0.52498 & 0.15834 & 0.15834 & 0 & 0.15834 & 0 & 0 & \\ 0.17499 & 0.61832 & 0 & 0.10334 & 0 & 0.10334 & 0 & 0 \\ 0.17499 & 0 & 0.61832 & 0.10334 & 0 & 0 & 0.10334 & 0 \\ 0 & 0.22999 & 0.22999 & 0.29633 & 0 & 0 & 0 & 0.24369 \\ 0.17499 & 0 & 0 & 0 & 0.61832 & 0.10334 & 0.10334 & 0 \\ 0 & 0.22999 & 0 & 0 & 0.22999 & 0.29633 & 0 & 0.24369 \\ 0 & 0 & 0.22999 & 0 & 0.22999 & 0 & 0.29633 & 0.24369 \\ 0 & 0 & 0 & 0.089647 & 0 & 0.089647 & 0.089647 & 0.73106 \end{pmatrix} \tag{13}$$

By inspecting the matrix it is possible to determine which configurations can be reached in a single step from each other configuration. For instance, starting from configuration $(0, 0, 0)$ it is not possible to reach any other configuration with more than one component equal 1. The transition matrix P will determine a number of important properties of the Markov chain. In particular, the vector probabilities for every state at step t $(\pi_t(x))$ can be computed as:

$$\pi_t(x) = \pi_0(x)P^t \tag{14}$$

where $\pi_0(x)$ represents the initial probabilities for all the states.

The transition matrix will determine whether the Markov chain is irreducible (all states communicate with each other after a finite number of states), aperiodic (the number of steps needed to move between two different configurations is not required to be multiple of an integer) and whether it will converge to a unique stationary distribution.

Figure 14 shows the evolution of the probabilities at each step of the Markov chain with transition probabilities represented by Equation 13 and started from a uniform probability $\pi_0(x)$. At each step, the probabilities have been computed using Equation 14.

It can be seen in Figure 14 that after a few number of steps the Markov chain converges to the target distribution. Also interesting is to appreciate that at the initial steps the initial probabilities can be far from the final ones.

When inserted within EDAs, the behaviour of the Gibbs sampling will depend on the initial probabilities (whether they are uniform, or conveniently chosen), the transition matrix, which depends as well on the structure of the neighbourhood and the probabilities learned from the data) and on the number of steps allowed to Gibbs sampling.

7 Conclusion

In this paper, we have proposed MOA, a local Markov property based EDA, as a promising member of the Markov network based EDA. The proposed algorithm has simpler workflow and is easier to implement than other global Markov property based EDAs. It incorporates features that have been independently employed in previous implementations of EDAs based on Markov models, but have not been used together. The resulting algorithm is qualitatively different to its predecessors. It does structural learning of the probabilistic model from the data but it can also take advantage of a priori structural information in a straightforward way.

Following are the key contributions of this paper.

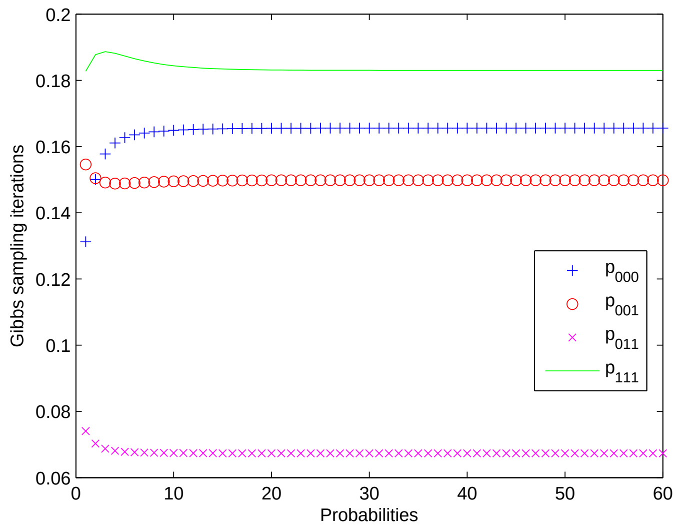


Figure 14: Configuration probabilities at different steps of the Markov chain started from a uniform probability distribution.

- Use of local conditional probability distribution in a Markov network EDA as an alternative to the joint probability distribution. This avoids complex approximation involving cliques and the corresponding potential functions, resulting in simpler and easy to implement Markov network based EDA.
- Use of Gibbs sampler as a method to sample from the undirected structure. This allows explicit control over the convergence of the probability distribution and also allows MOA to balance the exploration and exploitation of the search space.

As stated earlier, it is important to distinguish the difference between MOA and its other Markov network based counterparts, DEUM and MN-EDA. In particular, DEUM defines clique potential functions to encapsulate interaction between variable and builds a model of fitness function to approximate the joint probability distribution of a Markov network. Similarly, MN-EDA also estimates joint probability distribution by means of Kikuchi approximation. In contrast, MOA estimates conditional probabilities defined by the neighbourhood structure of the Markov network and does not estimate joint probability distribution. These conditional probabilities are then sampled using Gibbs sampler. The paper also shows both empirically and theoretically, the effect of the Gibbs sampler has on the workflow of the algorithm.

Interaction between variables in many real world problems can be naturally represented as an undirected graph. Well known examples of such problems include Ising spin glasses and SAT. Structure of such problems can be readily incorporated in MOA without requiring to have a structure learning step. This is an important property of MOA that distinguishes it from other Bayesian network based EDAs. We also notice that, the workflow of MOA is comparatively more similar to that of BN based EDAs than other global Markov network based EDAs, since it also estimates conditional probabilities for each variables by means of frequency counting.

A range of experiment was conducted in order to test the different aspects of MOA performance, which confirmed that MOA can effectively solve problems with complex interaction between variables. The results also showed that its performance is comparable to that of other Bayesian network based EDAs.

Further research should be done in order to improve the performance of MOA, particularly, on the implementation of a more efficient structure learning algorithm, and also on the use of other more efficient versions of the Gibbs sampler algorithm. These works are underway and interesting results are expected in near future.

References

[1] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In *Proceedings of the 14th International Conference on Machine Learning*, pages 30–38. Morgan Kaufmann.

[2] J. Besag. Spatial interactions and the statistical analysis of lattice systems (with discussions). *Journal of the Royal Statistical Society*, 36:192–236, 1974.

[3] C. Echegoyen, J. A. Lozano, R. Santana, and P. Larrañaga. Exact Bayesian network learning in estimation of distribution algorithms. In *Proceedings of the 2007 Congress on Evolutionary Computation CEC-2007*, pages 1051–1058. IEEE Press, 2007.

[4] R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 151–173, Havana, Cuba, 1999.

[5] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, pages 564–584. Kaufmann, Los Altos, CA., 1987.

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

[7] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. *Unpublished*, 1971.

[8] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 149–163. North-Holland, Amsterdam, 1988.

[9] M. I. Jordan, editor. *Learning in Graphical Models.* NATO Science Series. Kluwer Academic Publishers, Dordrecht, 1998.

[10] R. Kikuchi. A Theory of Cooperative Phenomena. *Physical Review*, 81:988–1003, Mar. 1951.

[11] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Combinatorial optimization by learning and simulation of Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 343–352, 2000. Stanford.

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

[13] S. L. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[14] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50:157–224, 1988.

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

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

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

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

[19] 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.

[20] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, 2002.

[21] 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.

[22] A. Ochoa, M. R. Soto, R. Santana, J. Madera, and N. Jorge. The factorized distribution algorithm and the junction tree: A learning perspective. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 368–377, Havana, Cuba, March 1999.

[23] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufman Publishers, Palo Alto, CA, 1988.

[24] 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.

[25] M. Pelikan and D. E. Goldberg. Hierarchical problem solving by the Bayesian optimization algorithm. IlliGAL Report No. 2000002, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 2000.

[26] 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.

[27] M. Pelikan, D. E. Goldberg, and E. Cantú-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.

[28] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[29] M. Pelikan, K. Sastry, M. V. Butz, and D. E. Goldberg. Hierarchical BOA on random decomposable problems. IlliGAL Report No. 2006002, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, January 2006.

[30] R. Santana. A Markov network based factorized distribution algorithm for optimization. In *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)*, volume 2837, pages 337–348, Dubrovnik, Croatia, 2003. Springer-Verlag.

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

[32] R. Santana, P. Larrañaga, and J. A. Lozano. Mixtures of Kikuchi approximations. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Proceedings of the 17th European Conference on Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Artificial Intelligence*, pages 365–376, 2006.

[33] S. Shakya. *DEUM: A Framework for an Estimation of Distribution Algorithm based on Markov Random Fields*. PhD thesis, The Robert Gordon University, Aberdeen, UK, April 2006.

[34] S. Shakya and J. McCall. Optimisation by Estimation of Distribution with DEUM framework based on Markov Random Fields. *International Journal of Automation and Computing*, 4:262–272, 2007.

[35] S. Shakya, J. McCall, and D. 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, volume 109 of Frontiers in Artificial Intelligence and Applications*, pages 15–25, Valencia, Spain, August 2004. IOS press.

[36] 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 (GECCO2005)*, pages 727–734, Washington, D.C., USA, 2005. ACM.

[37] S. Shakya, J. McCall, and D. Brown. Solving the Ising spin glass problem using a bivariate EDA based on Markov Random Fields. In *proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2006)*, pages 3250–3257, Vancouver, Canada, 2006. IEEE press.

[38] S. Shakya and R. Santana. An EDA based on local Markov property and Gibbs sampling. In *proceedings of Genetic and Evolutionary Computation COnference (GECCO2008)*, Atlanta, Georgia, USA, 2008. ACM.