IETE Journal of Research
Vol 48, No 5, September-October, 2002, pp 339-347

Probabilistic Graphical Model based Approach to Genetic Algorithm Design

CHIRANJIT ACHARYA
Cyber Technologies Laboratory of Sony Corporation, 3300 Zanker Road, San Jose, CA 95134, USA.

AND

SANKAR K PAL, FIETE
Machine Intelligence Unit of Indian Statistical Institute, 203, BT Road, Calcutta 700 035, India.

Genetic algorithms are traditionally formulated as search procedures that make use of selection, crossover and mutation operators to implement the search process. However, in recent times, there has been a growing interest in the GA community to replace the traditional two-parent recombination version of genetic algorithms by building and simulating probabilistic graphical models as the core decision making framework. In this new approach, the models guide the exploration of the search space by constructing the distribution of promising solutions and subsequent forward sampling from the distribution at every evolution step until convergence. In this paper, we survey the current literature of research towards this direction, and also give a detailed exposition of one variant of probabilistic graphical model, namely Bayesian network, which arguably subsumes and generalizes many other models mentioned in the literature.

Indexing terms: *Genetic algorithms, Bayesian network, Multinomial distribution, Model selection, Parameter estimation.*

GENETIC algorithms (GA) [1] have been devised to be general purpose search algorithms for various types of optimization problems. The working principle of these algorithms follows a reductionist model of evolution in natural genetic populations. The basic idea here is to maintain a population of chromosomes, each of which encodes a candidate solution to the problem in context. The chromosome population evolves through a process of competition and controlled variation. Each chromosome in the population has a fitness value associated with it. Based on these fitness values, a population-wide filtering is carried out at every evolution step to determine the subset of chromosomes that are eligible to form new ones. This competitive process is known as selection. Genetic operators, such as crossover and mutation are applied on the filtered subpopulation to create new chromosomes, and the whole process is iterated in succession. The wide measure of success that GA has achieved in search and optimization problems is attributed primarily to their ability to exploit the information accumulated about an initially unknown search space, which introduces useful amount of biases to steer the search in the direction of potential subspaces.

A major portion of the theory of this early version of genetic algorithms deals with the manipulation of building blocks or partial solutions, different permutations and combinations of which generate the solution chromosomes. Because, it has been observed quite often that problem independent, fixed crossover and mutation operators either break building blocks or do not shuffle them properly [2]. In effect, during the recombination process, crucial domain knowledge encoded in partial solutions are lost, causing the search process to be trapped in local minima. Experimental studies show that GA displays good performance only when building blocks preserve tight coupling in solution strings [3].

Several attempts have been made so far to stop or minimize the disruption of building blocks, which may be broadly classified into two approaches. In one approach, the chromosome representation is constrained to reduce the likelihood of disruption of partial solutions. Various reordering and mapping operators, in addition to the usual recombination operators, are used for this purpose [1]. These operators are, however, extremely slow and not powerful enough to ensure proper mixing of building blocks. The other approach makes a complete paradigm shift; instead of implicit reproduction of important building blocks and their mixing by two-parent re-combination operators, it simulates a probabilistic graphical model at every evolution step [4,5], and uses it as the core decision making framework to modify the population. The model preserves global domain knowledge by encoding the probability distribution of building blocks in selected solutions, which is then resampled to generate new solutions.

In this new paradigm, learning models and subsequent

Paper No 64-A;

resampling from them are important design considerations, in addition to more traditional issues, such as population sizing. Models are joint probability distributions of building blocks in a crafted population. Represented as a combination of graphs and conditional probability tables, these models are also known as probabilistic networks. Unless explicitly designed a priori, they are to be learned from data, which is provided by the population of chromosomes. Learning a network involves derivation of its topology and parameters from data, and their respective methods vary greatly with the size of the population. Once the probability distribution is learned, the next step is to simulate that distribution to generate more chromosomes and reorganize the population by replacing some of the old chromosomes with the new ones. Resampling high-fitness solutions from a distribution depends on how accurate the computed distribution is with respect to the true model of the domain, which, in turn, is related to the completeness and complexity of the population.

BAYESIAN NETWORKS

Probabilistic graphical models or probabilistic networks [6] are graph structured descriptions of finite stochastic systems containing finite number of random variables. In a graphical model representation of a problem domain, the graph encodes both the variables and the inter-variable dependency relationships. These graph-encoded dependency or independency relation assertions are further supplemented by the joint probability distribution of all the random variables – either in the form of conditional probability tables if the variables are discrete or discretized, or as continuous probability distributions if the variables are continuous. Usually such networks come in two flavours - as directed graphs, or as undirected graphs. Directed graphical models are commonly known as Bayesian networks, whereas undirected models are called Markov random fields. In this paper, our focus is on directed acyclic graph structures, though many of the related statistical methods are applicable to other models as well.

For a given problem domain with a set of n random variables $\mathbf{X} = \{X_1, X_2, X_3, \ldots, X_n\}$, a Bayesian network can be defined as an algebraic structure $\mathbf{B} = \langle \mathbf{G}, \mathbf{P} \rangle$, where $\mathbf{G} = \langle \mathbf{V}.\mathbf{L} \rangle$ is the graphical representation of the domain knowledge and $\mathbf{P}$ is the corresponding probability descriptions. $\mathbf{V} = \{V_1, V_2, V_3, \ldots, V_n\}$ is the set of nodes in $\mathbf{G}$, such that each random variable $X_i \in \mathbf{X}$ is represented by its corresponding node $V_i \in \mathbf{V}$. We will therefore use X_i to denote both the i-th domain variable and the i-th node in $\mathbf{B}$ interchangeably. $\mathbf{L}$ is the set of links in $\mathbf{G}$, such that if X_i is conditionally dependent on X_j then $\overrightarrow{X_j X_i} \in \mathbf{L}$. $\mathbf{P} = \{P_1, P_2, P_3, \ldots, P_n\}$ is the set of local conditional probability distributions, each one modeling the localized dependencies associated with a random variable. So, $\langle \mathbf{G}, \mathbf{P} \rangle$ as a whole defines the joint distribution over $\mathbf{X}$. The probabilities encoded in the Bayesian network can be either physical probabilities or belief measures. Physical probabilities are assumed when networks are learned from a database, and belief measures or Bayesian probabilities are assumed when networks are built with prior knowledge.

Let us assume, for convenience, that the random variables are ordered in a serialized fashion $\{X_1, X_2, \ldots, X_n\}$, such that for all X_i, its nondescendents in $\mathbf{G}$ will be $\mathbf{V}_i = \{X_{i+1}, X_{i+2}, \ldots, X_n\}$. Then, following the chain rule, the joint probability distribution over $\mathbf{X}$, given the background knowledge ξ, can be defined as:

$$P(X_1, X_2, \ldots, X_n | \xi) = \prod_{i=1}^{n} P(X_i | X_{i+1}, X_{i+2}, \ldots, X_n, \xi)$$

The probability component $P_i \in \mathbf{P}$ refers to the conditional distribution $P(X_i | X_{i+1}, X_{i+2}, \ldots, X_n, \xi)$. P_i takes the form of a density function in continuous domains, and a matrix form containing case-by-case enumerations in discrete domains. In a fully or densely connected Bayesian network, the number of parameters to be learned for describing all the local distributions is enormous. So in large scale problems, it is almost essential to create a parsimonious structure for $\mathbf{G}$, which will, in turn, reduce the description length of $\mathbf{P}$ by minimizing the number of parameters that needs to determined. This is achieved by introducing the principle of conditional independence, which says that any variable $X_i \in \mathbf{X}$ is conditionally independent of its nondescendents, given its immediate ancestors $\mathbf{A}_i \subseteq \{X_{i+1}, X_{i+2}, \ldots, X_n\}$. Under such conditional independence assertion, for every $\mathbf{X}_i$, all those links $\overrightarrow{X_j X_i} \in \mathbf{L}$ providing entries in the local distribution P_i become irrelevant where $X_j \in \mathbf{V}_i - \mathbf{A}_i$. So $P(X_i | X_{i+1}, X_{i+2}, \ldots, X_n, \xi)$ effectively reduces to $P(X_i | A_i, \xi)$, and the factorized joint distribution can be redefined as:

$$P(X_1, X_2, \ldots, X_n | \xi) = \prod_{i=1}^{n} P(X_i | \mathbf{A}_i, \xi)$$

Genetic algorithms, where building blocks are discrete, multivalued elements drawing their configuration from a finite alphabet, may be modeled with multinomial distributions in Bayesian networks. Assuming that each random variable X_i takes on v_i values $x_i^1, x_i^2, \ldots, x_i^{v_i}$, and the corresponding ancestral set A_i takes on v_i configurations $\mathbf{a}_i^1, \mathbf{a}_i^2, \ldots, \mathbf{a}_i^{v_i}$, in a Bayesian network with topology $\mathbf{G}$ and parameter set Θ, one parameter can be defined for each dependency relation $(X_i = x_i^k, \mathbf{A}_i = \mathbf{a}_i^j)$:

$$P(X_i = x_i^k | \mathbf{A}_i = \mathbf{a}_i^j, \mathbf{G}, \Theta, \xi) = \theta_{ijk}$$

Let Θ_{ij} denote the set of all parameters characterizing the distribution $P(X_i | A_i = \mathbf{a}_i^j, \mathbf{G}, \Theta, \xi)$. Then

$$\Theta_{ij} = \bigcup_{k=1}^{u_i} \{\theta_{ijk}\}$$

Then each local probability distribution $P(X_i | A_i, \mathbf{G}, \Theta, \xi)$ can be defined as Θ_i where:

$$\Theta_i = \bigcup_{j=1}^{u_i} \Theta_{ij} = \bigcup_{j=1}^{u_i} \bigcup_{k=1}^{u_i} \{\theta_{ijk}\}$$

Following this notation, the global parameter set Θ, that characterizes all the local distributions $P_1, P_2, \ldots, P_n$ in the Bayesian network, can be defined as:

$$\Theta = \bigcup_{i=1}^{n} \Theta_i = \bigcup_{i=1}^{n} \bigcup_{j=1}^{v_i} \bigcup_{K=1}^{u_i} \{\theta_{ijk}\}$$

Since for a given graph structure $\mathbf{G}$ of a Bayesian network $\mathbf{B}$, the probabilistic relational descriptions can be completely defined by Θ, we may have an alternative presentation of $\mathbf{B}$ as $\langle \mathbf{G}, \Theta \rangle$. Probabilistic *GA* requires learning both $\mathbf{G}$ and Θ, which is discussed later.

NETWORK LEARNING

Learning Bayesian networks can be viewed as a two-step process [7,8]: first, structure learning from data and second, parameter learning from structure and data. It takes different forms depending on whether the structure is known or unknown, and whether data is complete or incomplete - in the sense of full observability of data vectors. So, there are four distinct cases of learning: (1) known structure and complete data, (2) known structure and incomplete data, (3) unknown structure and complete data, and (4) unknown structure and incomplete data. In genetic algorithms, since all building blocks for all chromosomes have certain assigned states, it is perfectly safe to assume that the case of incomplete data will never arise while modeling distributions of building blocks. Therefore, we will discuss here learning methods for known and unknown structures under complete data assumption.

Known Structure & Complete Data

With the network structure already specified, the only task of the inducer in this case is to estimate the parameters, which is typically done by using the estimation techniques such as maximum likelihood (ML) or maximum-a-posteriori (MAP) Bayesian approximation [9]. As the number of samples grows, ML and MAP estimators converge to each other under uniform prior.

The most widely applied class of priors are the Dirichlet priors [10]. They provide closed form solutions, and by virtue of their conjugacy property, generate posteriors in the same functional form as priors. For example, let's take multinomial sampling in a single variable setting. The observed random variable X is discrete, having u possible values $x_1, x_2, \ldots, x_u$. This system is parameterized by a set of parameters $\Theta = \{\theta_1, \theta_2, \ldots, \theta_u\}$, such that $P(X = x_i | \Theta, \xi) = \theta_i$ and $\sum_i \theta_i = 1$. A simple conjugate prior may be defined over this system by a Dirichlet distribution with a set of hyperparameters $\alpha_1, \alpha_2, \ldots, \alpha_u$:

$$P(\Theta | \xi) = \mathrm{Dir}(\Theta | \alpha_1, \alpha_2, \ldots, \alpha_u) = \frac{\Gamma(\alpha)}{\prod_{i=1}^{u} \Gamma(\alpha_i)} \prod_{i=1}^{u} \theta_i^{\alpha_i - 1}$$

where $\alpha = \sum_i \alpha_i$ and $\alpha_i > 0$, $i = 1, \ldots, u$. If the sufficient statistics in the dataset $\mathbf{D}$ are $m_1, m_2, \ldots, m_u$, where m_i is the frequency of $X = x_i$ in $\mathbf{D}$, the posterior ditribution of the system is:

$$P(\Theta | \mathbf{D}, \xi) = \mathrm{Dir}(\Theta | \alpha_1 + m_1, \alpha_2 + m_2, \ldots, \alpha_u + m_u)$$
$$= \frac{\Gamma(\alpha + |\mathbf{D}|)}{\prod_{i=1}^{u} \Gamma(\alpha_i + m_i)} \prod_{i=1}^{u} \theta_i^{\alpha_i + m_i - 1}$$

These results can be extended for a Bayesian network of n multinomial, random variables $X_1, X_2, \ldots, X_n$. Each dependency relation $(X_i = x_i^k, \mathbf{A}_i = \mathbf{a}_i^j)$ being specified by a parameter θ_{ijk} with corresponding statistic m_{ijk}, the maximum likelihood estimate can be derived as: $\theta_{ijk} = m_{ijk}/m_{ij}$, where $m_{ij} = \sum_k m_{ijk}$. And for the Bayesian case, the posterior distribution is:

$$P(\Theta_{i,j} | \mathbf{D}, \mathbf{G}, \xi)$$
$$= \mathrm{Dir}(\Theta_{i,j} | \alpha_{ij1} + m_{ij1}, \alpha_{ij2} + m_{ij2}, \ldots, \alpha_{iju_i} + m_{iju_i})$$

This distribution is used to generate a maximum-a-posteriori estimate. Expectation maximization [11] is used for getting both ML and MAP estimate of Θ if $\mathbf{D}$ is actually generated from a given network topology $\mathbf{G}$, then both ML and MAP estimates converge asymptotically to the correct parameter setting of $\mathbf{G}$ in the limit. Otherwise, they converge to some distribution that is close to the actual joint distribution represented by the specified Bayesian network.

Unknown Structure & Complete Data

In this case, at first, structure learning procedures are applied to determine the network topology, and then parameters are derived for that topology. Once the structure is defined, same methods as defined earlier are used for parameter learning. Here, we will discuss only the structure learning methods.

There are two related components of structure learning: a scoring metric to decide fitness values of different network configurations with respect to data, and a heuristic search algorithm that makes use of the scoring metric to explore the structure space [8]. Several statistical methods are there to construct scoring function. Among them, most widely used ones are MAP, ML, extended likelihood methods, such as Akaike and Bayesian information criteria, and minimum description length principle.

In the context of learning the graph structure $\mathbf{G}$ of a Bayesian network from data $\mathbf{D}$, MAP computation seeks to find the structure that maximizes the posterior probability

over the G-space:

$$\mathbf{G}_{MAP} = \arg\max_{\mathbf{G}} P(\mathbf{G} \mid \mathbf{D}, \xi)$$

$$\approx \arg\max_{\mathbf{G}} [\log P(\mathbf{D} \mid \mathbf{G}, \xi) + \log P(\mathbf{G} \mid \xi)]$$

$\log P(\mathbf{D} \mid \mathbf{G}, \xi)$ is termed as sample likelihood. Given a fixed graph structure **G**, this may be derived by marginalizing over the parameter space of the graph:

$$P(\mathbf{D} \mid \mathbf{G}, \xi) = \int_{\Theta} P(\mathbf{D} \mid \mathbf{G}, \Theta, \xi)\, P(\Theta \mid \mathbf{G}, \xi)\, d\Theta$$

This approach requires defining the priors $P(\mathbf{G} \mid \xi)$ and $P(\Theta \mid \mathbf{G}, \xi)$, which can be either informative or noninformative. Informative priors incorporate critical domain knowledge which may be translated into constraints and preferences regarding the structure of the network, as well as preferences on parameter values. Noninformative priors lack such domain knowledge, but there are well-established empirical rules, eg maximum entropy, to construct this kind of priors. Sometimes, brute-force approximation, such as Bayes factor, are used where relative joint probability of structure and data is calculated with respect to some base structure $\mathbf{G}_0$:

$$\mathbf{G}_{MAP} = \arg\max_{\mathbf{G}} \frac{P(\mathbf{G}, \mathbf{D} \mid \xi)}{P(\mathbf{G}_0, \mathbf{D} \mid \xi)}$$

A simplification of the MAP Bayesian method is maximum likelihood, where uniform prior over the structure space is assumed:

$$\mathbf{G}_{ML} = \arg\max_{\mathbf{G}} P(\mathbf{D} \mid \mathbf{G}, \xi)$$

$$\approx \arg\max_{\mathbf{G}} \left[\arg\max_{\Theta} \log P(\mathbf{D} \mid \mathbf{G}, \Theta, \xi) \right]$$

In absence of any prior, this approach seeks the network **G** for which the data-likelihood over $\langle \mathbf{G}, \theta \rangle$ is maximum. When large sample data is available, employing the maximum likelihood approach is a pragmatic strategy [12], as it avoids the high computational complexity of the Bayesian methods. But the problem of maximum likelihood is overfitting in the dataspace. Akaike information criteria (*AIC*) and Bayesian information criteria (*BIC*) [9] have been proposed as penalized likelihood approach to minimize overfitting. The empirical form of *AIC* is:

$$AIC(\mathbf{G} : \mathbf{D}) = -\log P(\mathbf{D} \mid \mathbf{G}, \hat{\Theta}, \xi) + |\Theta|$$

and that of *BIC* is:

$$BIC(\mathbf{G} : \mathbf{D}) = -\log P(\mathbf{D} \mid \mathbf{G}, \hat{\Theta}, \xi) + \frac{|\Theta|}{2} \log |\mathbf{D}|$$

These likelihoods tend to draw a balance between maximum likelihood and maximum network parsimony. In both cases, the basic approach is to select the model for which the respective functional produces minimum value. *AIC* and *BIC* are asymptotically Bayesian but avoid specification of priors. Typical application of this kind of penalized likelihood methods for model selection in probabilistic networks appear in [13]. Structure learning research has sometimes been driven by the paradigm of minimizing encoding complexity, rather than maximizing posteriors or likelihood probabilities. One typical approach in this category is minimum description length (MDL) principle [14], which performs hypothesis induction by seeking a model that enables the most compact encoding of both the theory and available data. With this principle in place, a structure learning procedure tends to minimize the following functional:

$$MDL(\mathbf{G} : \mathbf{D}) = -\log P(\mathbf{D} \mid \mathbf{G}, \Theta, \xi) + \frac{|\Theta|}{2} \log |\mathbf{D}| + \left(\frac{|\Theta|}{2} + 1 \right) \log (|\Theta| + 2)$$

With different scoring metrics drived in sufficient details, the next step to learning is a heuristic search to find out a network configuration that returns maximum value for the specified scoring metric. The algorithm starts with a randomly initialized network, and then applying structure manipulation operators, such as edge addition, edge deletion and edge reversal, explore the model space. The most commonly used algorithm for optimization search is greedy hill climbing which is shown in Fig 1.

```
Select a random graph structure G.
Compute Θ and the score, f(G).
For i = 1 to restart_max
    For j = 1 to flip_max
        Generate successor graphs G_1, G_2, ..., G_q.
        For k = 1 to q
            Compute Θ_k and the score, f(G_k).
        EndFor
        Set G* = arg max_k f(G_k)
        If f(G*) > f(G), set G = G*
        Else return G.
        EndIf
    EndFor
EndFor
```

Fig 1 Greedy search algorithm

PROBABILISTIC GENETIC ALGORITHMS

Genetic algorithms that work under the general rubric of probabilistic graphical models are commonly referred as the estimation of distribution algorithms (EDA) in the GA literature. The basic difference of this EDA class of GA from traditional GA is that in the EDA there are no crossover and mutation operators; instead, a new set of chromosomes is sampled by simulating the distribution model estimated from the pool of selected chromosomes of

the previous population. This is explicit from the design of the EDA itself, but there are more subtle reasons for introducing probabilistic networks. In the traditional GA, dependency relations among building blocks are assumed but not necessarily modeled in an explicit manner, whereas in the EDA, such inter-block dependency relations are necessary components for graphical model building and are encoded in the joint probability distributions associated with each building block.

Drawing connection from earlier mentioned graphical models, a chromosome having n genes can be represented as an n-dimensional multivariate $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, where each random variable X_i corresponds to the i-th building block in the chromosome. The starting point of EDA is to construct a population $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m\}$, each $\mathbf{d}_i = \{d_i^1, d_i^2, \ldots, d_i^n\}$ being an instantiation of $\mathbf{X}$. During the evolution process, at every iteration step t, a high-quality chromosome subset $\mathbf{D}'$ is created through selection from $\mathbf{D}$. After that, population reproduction is a two-stage process - model building and forward sampling. In the model building phase, the joint probability distribution $P(\mathbf{X})$ is approximated as

$$P(\mathbf{X}) \approx P(X_1, X_2, \ldots, X_n | \mathbf{D}', \xi)$$

It is in this phase when probabilistic graphical modeling with all its subtleties of structure learning and parameter learning comes into play. Forward sampling is really much simpler, where all we need to do is to simulate the afore-built joint distribution to create new chromosomes that will eventually make its way into the modified population for further evolution.

EDA Research

The primary motivation for devising the EDA methodology is that a chromosome distribution model, when designed accurately, can readily capture and preserve the building-block structure of a problem and ensure effective mixing and reproduction of building blocks. Depending on the complexity of inter-block dependencies, EDA techniques are classified into three general categories: (1) EDA with no interactions, (2) EDA with pair-wise interactions and (3) EDA with complex multi-way interactions. In the next few subsections we will describe the current research on each of them in more detail.

EDA with No Interactions

The simplest approximation to the joint distribution of a multivariate $\mathbf{X}$ can be made by assuming that the variables are mutually independent, which means each variable takes up values regardless of the behaviour of the remaining variables. This renders the joint distribution factorized with n independent, univariate, local distributions:

$$P(X_1, X_2, \ldots, X_n | \mathbf{D}', \xi) = \prod_{i=1}^{n} P(X_i | \mathbf{D}', \xi)$$

Each $P(X_i | \mathbf{D}', \xi)$ may be interpreted as the frequency matrix of X_i conditioned on $\mathbf{D}'$. The model, used to generate new solution strings, contains only a set of such frequency matrices, and no structure, as $\mathbf{L} = \emptyset$ because of mutual independence among variables. Hence, learning in this type of EDA is only for parameters, and not for structures.

First known approach in this category is bit-based simulated crossover (BSC) [15]. BSC introduces a bit-by-bit modeling scheme where it assigns a parameter proportional to its evaluation function to every bit.

Population-based incremental learning (PBIL) [16] is another prominent no-interaction EDA, where the idea of learning is built around a core data structure known as probability vector. If each random variable X_i takes on u_i possible values, then the probability vector is of dimension $\Sigma_i u_i$, and each entry of that vector is initialized with maximum entropy assumption. After generating a number of new solutions, the fittest ones among them are selected, and the probability vector is shifted towards the selected solutions by using Hebbian learning rule [17].

In the univariate marginal distribution algorithm (UMDA) [18], each $P(X_i | \mathbf{D}', \xi)$ is estimated from the relative marginal frequencies of X_i within the scope of $\mathbf{D}'$. At each evolution step, these computed frequencies model the selected set of promising solutions, from which new solutions are generated. This process is continued until some convergence criteria are met. BSC and PBIL are arguably two special cases of UMDA [3].

The compact genetic algorithm (CGA) [19] replaces the population with a single probability vector, as is done in PBIL. However, unlike PBIL, it modifies the probability vector to comply with the direct correspondence between the population that is represented by the probability vector and the probability vector itself. Instead of shifting the vector components proportionally to the distance from either 0 or 1, each component of the vector is updated by shifting its value in proportion of the contribution of a single individual to the total frequency, assuming a particular population size.

All the above-mentioned algorithms perform in a similar fashion, i.e., they use frequencies to guide further search, and recreate new chromosomes according to those frequencies. They are especially well-suited for linear problems with no inter-variable dependencies, where they achieve almost linear or sub-quadratic performance, depending on the type of the problem, and they fail on problems where strong interactions among variables are observed.

EDA with Pairwise Interactions

The general idea of constructing EDA with pairwise interactions is to partition the multivariate $\mathbf{X}$ into a set of q mutually exclusive clusters $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_q$ and then to come

up with partition-wise optimal permutations that elucidate pairwise interactions among random variables. Every partition $\mathbf{X}_i$ has n_i elements: X_i^1, X_i^2, ... , $X_i^{n_i}$, such that $\sum_i n_i = n$, and for all $\mathbf{X}_i$ and $\mathbf{X}_j$, $\mathbf{X}_i \cap \mathbf{X}_j = \emptyset$. The objective is to approximate the original joint distribution $P(\mathbf{X} \mid \xi)$ with a fabricated one, $P(\mathbf{X} \mid \mathbf{D}', \xi)$, which exploits inter-partition independence as:

$$P(\mathbf{X} \mid \mathbf{D}', \xi) = \prod_{i=1}^{n} P(\mathbf{X}_i \mid \mathbf{D}', \xi)$$

Within each partition $\mathbf{X}_i$, every random variable X_i^j is linked to at most one ancestor X_i^{j+1}, and the resulting structure would not have any cycle. This is really a soft constraint, leaving some more space open for further constraints toward complete specification of intra-partition configurations. The structure can be linear, e.g., chain graph, or nonlinear, e.g. tree or forest, all of which satisfy the pairwise interaction property:

$$P(\mathbf{X}_i \mid \mathbf{D}', \xi) = \prod_{j=1}^{n_i} P(X_i^j \mid X_i^{j+1}, \mathbf{D}', \xi)$$

Combining the inter-partition independencies and intra-partition, pairwise dependencies, the overall joint distribution is realized as:

$$P(\mathbf{X} \mid \xi) \approx P(\mathbf{X} \mid \mathbf{D}', \xi) = \prod_{i=1}^{q} \prod_{j=1}^{n_i} P(X_i^j \mid X_i^{j+1}, \mathbf{D}', \xi)$$

The correctness of the estimated distribution model is measured in terms of the Kullback-Leibler divergence which computes the extent of maximization of mutual information among random variables:

$$H(\mathbf{X}) = \sum_{i=1}^{q} \sum_{j=1}^{n_i} \left[H(X_i^j) + \sum_{k=1, k \neq j}^{n_i} H(X_i^j \mid X_i^k) \right]$$

where $H(X_i^j)$ is the entropy of X_i^j, and $H(X_i^j \mid X_i^k)$ is the mutual information between X_i^j and X_i^k.

Some early work on pairwise-interaction EDA has been mentioned in [20], where the mutual information-maximizing input clustering (MIMIC) approach is described. In MIMIC, $q = 1$ and the additional constraint is: every random variable X_i^j has at most one descendent X_i^{j-1}, which results in a simple, chain graph structure. To avoid $O(n!)$ search-space complexity, a greedy algorithm is reportedly performed to find the best structure.

Dependency trees have been used in [21] to model promising solutions. Here also $q = 1$, but unlike MIMIC, a random variable can have multiple descendents. Carrying the legacy of PBIL, this method also uses a probability vector, containing all pairwise probabilities, as its learning framework.

In the bivariate marginal distribution algorithm (BMDA) [22], pairwise-interaction EDA is applied in its full form with $q > 1$. This results in a forest which generalizes all the other structures. During network learning process, Pearson's chi-square test is used to measure the fitness of plausible structures. Pairwise-interaction EDA is more powerful than EDA with no interaction, as the former is capable of modeling inter-block dependencies of order two. That arguably establishes them as good models for linear and quadratic problem [20-22].

EDA with Multiway Interactions

Multiway interaction among building blocks is a natural extension of the idea of pairwise interaction, where the initial constraint is further softened to make every random variable have multiple ancestors and multiple descendents. The overall structure is a probabilistic graph $\mathbf{G}$, which can be either a union of a set of disjoint subgraphs: $\mathbf{G} \bigcup_{i=1}^{q} \mathbf{G}_i$, each $\mathbf{G}_i$ corresponding to cluster $\mathbf{X}_i$, or a single, monolithic, connected network. For cluster oriented structures, the network distribution model is:

$$P(\mathbf{X} \mid \mathbf{D}', \xi) = \prod_{i=1}^{q} P(\mathbf{X}_i \mid \mathbf{D}', \xi)$$

whereas in monolithic structures, the distribution model is governed by the principle of conditional independence, rendering each random variable X_i, dependent only on its ancestral set $\mathbf{A}_i$

$$P(\mathbf{X} \mid \mathbf{D}', \xi) = \prod_{i=1}^{n} P(X_i \mid \mathbf{A}_i, \mathbf{D}', \xi)$$

EDA with multiway interactions can solve problems with highly overlapping building blocks, which do not fit in the design of pairwise interaction models.

A number of algorithms covering multivariate interactions have been presented in the literature. Among them, proper learning Bayesian networks have been reported in two recent works: the estimation of Bayesian network algorithm (EBNA) [5] and the Bayesian optimization algorithms (BOA) [4]. In both algorithms, factorization of the joint probability distribution, encoded by a Bayesian network, is learnt from the database containing filtered population of high-quality chromosomes. The learned network is then simulated to generate further solutions according to the probability distribution of the building blocks.

EBNA uses penalized likelihood approach to measure the goodness of fit of network structures, found in the search process, with respect to data. Two alternative likelihood criteria have been mentioned for ENBA: Bayesian information criteria (BIC) and penalized marginal likelihood.

The log-likelihood to a Bayesian network is:

$$\log P(\mathbf{D}' \mid \mathbf{G}, \Theta, \xi) = \sum_{i=1}^{n} \sum_{j=1}^{v_i} \sum_{k=1}^{u_i} m_{ijk} \log \left[\frac{m_{ijk}}{m_{ij}} \right]$$

and the information complexity corresponding to BIC is:

$$BIC(\mathbf{G} : \mathbf{D}) = -\log P(\mathbf{D}' \mid \mathbf{G}, \Theta, \xi) + \frac{\log |\mathbf{D}|}{2} \sum_{i=1}^{n} v_i (u_i - 1)$$

In penalized marginal likelihood, introduced in [7], K2 metric is used to measure the likelihood:

$$P(\mathbf{D}' \mid \mathbf{G}, \Theta, \xi) = \prod_{i=1}^{n} \prod_{j=1}^{v_i} \frac{(u_i - 1)!}{(m_{ij} + u_i - 1)!} \prod_{k=1}^{u_i} m_{ijk}!$$

Once the optimum network structure is determined, the next part is to compute the parameters associated with the local probability distributions, which is done in EBNA at every iterative step as follows:

$$E[\theta_{ijk} \mid \mathbf{D}', \mathbf{G}, \xi] = \frac{m_{ijk} + 1}{m_{ij} + u_i}$$

BOA incorporates similar methods for learning Bayesian networks, though it uses Dirichlet metric as a measure of quality of explored networks. Dirichlet metric also, in principle, is based on maximum likelihood criteria, where it measures the fitness consistency of observed data against network instances in the search space. The functional form of structure-likelihood $P(\mathbf{D}' \mid \mathbf{G}, \Theta, \xi)$ is given as

$$P(\mathbf{D}' \mid \mathbf{G}, \Theta, \xi) = \prod_{i=1}^{n} \prod_{j=1}^{v_i} \frac{\Gamma(m'_{ij})}{\Gamma(m'_{ij} + m_{ij})} \prod_{k=1}^{u_i} \frac{\Gamma(m'_{ijk} + m_{ijk})}{\Gamma(m'_{ijk})}$$

where m'_{ij} is the prior knowledge about the number of instances having $\mathbf{A}_i = a_i^j$, and m'_{ijk} is the prior knowledge about the number of occurrence of the dependency relation $(X_i = x_i^k, \mathbf{A}_i = \mathbf{a}_i^j)$. The detail derivation of this formulae can be found in [8].

Other than EBNA and BOA, there are few more algorithms in this category which use probabilistic graphical modeling in a restricted sense, and they may be presented as special cases of Bayesian network based modeling of building block distribution. In the extended compact genetic algorithm (ECGA) [23], the disjoint cluster model is adopted. Each cluster is a composite building block, resulting in vector-valued random variables $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_q$. ECGA uses a minimum description length metric for model selection, which prefers models with less encoding complexity. Here also, to find a good model, a simple greedy algorithm is used. Applying the theory of UMDA it can be shown that for problems that are decomposable into non-overlapping subproblems of a bounded order, ECGA with a good model runs with sub-quadratic time complexity, but does not produce good performance on problems with highly overlapping building blocks e.g., 2-D spin-glass systems or optimum network connection problems.

The fractorized distribution algorithm (FDA), proposed in [3], uses a fixed, factorized distribution as the model of promising solutions throughout the whole computation. The model is assumed to be supplied externally by some domain expert, and does not evolve over time. Therefore, no structural learning is involved in FDA; only marginal and joint probability distributions are updated according to the currently selected set of solutions. It has been proved that when the model is correct, FDA can solve decomposable problems efficiently [3]. Prior information about problems is incorporated into FDA in the form of its factorized decomposition. But in many real-world problems, this prior information is not readily available, thereby limiting its effectiveness only to problems where near accurate structure approximations are available.

Algorithms

With the background of the EDA class of genetic algorithms outlined as above, we can now describe the different algorithms components of EDA in Figs 2 and 3.

Figure 2 is the pseudocode for the main routine. The induction of the graph structure is carried out by a search process, for which there are several candidates, such as greedy search, simulated annealing or best-first search. In general, it is assumed that the order of interaction k is bounded. For $k = 0$, the case is trivial, as it will be an empty network. For $k = 1$, there exists a polynomial time algorithm [7,24], as the problem can be easily reduced to a maximal branching problem. For $k > 1$, determining the best network structure from data with any specified metric is NP-complete for most probabilistic graphical models [25]. Greedy algorithms give good performance, but are suboptimal, as they often get stuck at local minima. To avoid this problem, simulated annealing or best-first search with good bias can be used. Greedy search with random restart also gets rid of some of these common problems.

Next important algorithmic component is simulation and forward sampling, for which the pseudocode is given in Fig 3.

There are quite a few algorithms described in literature

```
Generate initial population D_0 of size s.
For t = 1 to generation_max
    Filter out D'_{t-1} of size m ≤ s from D_{t-1}.
    Create a graph G_t from D'_{t-1}.
    Determine parameters Θ_t for G_t.
    Forward sample D_t of size s from B_t = (G_t, Θ_t)
EndFor
```

Fig 2 EDA general procedure

```
Find an ordering ≺^t in X according to G_t.
For i = 1 to s
  For j = 1 to n
    Sample an instance of X_i^j from P(X_i^j | A_i^j).
  EndFor
  Set D_t[i] = [X_1^j, X_2^j, ..., X_n^j].
EndFor
```

Fig 3 Forward sampling algorithm

for sampling Bayesian networks to generate new data [26]. A popular method is probabilistic logic sampling where configuration of the multivariate $\mathbf{X}$ is generated in a forward way. That means, all root variables are sampled first; then following the lineage link all descendents are sampled in succession.

The EDA class of genetic algorithms, particularly multivariate interaction EDA, show very good performance for large class of separable problems. However, an important issue is about the very nature of problems that are to be solved by EDA. All separable problems - in the sense that they can be decomposed into terms of a bounded order - are not necessarily admissible into this category. Only those separable problems, where true model order can be approximated at a level of solutions of lower order, and by combining the best of which an optimal or near-optimal solution to the original problem can be constructed, are supposed to be approached with probabilistic networks. This order simplification introduces a necessary bias in the search process, because of which the space complexity explored by the algorithm is also substantially reduced.

CONCLUSIONS

In this paper we have reviewed a promising new area of research that attempts to apply probabilistic graphical models to design population-based search algorithms. Like traditional GA, here also population sizing is an important design issue, the importance being highlighted in the context of learning solution distributions in a problem domain. Probabilistic networks, unless explicitly designed a priori, are to be learned from data which is provided by the population of chromosomes. So, the nature of learning varies greatly with the population or sample size. For a small sample, learning follows initial bias or prior introduced in the induction system. Unless prior encodes the underlying distribution reasonably well, sparse approximation with small sample data has less likelihood to produce good prediction for novel cases. This happens because of overfitting in the structure space and the parameter space. It becomes more apparent when priors are left out and entire estimation is made based on maximum likelihood, as it turns out that the maximum likelihood, approximation is in most cases an overestimate of the true sample likelihood. With a large population, learning asymptotically approaches the true model with high probability, since the maximum likelihood estimate gradually converges to the true sample likelihood, the covergence being measured according to some utility criteria such as mean-square error or Kullback-Leibler distance. For medium-sized samples, algorithms perform with varying results, depending on how well their respective biases align with the true model. In all cases, however, the asymptotic error is bounded below by Bayesian optimal eror [12], if Bayesian network is used for modeling the distribution of chromosomes.

Most of the algorithms that have been developed following the EDA methodology adopt one particular form of learning network structures from data which is termed as model selection. This definition arises from the fact that each network topology corresponds to a distinct model, and only one of them is selected based on the data. A related important concept is model uncertainty [13] which brings a different approach to structured learning. Oftentimes it is observed that selection of a single best model from an exponential-sized family of models - as is the case for learning Bayesian networks - is not feasible. Such empirical shortcomings have motivated researchers to find out a new approach, called selective model averaging, where a finite subset of plausible models with associated uncertainty is derived and averaged for any practical prediction. It is computationally harder than model selection approach - both in terms of sample complexity and computational compexity, but can be a good practical approximation to full Bayesian approach.

To sum it up altogether, EDA is a fundamentally new paradigm for genetic algorithms. But, there are still a lot of open areas which need some new directions for further advancement. For example, most of the methods applied so far are maximum or extended likelihood, and minimum information complexity measure. A full Bayesian approach with efficient space-time complexity, or its finite approximation with model averaging, is yet to be seen. The EDA models have been primarily used for binary-coded or finite alphabet chromosomes. They may be extended for real-coded genetic algorithms also, and one obvious drift toward that end will to replace the multinomial and Dirichlet distributions, as they are not suitable for real-valued random variables. Modeling distributions of variable length solution strings is also an open frontier where variable model complexity measure and model averaging with some modern techniques like reversible jump Monte Carlo may provide a robust tool for further research.

REFERENCES

1. D E Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

2. M Pelikan, D E Goldberg & Ecantu-Paz, *Linkage Problem, Distribution Estimation, and Bayesian Networks*, IlliGAL

Technical Report No 98013, 1998.

3 H Muhlenbein, T Mahnig & A O Rodriguez, Schemata, Distributions and Graphical Models in Evolutionary Computation, *Journal of Heuristics*, vol 5, 1999.

4 M Pelikan, D E Goldberg & Ecantu-Paz, *BOA : The Bayesian Optimization Algorithm*, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-99, vol I, Morgan kaufmann, 1999.

5 R Etxeberria & P Larranaga, *Global Optimization Using Bayesian Networks*, Proceedings of the 2nd Symposium on Artificial Intelligence, CIMAF-99.

6 J Whittaker, *Graphical Models in Applied Multivariate Statistics*, John Wiley & Sons, 1990.

7 G F Cooper & E H Herskovits, *A Bayesian Method for the Induction of Probabilistic Networks from Data*, Machine Learning, vol 9, 1992.

8 D Heckerman, D Geiger & M Chickering, *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data*, Microsoft Research Technical Report No MSR-TR-94-09, 1994.

9 J M Bernardo & A F M Smith, *Bayesian Theory*, John Wiley & Sons, 1994.

10 D Geiger & D Heckerman, A Characterization of the Dirichlet Distribution with Application to Learning Bayesian Networks, *Proceedings of the 11th UAI conference*, 1995.

11 A P Dempster, N M Laird & D B Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, vol 39, 1997.

12 W Buntine, Operations for Learning with Graphical Models, *Journal of Artificial Intelligence Research*, vol 2, 1994

13 D Madigan & A E Raftery, Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window, *Journal of the American Statistical Association*, vol 89, 1994.

14. J Rissanen, Stochastic Complexity, *Journal of the Royal Statistical Society*, vol 49, no 3, 1987.

15. P Larranaga, R Etxeberria, J A Lozano, B Sierra, I Inza & J M Pena, *A Review of the Cooperation Between Evolutionary Computation and Probabilistic Graphical Models*, Proceedings of the 2nd Symposium on Artificial Intelligence, CIMAF-99.

16. S Baluja, Population Based Incremental Learning: A Method for Intergrating Genetic Search Based Function Optimization and Competitive Learning, *Technical Report No. CMU-CS-94-163*, Carnegie Mellon University, Pittsburg, 1994.

17. J A Hertz, A S Krogh & R G Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

18. H Muhlenbein & G Paass, *From Recombination of Genes to the Estimation of Distributions I Binary parameters*, Parallel Problem Solving from Nature, PPSN-IV, 1996.

19. H Linhart & W Zucchini, *Model Selection*, Johan Wiley & Sons, 1990.

20. J S De Bonet, C L Isbell & P Viola, *MIMIC: Finding Optima by Estimating Probability Densities*, Advances in Neural Information Processing Systems, Volume 9, MIT Press, 1997.

21. S Baluja & S Davies, *Using Optimal Dependency Trees for Combinatorial Optimization: Learning the Structure of the Search Space*, Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, 1997.

22. M Pelikan & H Muhlenbein, *The Bivariate Marginal Distribuition Algorithm*, Advances in Soft Computing Engineering Design and Manufacturing, Springer-Verlag, 1999.

23. G Harik, *Linkage learning via probabilistic modeling in the ECGA*, IlliGAL Technical Report No 99010, 1999.

24. D Heckerman, *Bayesian Networks for Knowledge Representation and Learning, Advances in Knowledge Discovery and Data Mining*, MIT Press, 1995.

25. D M Chickering, *Learning Bayesian Networks is NP-*complete, Proceedings of AI and Statistics, 1995.

26. M Henrion, *Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling*, Proceedings of the 2nd UAI conference, 1998.