Estimation of Distribution Algorithm using Factor Graph and Markov Blanket Canonical Factorization

B. Hoda Helmi
Iran University of Science and Technology
Tehran, Iran 13114-16846
helmi@iust.ac.ir

Adel T. Rahmani
Iran University of Science and Technology
Tehran, Iran 13114-16846
rahmani@iust.ac.ir

ABSTRACT

Finding a good model and efficiently estimating the distribution is still an open challenge in estimation of distribution algorithms (EDAs). Factorization encoded by models in most of the EDAs are constrained. However for optimization of many real-world problems, finding the model capable of representing complex interactions without much computational complexity overhead is the key challenge. On the other hand factor graph which is the most natural graphical model for representing additively decomposable functions is rarely employed in EDAs. In this paper we introduce Factor Graph based EDA (FGEDA) which learns factor graph as the model and estimate the probability distribution represented by the learned factor graph using Markov blanket canonical factorization. The class of factorization that is employed for approximation of distribution in FGEDA is expanded relative to famous EDAs. We have used matrix factorization for learning the factor graph of the problem based on the pairwise mutual information between pair of variables. Gibbs sampling and BB-wise crossover are used to generate new samples. Empirical evaluation as well as theoretical analysis of the approach show the efficiency and power of FGEDA in the optimization of functions with complex interactions. It is showed experimentally that FGEDA outperform other well-known EDAs.
GECCO track: Estimation of Distribution Algorithm.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithm, Design, Experimentation, Performance

Keywords

Optimization problems, Estimation of distribution algorithms, Factor graph, Linkage learning, Mutual information, Matrix factorization

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598266 .

1. INTRODUCTION

Estimation of Distribution Algorithms (EDA) are algorithms that learn the variable's dependencies' structure and overcome the difficulty of optimization for problems with complex variable interactions. EDAs estimate the joint distribution of the multinomial data in order to generate new solutions. Encoding the joint distribution of the data is usually done using a graphical model. Bayesian networks [1] and Markov models [2] are widely used as the model in the model building process of EDAs, but factor graphs which are the ideal graphical model to encode the distribution of the data are rarely used possibly because there has been no efficient learning algorithm to learn this kind of graphical models.

In this paper, we introduce an efficient factor graph learning algorithm. To learn the factor graph we apply nonnegative matrix factorization on mutual information matrix. After learning the factor graph, the joint probability distribution which is encoded by factor graph is approximated. New potential solutions are produced using Gibbs sampling based on the factor nodes and their Markov blankets. The process of model building and creating new samples are repeated until the optimum solution is found.

Utilizing the factor graph has some advantages over other models. First and most important of all, this graphical model explicitly shows the structure of the dependencies between variables of the problem. This human-readable information can easily be used by experts to analyze the problem and the results and the revealed structure of the problem can be used for analyzing similar problems. Second, knowing the linkage groups we can use this information to employ BB-wise genetic operators besides re-sampling the new potential solutions via Gibbs sampling which is a computationally expensive task.

The outline of this paper is as follows: In the next section, some related works are reviewed. In section 3 some related background information on factor graph and symmetric nonnegative matrix factorization (SNMF) are reviewd. Section 4 introduces the proposed factor graph based estimation of distribution algorithm and explaines the model learning and sampling method used in the algorithm. In section 5, the behavior of the algorithm is analyzed. Section 6 presents the complexity of the algorithm. Section 8 presents the experimental results of the algorithm. To show the performance and efficiency of the proposed approach, it is tested on some benchmark problems and its effectiveness is discussed. Finally in section 9, some conclusion remarks, highlights of the algorithm and future directions are presented.

2. RELATED WORKS

Estimation of distribution algorithms (EDAs) [3] or Probabilistic Model Building Genetic Algorithms (PMBGAs) are algorithms that try to learn the structure of the problem in order to effectively explore the search space to find the optimum. Several EDAs are introduced [4], [1], [5], [6]. Most of these approaches are population based and generational search processes. Among all the methods used to learn and encode the distribution, graphical models like Bayesian networks and Markov networks are employed and investigated in many studies. There are pros and cons for using each of these graphical models in EDAs. On one hand learning Bayesian network from data is a computationally costly task, on the other hand sampling from undirected graphical models is difficult and requires Gibbs sampling which is computationally expensive.

Factor graphs are another type of undirected graphical models that subsume both Bayesian networks and Markov networks, in that every Bayesian network or Markov network can be converted to its corresponding factor graph of the same size [7]. These graphical models are the most natural graphical representations for the structure of Additively Decomposable Functions (ADF) [7]. Learning of factor graphs for ADF is more representative than learning of Bayesian networks because they match the ADF structure better [8]. Factor graph clearly shows the underlying structure of the problem and is readable for experts. This is an important property for the expert, when optimizing the problems with unknown dependency structure. But learning the structure of these kinds of graphical models and sampling using these distributions is difficult.

There are EDAs that utilize factor graphs to encode the distribution of the problem. Mühlenbein in [8] proposed using factor graphs as the model to represent the distribution of the problem in EDAs. In [9], an EDA approach is introduced in which the structure of the factor graph is learned using the Chi-square independence test. The Kikuchi approximation of the distribution is then used to approximate the distribution and finally Gibbs sampling is applied to sample new potential solutions. In [10], to search for the optimum, loopy belief propagation (LBF) is used to find the most probable configuration of the distribution. They first learn a Bayesian network from a population of potential solutions and then convert it to its corresponding factor graph to apply the LBF, and find the most probable configuration of the distribution.

Here, we introduce an EDA algorithm which directly learns factor graph from pair-wise statistics calculated from selected population and then samples new solutions based on the model learnt.

3. BACKGROUND

3.1 Factor Graph

A factor graph is a bipartite graph that describes the way a function of many variables factors into a product of several functions. For example, assume $g(x1, x2, x3, x4, x5)$ be a function with five variables. This function can be factored into a product of three functions as follows:

$$g(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2, x_4) f_2(x_2, x_3) f_3(x_4, x_5) \quad (1)$$

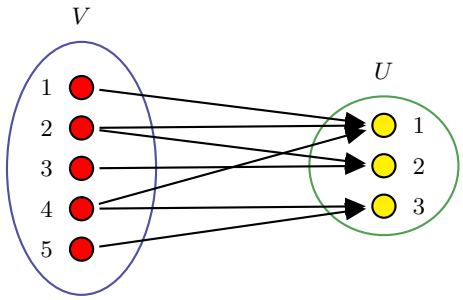

Figure 1: A factor graph that shows the factorization of equation (1)

The corresponding factor graph is illustrated in figure 1. The factor graph is consisted of variable nodes, factor nodes and the edges between these two kinds of nodes. For each variable of the function, there is a corresponding variable node in the factor graph. Each factor node corresponds to a factor. A variable node is connected to a factor node if and only if the variable is an argument of the corresponding function. In order to find the factor graph, we use a matrix factorization technique which has been mathematically proven to converge in polynomial time. To do so, we employ pair-wise variable dependencies to partition the variables probabilistically. Based on the concept of random walks on graphs, in our approach, the variable dependencies are modelled as empirical transitions generated by a mixture of latent factors.

3.2 Symmetric Non-Negative Matrix Factorization (SNMF)

Given a nonnegative matrix X, non-negative matrix factorization (NMF) is finding a lower-rank matrix approximation. Rank of a matrix is the number of linearly independent rows or columns of the matrix. So, NMF is formalized as:

$$X \approx CG^T \quad (2)$$

that can be achieved by calculating,

$$min||X - CG^T||, \quad (3)$$

where $||.||$ denotes a distance metric like divergence distance and $X \in \Re_+^{m*n}$, $C \in \Re_+^{m*k}$, $G \in \Re_+^{n*k}$.

The NMF can be applied to a similarity matrix. In this case the matrix factorization is a symmetric non-negative matrix factorization problem with the formulation:

$$\min_{H \geq 0} ||A - HH^T||_F^2 \quad (4)$$

where A is the similarity matrix of size $n \times n$ and H is a non-negative matrix of size $n \times m$ and the $||.||_F$ denotes the Forbenius norm. The largest entry in the i^{th} row of H, indicates the factor/cluster of the i^{th} item in A due to the non-negativity of H.

The symmetric non-negative matrix factorization is developed specially for soft clustering in data sets with non-linear clusters. There are algorithms for symmetric non-negative matrix factorization, which guarantee to produce stationary point solutions. The uniqueness of SNMF in terms of factorization quality, the independence on the eigenspace of A, and the sensitiveness to the fluctuation in the similarity matrix A, for soft clustering application is shown in [11].

4. FACTOR GRAPH BASED ESTIMATION OF DISTRIBUTION ALGORITHM

Variable dependencies can be encoded by an undirected graph, where vertices denote variables and edges' weights represent variable dependencies. In the proposed approach, this graph called mutual information matrix (MIM) is first constructed from a selected population of potential solutions using mutual information between every pair of variables. MIM is then factorized into a bipartite graph which represents the factor graph of the problem. So in the proposed approach, we are converting pair-wise dependencies to higher-order dependencies. After the factor graph is constructed new potential solutions are sampled using Gibbs sampling [12] and uniform BB-wise crossover [13].

The approach, as will be explained here, is capable of learning disjoint and overlapped linkage groups even for problems with linkage groups of varying size. The remainder of this section describes details of the proposed approach. First MIM is constructed by calculating MI between every pair of variables, then to accelerate the process of matrix factorization, some of the too small elements of MIM are pruned using a simple threshold. Based on our assumption of $k << n$, the threshold is set to μ, where μ is the mean of the elements in MIM. It should be noted that this pruning is by no means necessary for the algorithm.

Factorizing the MIM into a factor graph

Let $K(V,U,F)$ be the bipartite graph, where $V=\{v_i\}_{i=1}^{n}$ is a collection of variables and $U=\{u_p\}_{p=1}^{m}$ is a set of factor nodes (V and U are disjoint sets) and F contains all the edges connecting V and U. Let $B=\{b_{ip}\}$ denote the $n*m$ adjacency matrix with $b_{ip}\geq 0$ being the weight for edge $[v_i,u_p]$. To factorize the MIM and finding the corresponding bipartite graph, we use the similarity relation between v_i and v_j in the bipartite graph K presented in [14].

$$mim_{ij}=\sum_{p=1}^{m}\frac{b_{ip}b_{jp}}{\lambda_p}=(B\Lambda^{-1}B^T)_{ij},\quad \Lambda=diag(\lambda_1,...,\lambda_m) \tag{5}$$

Where $\lambda_p=\sum_{i=1}^{n}b_{ip}$ is the degree of vertex $u_p\in U$.

The above equation can be analyzed based on the random walks on graphs. e_{ij} is proportional to the stationary probability of transition between v_i and v_j, $p(v_i,v_j)$. All the paths between vertices in V, must go through vertices in U in the factor graph, therefore:

$$\begin{aligned} p(v_i,v_j)&=p(v_i)p(v_j|v_i)\\ &=d_i\sum_p p(u_p|v_i)p(v_j|u_p)\\ &=\sum_p\frac{p(v_i,u_p)p(u_p,v_j)}{\lambda_p}, \end{aligned} \tag{6}$$

where $d_i=p(v_i)$ is the degree of v_i. When $b_{ip}=p(v_i,u_p)$, equation (5) and equation (6) are the same. $p(u_p|v_i)=\frac{b_{ip}}{d_i}$ is the conditional probability of transitions from v_i to u_p and indicates how likely variable i belongs to factor nodes p.

Based on equation (5), the bipartite graph can be approximated by minimizing the $distance(MIM, B\Lambda^{-1}B^T)$. To make the problem easy we use $H=B\Lambda^{-1/2}$, then we have

$$\min_{H\in\Re_+^{n\times m}} distance(MIM,HH^T),\quad s.t.\ h_{ip}\geq 1, \tag{7}$$

This problem is a symmetric non-negative matrix factorization (SNMF) [15]. There are different numerical methods to find the local minima of this problem. Here we use the gradient descent method to minimize the divergence distance between the two adjacency matrices (equation (8)).

$$DD(X,Y)=\sum_{ij}(x_{ij}log\frac{x_{ij}}{y_{ij}}-x_{ij}+y_{ij}) \tag{8}$$

Theorem 4.1 *The distance is non-increasing under the following update rule*

$$\hat{h_{ip}}=\frac{h_{ip}}{\sum_j h_{jp}}\sum_j\frac{mim_{ij}}{(HH^T)_{ij}}h_{jp}. \tag{9}$$

The distance is invariant under the update rule if and only if H is at stationary point of the distance [16].

After H is calculated, B can be calculated using $B=H\Lambda^{1/2}$ and $\Lambda=diag(\lambda_1,...,\lambda_m)$ where $\lambda_p=\sum_{i=1}^{n}h_{ip}$.

Proof of the convergence of the algorithm is available in [16]. We later use λ_p as the degree for factor node p. In order to provide an algorithmic scheme look at the following pseudo-code:

Algorithm 1 Learning the Factor Graph via SNMF

Input: A non-empty $|V|\times|V|$ matrix MIM. $maxFN=n$, $MN=k+\epsilon$

Output: factor graph

Initialize: Start with a $n\times m$ matrix H, where $m=maxFN$ and $H_{i\in V,j\in U}\in[0,1]$ is randomly set.

repeat

Update H based on equation (9).

until H is invariant.

Construct $\Lambda=diag(\lambda_1,...,\lambda_m)$ using $\lambda_p=\sum_{i=1}^{n}h_{ip}$.

Construct factor nodes. ($\forall i\in rows(H)$ create a factor node FN_i. Assign the MN largest rows of the $H[i]$ to the FN_i). Keep unique FNs as the set of factor nodes.

Parameter $maxFN$ is used to determine the dimensions of matrix H. For all the experiments this parameter is set to n which is number of variables in the problem. Parameter MN is the maximum allowed interactions between variables. This parameter is used to avoid an overly complex network. We do not set MN equal to the true value according to the problem structure. For each problem we add $\epsilon=3$ to what is the true value of MN which is k. It should be noted that almost all the other methods also use such parameter to bound the size of the building blocks.

Generating new solutions

In this step, after learning the factor graphs and having the factors, the optimum value of the variables should be found. To do so, in order to be able to use Gibbs sampling we should approximate the distribution that is encoded in Factor Graph. We use the distribution approximation approach based on Markov blanket canonical factorization introduced in [17].

In [17] a new parameterization approach based on canonical parameterization and Markov blanket is presented in which every canonical factor can be calculated by local probabilities only. The Markov blanket canonical parameterization is defined relative to an arbitrary set of default assignments $\bar{\mathbf{x}} = (\bar{x}_1, ..., \bar{x}_n)$. Let $\mathbf{D} = \langle X_{i_1}, ..., X_{i_{|\mathbf{D}|}} \rangle$ be any subset of variables, and $\mathbf{d} = \langle x_{i_1}, ..., x_{i_{|\mathbf{D}|}} \rangle$ be any assignment to $\mathbf{D}$. For any $\mathbf{U} \subseteq \mathbf{D}$, $\emptyset\check{\mathrm{N}}\sigma_{\mathbf{U}:\mathbf{D}}[\mathbf{d}]$ is defined to be the restriction of the full instantiation $\sigma_{\mathbf{U}}[\mathbf{d}]$ of all variables in $\mathcal{X}$ to the corresponding instantiation of the subset $\mathbf{D}$. Let $\mathbf{D} \subseteq \mathcal{X}$ and $\mathbf{Y} \subseteq \mathcal{X} - \mathcal{D}$ then the Markov blanket canonical factor $f^*_{\mathbf{D}|\mathbf{Y}}$ is defined as equation (10).

$$f^*_{\mathbf{D}|\mathbf{Y}}(\mathbf{d}) = exp(\sum_{\mathbf{U} \subseteq \mathbf{D}} (-1)^{|\mathbf{D}-\mathbf{U}|} \log P(\sigma_{\mathbf{U}:\mathbf{D}}[\mathbf{d}]|\mathbf{Y} = \bar{\mathbf{y}})) \tag{10}$$

For positive Gibbs distribution and factor nodes with scopes $\{C_j\}_{j=1}^{J}$ (scopes of a factor node is the set of all the variables connected to it), and $\{C_j^*\}$ defined as $\{C_j^*\}_{j=1}^{J^*} = \cup_{j=1}^{J} 2^{C_j} - 0$, for any $\mathbf{D} \subseteq \mathcal{X}$

$$f^*_D = f^*_{\mathbf{D}|\mathcal{X}-\mathbf{D}} = f^*_{\mathbf{D}|MB(\mathbf{D})} \tag{11}$$

And

$$\begin{aligned} P(\mathbf{x}) &= P(\bar{\mathbf{x}}) \prod_{j=1}^{j^*} f^*_{C_j^*|(X)-C_j^*}(c_j^*) \\ &= P(\bar{\mathbf{x}}) \prod_{j=1}^{j^*} f^*_{C_j^*|MB(C_j^*)}(c_j^*), \end{aligned} \tag{12}$$

where c_j^* is the instantiation of C_j^* consistent with $\mathbf{x}$.

So according to equation (12), the joint probability distribution of a factor graph is approximated using probabilities over factor nodes and their Markov blankets only.

Now we can create new samples by performing Gibbs sampling. To create a new sample, we start with an initial state $x^0 = \{x_1^0, ..., x_n^0\}$. Based on the transition rule in (13), at each time t a variable i is to be updated. In one cycle of the Gibbs sampling all the n variables are updated.

$$\mathbf{X}^{t+1} = \begin{cases} x_j^{t+1} = x_j^t & \text{for } j \neq i \\ x_j^{t+1} \approx P(x_i^t | MB(x_i^t)) & \text{for } j = i \end{cases} \tag{13}$$

To generate L new solutions, Gibbs sampling should be applied L times. Instead of starting from random initial state, we select a high fitness solution from the population and repeat the Gibbs sampling C_n times.

Since we have learned the linkage groups in our model building phase, we can also perform BB-wise crossover and/or we can perform a search on each building block to find the optimum. In order to get advantage of both methods, we first generate $N/2$ samples by Gibbs sampling and then the other $N/2$ samples are generated by BB-wise crossover.

5. ALGORITHM ANALYSIS

In this section, we look closely at the SNMF approach by looking at the examples learned by this approach. In subsection 5.1, we see examples of SNMF performance for learning linkage groups in decomposable problems in the context of genetic algorithm and linkage learning. We analyze the matrix H with an example through different generations. In the second subsection, algorithm 1 is used to find the linkage groups in problems with overlapping building blocks to see if the approach is capable of finding the building blocks for additively decomposable problems with overlapping BBs or not. The next subsection is dedicated to the analysis of the effect of the $maxFN$ parameter used in algorithm 1 on the performance of the algorithm.

5.1 SNMF for learning the linkage groups in problems with non-overlapping BBs

In figure 2, the $MIM_{n\times n}$ ($n = 50$) matrix of mutual information, the matrix $H_{n\times maxFN}$ (SNMF resultant matrix, $maxFN = 50$) and linkage groups matrix for concatenated Trap-5 problem with 50 variables in different generations is depicted (concatenated trap function with tight linkage groups is used for the better visualization, algorithm is not sensitive to the place of variables in the string). The population size is set to 2600 and the $maxFN$ is set to n.

The rows show the information of the generations 1, 4 and 10 from top to bottom. The matrix H (second column in the figure), is an $n \times maxFN$ matrix that shows the value with which each variable (rows of the matrix) is related to each factor node (columns of the matrix). The intensity of each cell of the both matrices is relative to its value. Lighter cells have larger values and darker cells smaller values. The linkage groups matrix (third column in the figure), is an $n\times n$ matrix, which each of its entries correspond to memberships in the linkage groups. If the value of the i^{th} row and j^{th} column is one(white), the i^{th} and j^{th} variables are in a same linkage group and if zero (black), otherwise. Since we are solving a concatenated Trap-5 problem, we expect the error-free MIM to have ten 5×5 white squares along the main diagonal, while all other cells of the matrix are expected to be zero (black). For the matrix H, in the error-free case, for each of the linkage groups there should be at least one factor node (corresponding to the columns in the matrix H) with a light bar, without any other disconnected light cells in the same column.

As it is obvious in the figure, in the fourth generation (second row, third column) all the linkage groups are identified. But although the linkage groups are identified in the fourth generation, the optimum of the problem is not found up until the tenth generation (fourth row). That is due to the fact that some more generations are required after discovering the BBs, to search inside the BBs and mix the BBs in different individuals, in order to find the optimum solution. In this sample only BB-wise crossover is used to find the optimum configuration.

It is obvious from the figure that some of the columns (factor nodes) of the matrix H (second column of the figure 2) have very low degree (λ) (λ is used as the degree of the factor nodes and is defined in section 4), so that their corresponding cells are almost zero. Along with the generations, the amount of noise in the MIM and H are getting smaller. For the first generation, the number of the factor nodes is set to $maxFN = n = 50$. Along with the optimization algorithm, the number of unique and high-degree factor nodes approaches to its actual value which in this example is 10. The excess factor nodes (other 40 factor nodes) are either duplicates of the true factor nodes or have a very low degree. For example in the bottom row of the figure 2, 43 out of 50 columns have visually distinguishable degrees and these 43 columns are duplicate and there are no false factor nodes among them. Ten unique factor nodes out of these 43 factor

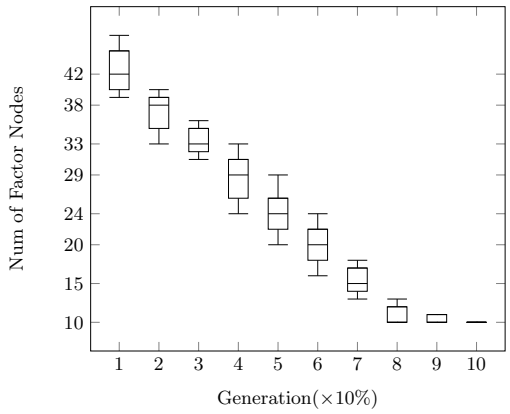

Figure 3: Number of factor nodes for the concatenated trap-5 problem with 50 variables over 100 runs.

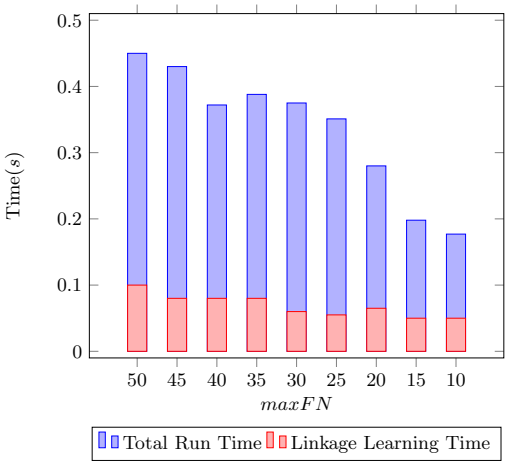

Figure 4: The effect of $maxFN$ on the total run time and linkage learning time for Trap5 problems with 50 variables.

nodes are the actual factor nodes which are utilized as the input to other steps of the algorithm.

Looking at the matrix H in the second row of the figure, it is obvious visually from the figure that there are almost no errors (disconnected light cells) at the columns with high degrees (columns with light bars). As it is shown in the figure, although we do not provide the algorithm with the actual number of factor nodes of the problem, our SNMF approach can find the necessary number of factor nodes and thus the linkage groups. Initially, the number of factor nodes is set equal to the number of variables of the problem. Again it is shown that the optimization search for the optimum value which is performed by BB-wise crossover takes some more generations after completely learning the linkage groups (in the example, from generation 4 to generation 10 that the optimum is found). We may be able to decrease the necessary generations by applying some other sampling method, instead of the simple BB-wise crossover. More research on this issue is left for the future work.

5.2 Parameter maxFN

In figure 3, the number of identified unique factor nodes during the algorithm for non-overlapping Trap-5 problem with 50 variables is averaged over 100 runs and depicted as box plots. During the algorithm, number of identified unique factor nodes converges to the actual number of BBs (figure 3). In figure 4, the total running time of the algorithm and the time spent on the factorization part of the algorithm for different initial value of $maxFN$ parameter is depicted. As it is expected, the running time gets lower when $maxFN$ is set closer to the real m.

5.3 SNMF for learning the linkage groups in overlapping problems

In this subsection, the learned factor graphs for problems with overlapping building blocks are presented. In figure 5, the MIM, H and linkage group matrices of a concatenated trap-5 problem with 32 variables and 10 building blocks is presented for the population size 20000. Each two neighbouring BBs have 2 variables in common. $maxFN$ is set to $n/2$ in this example. As it can be seen the SNMF approach have identified the linkage groups almost perfectly. As mentioned before because BB-wise genetic operators are unable to find optimum configuration for problems with overlapping BBs, we have applied the SNMF learning approach only for one generation for this problem.

6. COMPLEXITY ANALYSIS

In this section, the number of function evaluations and the time complexity of the proposed optimization algorithm are discussed. As explained above, the algorithm is comprised of three main steps of (1) constructing the MIM, (2) learning the factor graph (3) Generating new samples through Gibbs sampling and BB-wise crossover.

Number of evaluations: In the first step of the algorithm evaluations are done for constructing the MIM. Each individual in the population is evaluated exactly once in the first step. Therefore, if the population size is N, number of fitness evaluations in each iteration would be N and if number of generations is g, the overall number of evaluations of the algorithm would be $N \times g$.

Time: Pair-wise dependencies (mutual information) are computed by a pass through all the strings in the population. As MI values are calculated for each pair of variables therefore this computation is done in $O(n^2 \times N)$, where n is the problem size and N is the population size.

In the process of learning the factor graph, using the divergence distance for minimizing the distance between the two adjacency matrices (equation (8)), it is only needed to sum over all non-zero terms of matrix MIM for each update. So, If MIM is sparse, the time complexity of the equation (9) is $O(m^2 \times L)$, where L is the number of nonzero entries in MIM, m is the number of factor nodes. L is of order $O(n^2)$ in the worst case. The complexity of the sampling part of the algorithm is $N/2 \times C_n \times \sum_{i=1} n2^{MN} \approx N \times n \times 2^{MN}$. Therefore the overall time complexity of the algorithm is $O(n^2 \times N + m^2 \times n^2 + N \times n)$. Considering the worst case where $L = n^2$, time complexity of the FGEDA is $O(n^2 N + m^2 n^2)$.

7. REFERENCE ALGORITHMS

Two approaches are used as reference algorithms. Both are well studied optimization approaches in the category of

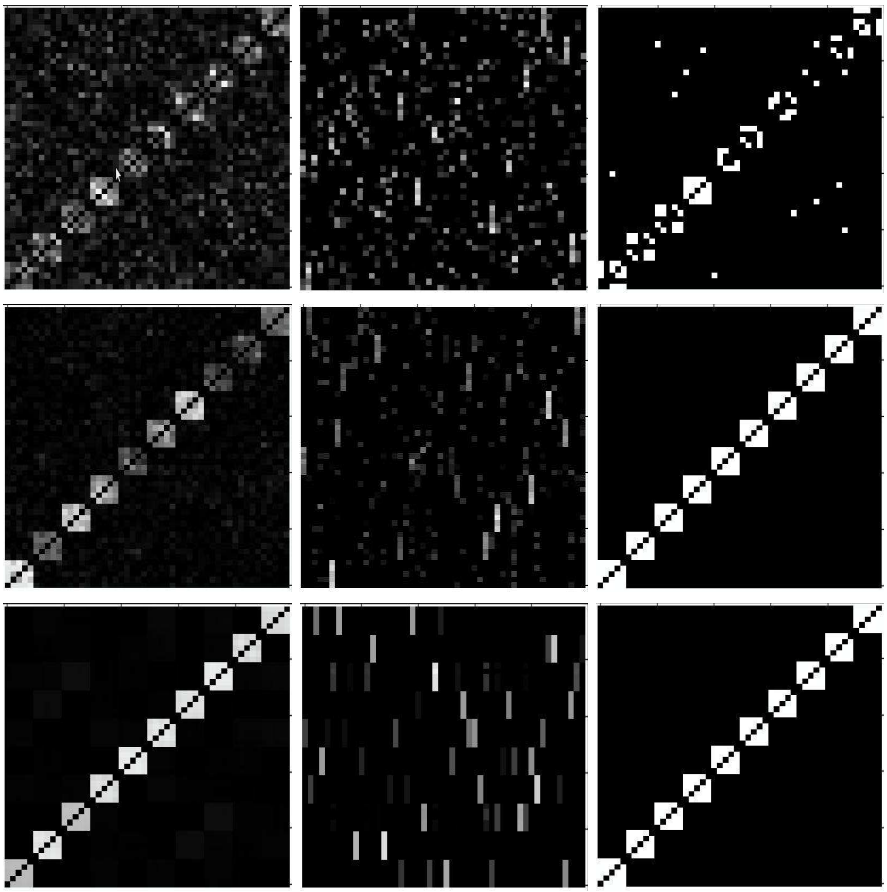

Figure 2: Matrix of MI, SNMF and linkage groups from left to right in generations 1, 4 and 10 from top to bottom for Trap-5 problem with 50 variables

estimation of distribution algorithms. Hierarchical Bayesian optimization algorithm (hBOA) [1] and dependency structure matrix genetic algorithm (DSMGA) [6] are used to compare the results. DSMGA uses mutual information to construct the dependency structure matrix and then cluster the DSM by an evolutionary strategy with a fit-to-data objective function. Here we have used DSMGA++ version [6] of the DSMGA which uses (s+w) algorithm for generating new solutions (for more information refer to [6]).

hBOA uses conditional probabilities of variables, a greedy algorithm and a scoring metric to learn a Bayesian network as the underlying probabilistic model of the problem. hBOA then uses the learned model and conditional probabilities to sample new potential solutions. For all the three algorithms, restricted tournament replacement is used to replace the population. For both DSMGA++ and hBOA parameters are set to their best configurations reported in [1, 6].

8. EXPERIMENTAL RESULTS

In this section, the experimental results are presented. First the experiments and test functions are explained and then the results are shown and analyzed.

Overlapping Trap

An additively decomposable Trap function (equation (14)) of order k is defined as the sum of single Trap functions of order k:

$$f(x) = \begin{cases} fHigh & \text{if } u(x) = k \\ fLow - u(x) * fLow/k & \text{else} \end{cases} \quad (14)$$

where $u(x)$ returns the number of 1s in string x. Usually $fHigh = 1$ and $fLow = 0.9$. Here we use overlap of order one between Trap sub-functions in which a variable in a block is common to the next block.

Neaest-neighbour NK landscape

An NK fitness landscape is defined by: (1) The number of bits, n. (2) The number of neighbours per bit, k. (3) A set of k neighbours $\Pi(X_i)$ for the i-th bit, X_i, for every $i \in \{0, ..., n-1\}$. (4) A sub-function g_i defining a real value for each combination of values of X_i and $\Pi(X_i)$ for every $i \in \{0, ..., n-1\}$. A look-up table with 2^{k+1} values defines each sub-function. Maximizing the objective function (15) is the goal.

$$f_{nk}(X_0, ..., X_{n-1}) = \sum_{i=0}^{n-1} g_i(X_i, \Pi(X_i)). \quad (15)$$

The four components of an NK problem instance affect the difficulty of optimizing NK landscapes. NK landscapes have attracted researchers in stochastic optimization because of their difficulties and properties.

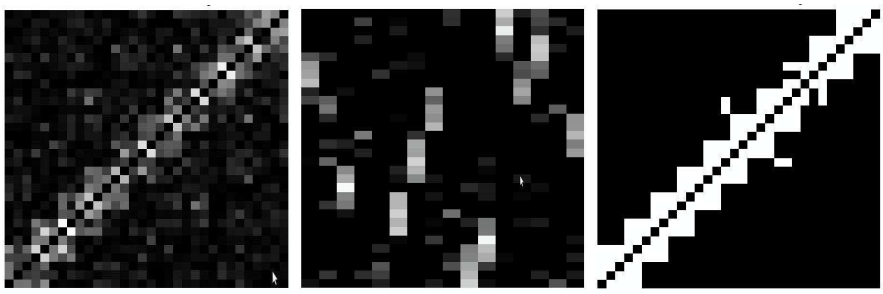

Figure 5: Left: 32×32 matrix of mutual information for trap5 problem of size 32 with overlapping building blocks (2 variables overlap). Middle: the matrix H. Right: the linkage groups constructed using SNMF

Algorithm 2 Algorithm of the proposed approach

Output: optimum solution

1: Create random population.
2: **repeat**
3: Evaluate the population.
4: Select population by tournament selection.
{Computing the pairwise dependencies $e_{ij} \in E$ and constructing the MIM, $MIM(V, E)$}
5: **for** all the individuals in the population **do**
6: **for** each variable i and variable j, **do**
7: Calculate $e_{ij} = MI(i, j)$
8: **end for**
9: **end for**
10: Prune the MIM by threshold $= \mu_{MIM}$.
11: Do the SNMF and find the factor nodes using the algorithm 1.
12: Calculate the Markov blanket factors.
13: Generate $N/2$ solutions by Gibbs sampling.
14: Generate $N/2$ solutions by BB-wise crossover.
15: Replace population.
16: **until** optimum found

Results

Each run is terminated either when the global optimum has been found or when the population has converged to a single solution or when the maximum number of iterations $g \approx n$ has been reached. The results are shown for the overlapping Trap functions with linkage groups of sizes 5 and 7. Population size is determined by bisection with 30 successful runs. The reported number of fitness evaluations are averaged over 30 independent runs. For the NK landscapes problem, we use $k = 5$. For each n, we use 1000 unique, independently generated instances.

To evaluate performance of the algorithm, the overall number of fitness evaluations is reported. The number of fitness evaluations for hBOA, DSMGA++ and FGEDA to solve overlapping Trap-5 and overlapping Trap-7 functions are depicted in figures 6 and 7. Results on NK-Landscape with $k = 5, 6$ are shown in figures 8 and 9. Comparing the results visually, the FGEDA needs smaller number of fitness evaluations and grows slower with regard to the problem size.

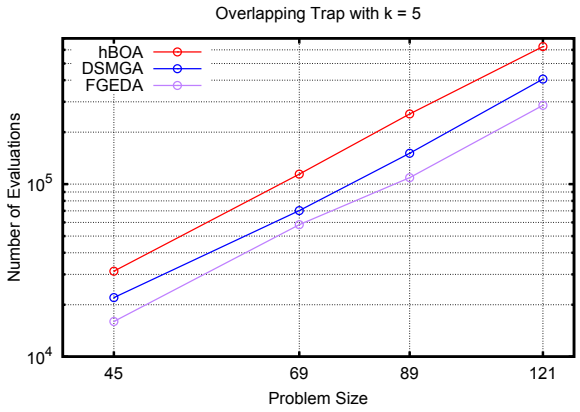

Figure 6: Number of fitness evaluations for hBOA, DSMGA++ and FGEDA to solve overlapping Trap-5.

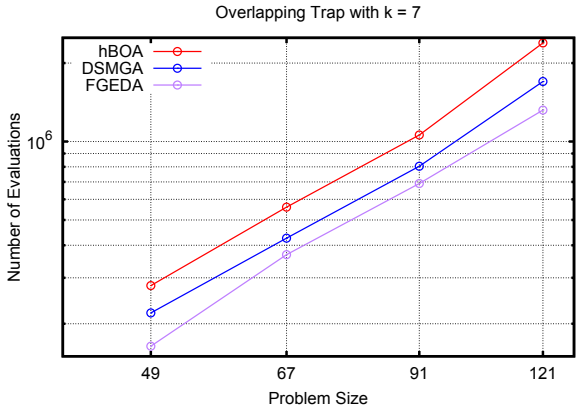

Figure 7: Number of fitness evaluations for hBOA, DSMGA++ and FGEDA to solve overlapping Trap-7.

9. SUMMARY AND CONCLUSION

This paper introduces a estimation distribution algorithm. We have used symmetric non-negative matrix factorization

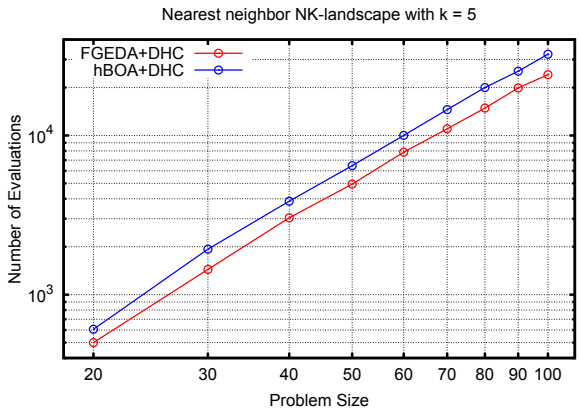

Figure 8: Number of fitness evaluations for nearest neighbor NK-Landscape with $k = 5$ for FGEDA and hBOA

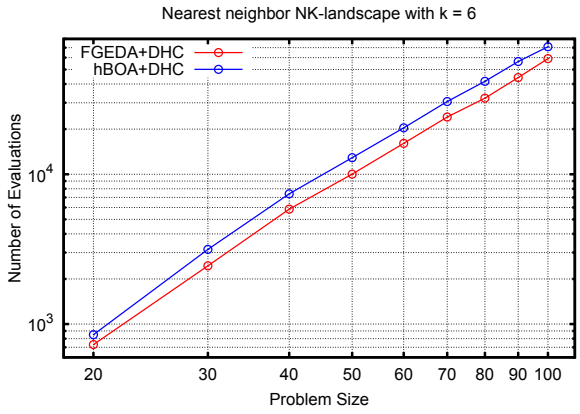

Figure 9: Number of fitness evaluations for nearest neighbor NK-Landscape with $k = 6$ for FGEDA and hBOA

for factorizing the matrix of mutual information and learning the factor graph of the problem. Markov blanket canonical parameterization is used to approximate the Gibbs distribution encoded by the factor graph. Having both the linkage groups and the joint distribution, new potential solutions are generated using Gibbs sampling and BB-wise crossover. To demonstrate the performance of the FGEDA, the results on overlapping Trap functions and NK-landscape problems are reported. It is shown that FGEDA can solve all the tested problems with polynomial number of fitness evaluations in polynomial time with respect to length of the problem. The results are compared to the results of two well-known reference approaches and it is shown that performance of the three methods is comparable in terms of the number of function evaluations. One of the advantages of FGEDA is that it learns the factor graph which is naturally the ideal model to encode the variable dependencies, in polynomial time. The algorithm description is easy and there is no black-box part in the algorithm. The learned model is simple to understand for humans. This can be a valuable property for the experts. The learning process has strong mathematical background and it is mathematically proved to converge. Having the factor graph we can also apply other methods like LBP to calculate the MAP for enhancing the sampling phase of the algorithm which is left for future works.

10. REFERENCES

[1] M. Pelikan. *Hierarchical Bayesian Optimization Algorithm*. Springer-Verlag Berlin Heidelberg, 2005.

[2] S. Shakya and A. Brownlee and J. McCall and F. Fournier and G. Owusu. A fully multivariate DEUM algorithm. *In (CEC-2009)*, pages 479-486. 2009.

[3] P. Larranaga and J. Lozano. *Estimation of Distribution Algorithms: A New Tool For Evolutionary Computation*. Kluwer Academic Pub, 2002.

[4] T. Mahnig and H. Muhlenbein. FDA - a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evol. Comput.*, 7(4):353–376, 1999.

[5] K. Sastry and D. E. Goldberg. *On Extended Compact Genetic Algorithm*. (IlliGAL Report No. 2000026), UIUC, Urbana, IL, 2000.

[6] T.-L. Yu. *A Matrix Approach for Finding Extrema: Problems With Modularity, Hierarchy, and Overlap*. PhD thesis, UIUC, Urbana, IL, 2006.

[7] F. R. Kschischang and B. J. Frey and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2):498-519. 2001.

[8] H. Mühlenbein. *Convergence of Estimation of Distribution Algorithms for Finite Samples*. Technical Report. Germany: Fraunhofer Institute Autonomous intelligent Systems, 2008.

[9] R. Santana. Estimation of distribution algorithms with Kikuchi approximations. *Evol. Comput.*, 13(1):67-97. 2005.

[10] A. Mendiburu and R. Santana and Jose A. Lozano. *Introducing Belief Propagation in Estimation of Distribution Algorithms: A Parallel Approach*. Technical Report EHU-KAT-IK-11-07. Department of Computer Architecture and Technology, The University of the Basque Country, 2007.

[11] D. Kuang and H. Park and C. H. Q. Ding. Symmetric non-negative matrix factorization for graph clustering. *12th SIAM Int. Conf. on Data Mining*, pages 106-117. 2012.

[12] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and Bayesian rotation of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, (6):721-741

[13] G. Harik. *Linkage Learning Via Probabilistic Modelling in the ECGA*. (IlliGAL Report No. 99010). UIUC, Urbana, IL, 1999.

[14] D. Zhou and B. SchÃűlkopf and T. Hofmann. Semi-supervised learning on directed graphs. In *(NIPS-2005)*, pages 1633-1640, 2005.

[15] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. In *(NIPS-2009)*, pages 556-562, 2009.

[16] K. Yu and S. Yu and V. Tresp. Soft clustering on graphs. In *(NIPS-2005)*, pages 0-5, 2005.

[17] P. Abbeel and D. Koller and A. Y. Ng. Learning factor graphs in polynomial time and sample complexity. *J. Mach. Learn. Res.*, 7:1743–1788. 2006.