A Genetic Algorithm for Learning Parameters in Bayesian Networks using Expectation Maximization

Priya Krishnan Sundararajan PRIYA.SUNDARARAJAN@SV.CMU.EDU
Ole J. Mengshoel OLE.MENGSHOEL@SV.CMU.EDU
Carnegie Mellon University
Silicon Valley

Abstract

Expectation maximization (EM) is a popular algorithm for parameter estimation in situations with incomplete data. The EM algorithm has, despite its popularity, the disadvantage of often converging to local but non-global optima. Several techniques have been proposed to address this problem, for example initializing EM from multiple random starting points and then selecting the run with the highest likelihood. Unfortunately, this method is computationally expensive. In this paper, our goal is to reduce computational cost while at the same time maximizing likelihood. We propose a Genetic Algorithm for Expectation Maximization (GAEM) for learning parameters in Bayesian networks. GAEM combines the global search property of a genetic algorithm with the local search property of EM. We prove GAEM's global convergence theoretically. Experimentally, we show that GAEM provides significant speed-ups since it tends to select more fit individuals, which converge faster, as parents for the next generation. Specifically, GAEM converges 1.5 to 7 times faster while producing better log-likelihood scores than the traditional EM algorithm.

Keywords: Bayesian networks, parameter learning, expectation maximization, genetic algorithms, age-layered evolutionary algorithms, probabilistic crowding.

1. Introduction

The expectation maximization (EM) algorithm (Dempster et al., 1977) is a prominent method for estimating parameters in probabilistic models when there is missing or latent information. EM alternates between an expectation step (E-step), which computes an expectation of the likelihood, and a maximization step (M-step), which updates model parameters by maximizing the E-step's expected likelihood. The parameters computed by the M-step are then used to begin another E-step and the process is repeated until convergence. The EM algorithm has been widely used in many areas of artificial intelligence, engineering, and statistics, from estimating transition and emission probabilities in hidden Markov models in speech recognition (Rabiner, 1989) to microarray gene expression clustering in computational biology (Do and Batzoglou, 2008).

The EM algorithm has three severe problems: local optima, computational cost, and slow convergence. First, and unlike for the complete data case, when data are incomplete, parameter estimation becomes a multi-modal problem and the EM algorithm can get stuck in *local optima*. Second, the *computational cost* of each iteration of the EM algorithm can be high, especially because of inference in the E-step. Third, the *slow convergence* of EM, in terms of the number of iterations undergone, can also make the algorithm computationally expensive and time consuming.

In this paper, we address the local optima and slow convergence problems mentioned above by adding stochasticity to EM. Stochasticity is introduced via a genetic algorithm (GA). GAs compute solutions to optimization problems in a black-box fashion, using techniques inspired by natural evolution: populations, inheritance, selection, mutation, and crossover (Holland, 1975).

We focus on the problem of using the EM algorithm for parameter estimation in discrete Bayesian networks (BNs). A BN is a probabilistic graphical model that represents a set of random variables and their conditional dependencies and independencies using a directed acyclic graph. Our contributions are the following. First, we present and analyze a Genetic Algorithm for EM (GAEM). The goal of GAEM is to reduce the number of iterations in a multiple random starting point EM setup by selecting more fit individuals as parents for the next generation. The fitness of an EM run is measured by its log-likelihood. Second, we classify BNs based on the hardness of their search landscapes. We experimentally study the role of different GAEM replacement techniques for BN search landscapes of varying hardness. Third, we develop a new replacement mechanism, GAEM-ALEM replacement, where the child EM run's fitness is compared to that of a parent EM run during the child's EM optimization. If the child EM run is less fit than its parent, after a certain number of iterations, then it is discarded and replaced by the parent. Fourth, we perform experiments to measure GAEM's performance in terms of log-likelihood, total number of iterations, and CPU time for varying sample sizes and replacement mechanisms. We find that GAEM's solution quality is generally higher than of traditional EM. In addition, the GAEM-ALEM replacement mechanism provides a GAEM speed-up of 1.5 to 7, in terms of the number of EM iterations, relative to traditional EM.

This paper is organized as follows. We discuss related work in Section 2. We present our Genetic Algorithm for Expectation Maximization (GAEM) in Section 3. In Section 4, we derive conditions on the parameters of GAEM which ensure its convergence to a global optimum. Section 5 experimentally demonstrates GAEM's ability to reduce the number of EM iterations and improve solution quality. We conclude with a summary of contributions and future work in Section 6.

2. Discussion of Related Work

The EM algorithm has three problems: local optima, computational cost of the E-step and the M-step, and slow convergence. Several variants of the EM algorithm have been proposed to address these problems, as discussed in Section 2.1 below before comparing with GAEM in Section 2.2.

2.1 Related Work

The most prevalent way to mitigate the *local optima* problem is the multiple restart strategy (Chickering, 2002), in which the EM algorithm is initialized from n random starting points. After these n EM runs have completed, the run that results in the highest log-likelihood is used as an estimate of the global optimum.

The GA was introduced by Holland (1975) based on Darwin's principle of natural selection in the evolution of species. Jank (2006) reviewed some of the challenges of stochastic EM implementation and proposed a GA method for solving the local maxima problem in EM. Pernkopf and Bouchaffra (2005) combine a GA and EM for learning Gaussian mixture models. Their algorithm escapes from local optima and identifies the number of Gaussian components more accurately than traditional EM. Zhao et al. (2012) propose a random swap EM algorithm for learning Gaussian mixture models, where a randomly selected component is swapped to a new location in the feature space. In summary, Jank (2006), Pernkopf and Bouchaffra (2005) and Zhao et al. (2012) all focus on learning Gaussian mixture models from multivariate data. They show that a GA-based EM algorithm outperforms traditional EM, as it achieves the same fitness score while identifying the correct number of Gaussian components more often.

The problem of the *computational cost* of the E-step and M-step has been addressed by many EM variants (Delyon et al., 1999) (Meng and Rubin, 1993). For clustering of large datasets, there are incremental EM and lazy EM speed-up methods (Thiesson et al., 2001). Both these methods are based on a partial E-step and provide encouraging results for Gaussian mixtures. In incremental EM, the EM algorithm cycles through the samples in blocks and updates the parameters incrementally. A method to determine the optimal block size is proposed. In lazy EM, at scheduled EM iterations, only samples that cause significant changes in the parameters are used by EM.

Several methods have been developed to address the problem of *slow convergence* of the EM algorithm. These methods use conjugate gradient, modified Newton Raphson techniques (Jamshidian and Jennrich, 1997), or age-layered methods (Saluja et al., 2012). Most of these methods require a modification to the original EM algorithm. This makes the overall method more complex, more difficult to analyze, and hence not popular in practice. Age-layered EM has been extended to the MapReduce framework (Reed and Mengshoel, 2012).

Other research, even though it does not study the EM algorithm directly, is also relevant to this work. Larrañaga et al. (2013) review the application of evolutionary algorithms for solving NP-hard problems in BN inference and learning. Larrañaga et al. (1996) developed a GA for learning BN structures. This GA searches for the best ordering of the BN variables, where the score of an order is the score of a single high-scoring BN structure (Cooper and Herskovits, 1992). Myers et al. (1999) proposed an evolutionary Markov chain Monte Carlo (EMCMC) method, which combines an evolutionary algorithm with a Markov chain Monte Carlo algorithm for learning BN structures from incomplete data.

2.2 Comparison of GAEM to Related Work

We now compare GAEM to related methods. Unlike Jank (2006) and Pernkopf and Bouchaffra (2005), GAEM uses converged EM runs as parents for the next generation. This allows GAEM to explore close-to-optimal regions of the search space and helps the algorithm escape local optima. GAEM does not modify the EM algorithm (Jamshidian and Jennrich (1997), Delyon et al. (1999), Thiesson et al. (2001), Meng and Rubin (1993)) or the training data (Elidan et al. (2002)). Instead, GAEM is as a wrapper around the EM algorithm and uses the full set of training data to learn the parameters. Jank (2006), Pernkopf and Bouchaffra (2005), and Zhao et al. (2012)) deal with learning Gaussian mixture models using EM. Larrañaga et al. (1996) and Myers et al. (1999) are focused on Bayesian network structure learning. GAEM is focused on parameter learning in discrete Bayesian networks using the EM algorithm.

3. Genetic Algorithm for Expectation Maximization (GAEM)

3.1 Notation and Definitions

Let $\boldsymbol{X} = \{X_1, X_2, ..., X_R\}$ denote the set of random variables in a Bayesian network (BN). Each random variable X_k is associated with a conditional probability table (CPT). An individual ρ^t consists of a vector of CPTs of random variables in a BN.

Definition 1 *The CPT estimate of an individual i at generation t is denoted by Θ_i^t. An individual is defined as vector consisting of CPTs: $\rho_i^t = (\Theta_1^t, \Theta_2^t, .., \Theta_R^t)$.*

A CPT is generated based on the constraint that the sum of probabilities for different states of the random variable should be equal to 1 for a parent instantiation. A CPT is given by: $\Theta_j^t = (\theta_1^t, \theta_2^t, ..)$ where $\theta_l^t \in [0,1]$ denotes a probability value for a particular state given a parent instantiation.

3.2 GAEM: The Algorithm

```
procedure GAEM($\boldsymbol{\rho}^0, n_g, n_p, p_c, p_m, \alpha, \omega, \epsilon$)
    $t \leftarrow 1$                                   ▷ Initialize population
    for $i \leftarrow 1, n_p$ do
        $\boldsymbol{\rho}^t \leftarrow \boldsymbol{\rho}^t \cup \rho_i^t$
    end for
    $\boldsymbol{\rho}^{t'} \leftarrow$ POPEM($\boldsymbol{\rho}^t$)  ▷ Compute learned individuals
    repeat
        $\boldsymbol{\rho}^{t'''} \leftarrow$ GA($\boldsymbol{\rho}^{t'}$)
        if $\alpha$ = GAEM-ALEM & $t \geq 2$ then
            $\boldsymbol{\rho}^{t+1'} \leftarrow$ GAEM-ALEM($\boldsymbol{\rho}^{t'''}$)
        else
            $\boldsymbol{\rho}^{t''''} \leftarrow$ POPEM($\boldsymbol{\rho}^{t'''}$)
        end if
        if $t \geq 2$ then
            $\boldsymbol{\rho}^{t+1'} \leftarrow$ REPLACE($\boldsymbol{\rho}^{t'}, \boldsymbol{\rho}^{t''''}, \alpha$)
        end if
        $\rho^* \leftarrow$ SELECTBEST($\boldsymbol{\rho}^{t+1'}, \rho^*$)
        $t \leftarrow t + 1$
    until $t > n_g$
    return $\rho^*$
end procedure
procedure POPEM($\boldsymbol{\rho}^t$)
    for $i \leftarrow 1, n_p$ do
        $\rho_i^{t'} \leftarrow$ EM($\rho_i^t, \omega, \epsilon$)      ▷ EM till convergence
        $\boldsymbol{\rho}^{t'} \leftarrow \boldsymbol{\rho}^{t'} \cup \rho_i^{t'}$
    end for
    return $\boldsymbol{\rho}^{t'}$
end procedure
procedure GA($\boldsymbol{\rho}^t$)
    for $\rho_i^{t'}, \rho_j^{t'} \leftarrow \boldsymbol{\rho}^{t'}$ do
        $(\rho_i^{t''}, \rho_j^{t''}) \leftarrow$ CROSSOVER($\rho_i^{t'}, \rho_j^{t'}, p_c$)
        $\rho_i^{t'''} \leftarrow$MUTATION($\rho_i^{t''}, p_m$)
        $\boldsymbol{\rho}^{t'''} \leftarrow \boldsymbol{\rho}^{t'''} \cup \rho_i^{t'''}$
        $\rho_j^{t'''} \leftarrow$ MUTATION($\rho_j^{t''}, p_m$)
        $\boldsymbol{\rho}^{t'''} \leftarrow \boldsymbol{\rho}^{t'''} \cup \rho_j^{t'''}$
    end for
    return $\boldsymbol{\rho}^{t'''}$
end procedure
procedure GAEM-ALEM($\boldsymbol{\rho}^{t'''}$)
    $\boldsymbol{\rho}^{t''''} \leftarrow$ STARTEM($\boldsymbol{\rho}^{t'''}$)
    while $|\boldsymbol{\rho}^{t''''}| \neq 0$ do
        for $\rho_i^{t''''} \leftarrow \boldsymbol{\rho}^{t''''}$ do
            if $\eta(\rho_i^{t''''}) > n$) then
                STOPEM($\rho_i^{t''''}$)
                if ( $f(\rho_i^{t''''}) < f(\rho_i^{t'})$ ) then
                    $\boldsymbol{\rho}^{t+1'} \leftarrow \boldsymbol{\rho}^{t+1'} \cup \rho_i^{t'}$
                    $\boldsymbol{\rho}^{t''''} \leftarrow \boldsymbol{\rho}^{t''''} - \rho_i^{t''''}$
                else
                    RESUMEEM($\rho_i^{t''''}$))
                end if
            else
                if ($\kappa(\rho_i^{t''''}, \omega, \epsilon)$) then
                    $\boldsymbol{\rho}^{t+1'} \leftarrow \boldsymbol{\rho}^{t+1'} \cup \rho_i^{t''''}$
                    $\boldsymbol{\rho}^{t''''} \leftarrow \boldsymbol{\rho}^{t''''} - \rho_i^{t''''}$
                end if
            end if
        end for
    end while
    return $\boldsymbol{\rho}^{t+1'}$
end procedure
```

Figure 1: The GAEM algorithm, with inputs: initial bag of EM runs ($\boldsymbol{\rho}^0$), number of generations (n_g), population size (n_p), crossover probability (p_c), mutation probability (p_m), iterations after which comparison is performed in the GAEM-ALEM method (n), the name of the replacement method (α), iteration threshold (ω), and LL tolerance (ϵ). We include pseudocode for GAEM-ALEM. Other replacement methods are discussed in Section 3.3. The procedure PopEM runs EM on a population until convergence; StartEM starts EM on a population; StopEM stops an EM run; and ResumeEM resumes an EM run. The procedure κ checks convergence of an EM individual. The output of GAEM is the EM run with the highest LL found.

We now discuss GAEM, see pseudocode in Figure 1. An individual ρ is created by initializing its random variables with randomly generated initial probabilities for the CPT, observing the sum constraint. Let n_p denote the population size, and $\boldsymbol{\rho}^t$ denote the bag of n_p individuals at the t-th generation: $\boldsymbol{\rho}^t = (\rho_1^t, \rho_2^t, .., \rho_{n_p}^t)$. Let n_g denote the number of generations. The number of iterations reached by the i-th EM run is given by $\eta(\rho_i^t)$. An EM run is converged when it has the reached the maximum number of iterations ω or when the relative difference in log-likelihood between successive iterations is less than ϵ. A learned individual $\rho_i^{t'}$ is produced when the EM algorithm (line 24 in Figure 1) is run on an individual ρ_i^t until convergence. For each GA generation, the learned individuals form a bag $\boldsymbol{\rho}^{t'} = (\rho_1^{t'}, \rho_2^{t'}, .., \rho_{n_p}^{t'})$. Crossover and mutation (lines 31, 32 and 34 in Figure 1) are applied to $\boldsymbol{\rho}^{t'}$ to create offspring $\boldsymbol{\rho}^{t'''}$.

Consider an example BN with $R = 5$ random variables. We now describe the crossover, mutation, and replacement mechanisms of GAEM for this BN.

Crossover The GA selects two learned individuals randomly from the learned individual set $\boldsymbol{\rho}^{t'}$ as parents. Let $\rho_a^{t'}$ and $\rho_b^{t'}$ be two individuals from the parent population: $\rho_a^{t'} = (\Theta_{a1}^{t'}, \Theta_{a2}^{t'}, \Theta_{a3}^{t'}, \Theta_{a4}^{t'}, \Theta_{a5}^{t'})$ and $\rho_b^{t'} = (\Theta_{b1}^{t'}, \Theta_{b2}^{t'}, \Theta_{b3}^{t'}, \Theta_{b4}^{t'}, \Theta_{b5}^{t'})$ We apply a single point crossover. Let $c \in \{1, 2, 3, 4\}$ be a random crossover point. Crossover happens with probability p_c. The resulting individuals ($\rho_i^{t''}$ and $\rho_j^{t''}$) are the children (line 31 in Figure 1). For instance, if $c = 2$, then the children of $\rho_a^{t'}$ and $\rho_b^{t'}$ are: $\rho_a^{t''} = (\Theta_{a1}^{t''}, \Theta_{a2}^{t''}, \Theta_{b3}^{t''}, \Theta_{b4}^{t''}, \Theta_{b5}^{t''})$ and $\rho_b^{t''} = (\Theta_{b1}^{t''}, \Theta_{b2}^{t''}, \Theta_{a3}^{t''}, \Theta_{a4}^{t''}, \Theta_{a5}^{t''})$

Mutation The purpose of mutation is to introduce diversity. Mutation helps to avoid premature convergence and gives a broader exploration of the search space. Mutation alters the values of one or more CPTs in an individual (lines 32 and 34 in Figure 1). For example, mutation in individual $\rho_b^{t''}$ happens with probability p_m at the CPT $\Theta_i^{t''}$ level. Mutation replaces the random variable's CPT $\Theta_i^{t''} = (\theta_1^{t''}, \theta_2^{t''}, ..)$ with randomly chosen values. Specifically, $\Theta_{a4}^{t'''}$ is mutated by choosing random values satisfying the probability constraint that the state probabilities sum to 1 for a given parent instantiation. The mutated individual $\rho_b^{t'''} = (\Theta_{b1}^{t''}, \Theta_{b2}^{t''}, \Theta_{a3}^{t''}, \Theta_{a4}^{t'''}, \Theta_{a5}^{t''})$. If mutation is applied on all random variables, then the resulting EM individual will be similar to a new random starting point and the EM algorithm will take more number of iterations to converge. Hence, it is useful to apply mutation on only a few random variables.

Replacement The replacement mechanism picks individuals from two sets of learned individuals ($\boldsymbol{\rho}^{t'}$ and $\boldsymbol{\rho}^{t''''}$) to form parents for the next generation. The comparison is done using the fitness f of learned individuals (line 45 in Figure 1). We define fitness as $f(\rho_i)$ = LL(ρ_i). The learned individuals of the first generation ($t = 1$) are $\boldsymbol{\rho}^{1'} = (\rho_1^{1'}, \rho_2^{1'}, .., \rho_{n_p}^{1'})$. The set of learned individuals in $\boldsymbol{\rho}^{1'}$ undergo crossover and mutation as explained above and generate offspring $\boldsymbol{\rho}^{1'''} = (\rho_1^{1'''}, \rho_2^{1'''}, .., \rho_{n_p}^{1'''})$. The EM algorithm is run on each individual in $\boldsymbol{\rho}^{1'''}$ to generate a set of learned individuals $\boldsymbol{\rho}^{1''''}$. Now, the two sets of learned individuals ($\boldsymbol{\rho}^{1'}$ and $\boldsymbol{\rho}^{1''''}$) will undergo replacement and the parents for the third generation are selected (lines 9 and 14 in Figure 1).

3.3 Replacement Mechanisms

We studied four different replacement mechanisms: traditional replacement, deterministic replacement, probabilistic replacement and ALEM-based replacement. We now explain each of them.

Traditional Replacement (GAEM-TRAD) Using this method, we compare the fitness of each parent $\rho_i^{t'}$ with its child $\rho_i^{t''''}$ and select the individual that has higher fitness (during crossover, a child is tied to the parent from which it gets the first part). Using the above example, we first compare the fitness of $\rho_1^{1'}$ with $\rho_1^{1''''}$, then compare the fitness of $\rho_2^{1'}$ with $\rho_2^{1''''}$, and so on. If the parent $\rho_i^{1'}$ is best, then it replaces its corresponding child $\rho_i^{1''''}$, otherwise we pick the child. This method illustrates a competition within the family, where one parent competes with only one child.

Deterministic Replacement (GAEM-DETER) This method is based on deterministic crowding (Mahfoud, 1992), where an offspring competes against its most similar parent. The distance $KL(\rho_a^{t'}, \rho_a^{t''''})$ is computed using KL divergence (Kullback and Leibler, 1951). Under this replacement scheme, each offspring competes for survival with its most similar parent. The winner is the individual with the highest fitness. This crowding scheme can be used to efficiently preserve diversity in the population.

Probabilistic Replacement (GAEM-PC) Probabilistic crowding (Mengshoel and Goldberg, 1999) uses a non-deterministic rule to establish the winner of a competition between parent and child. The probability that a parent ($\rho_a^{t'}$) replaces a child ($\rho_a^{t''''}$) is: $P_{\rho_a^{t'}} = \frac{f(\rho_a^{t'})}{f(\rho_a^{t'})+f(\rho_a^{t''''})}$. In this replacement mechanism, the fitter individuals do not always win over the weaker individuals, they win proportionally according to their fitness. This strategy helps to preserve diversity in the population.

ALEM based Replacement (GAEM-ALEM) This technique is inspired by the age-layered EM (ALEM) technique (Saluja et al., 2012), where EM runs compete within age brackets. Age is interpreted as the number of EM iterations. The assumption is that the probability of a "young" and poorly performing EM run (relative to its age bracket) turning into an "old" but strongly performing run (relative to its age bracket) is low. Such a "young" and poorly performing EM run can thus be discarded early to save CPU cycles for other, better performing EM runs. After an initial n iterations in GAEM, the fitness of a child EM run is compared with the fitness of its parent EM run at each iteration (lines 43 and 45 in Figure 1) . If the parent has a higher fitness than the child, then parent replaces child. If the child has a higher fitness, then the child is allowed to iterate but the comparison check takes place at each EM iteration of the child. The final set of individuals ($\boldsymbol{\rho}^{t+1'}$) form the parents for the next generation (line 59 in Figure 1).

4. Analysis

Using finite Markov chain theory, it has been shown that the canonical GA converges to the global optimum when the best solution is maintained in the population (Rudolph, 1994). We will prove the convergence of GAEM along similar lines as an earlier proof (Krishna and Murty, 1999), namely by deriving conditions for the parameters so that the convergence to global optimum is feasible. Some definitions from matrix theory are given below:

Definition 2 *A square matrix A is positive if its elements $a_{ij} > 0\ \forall\, i, j \in \{1, 2, ..., m\}$. A square matrix A is primitive if there exists a $k \in \mathbb{N}$ such that A^k is positive. A square matrix is column allowable if there exists at least one positive entry in each column. If the elements of a square matrix A, $a_{ij} \in [0, 1]$ and $\sum_{j=1}^{m} a_{ij} = 1$, then it is a stochastic matrix.*

In GAEM, the population of individuals at generation t is denoted by $\boldsymbol{\rho}^t$ (see Section 3.1). A CPT is generated based on the constraint that the sum of probabilities of the states of a random variable should be equal to 1 for a given parent instantiation. We define the state space S of this problem as the space of all possible random starting points satisfying the CPT constraint. The state space can be numbered from 1 to $|S|$. In GAEM, the process of generating a population can be denoted by a random variable Y, $Y(t) = \boldsymbol{\rho}^t{}_{t\geq 0}$ is dependent only on the previous population $Y(t-1)$ at generation $(t-1)$. Hence $Y(t) = \boldsymbol{\rho}^t{}_{t\geq 0}$ is a Markov chain and is denoted by:

$$\Pr(Y(t) = \boldsymbol{\rho}^t | Y(t-1) = \boldsymbol{\rho}^{t-1}, \ldots, Y(0) = \boldsymbol{\rho}^0) \ = \ Pr(Y(t) = \boldsymbol{\rho}^t | Y(t-1) = \boldsymbol{\rho}^{t-1}).$$

The transition probabilities p_{ij} are defined as the probability of generating population $\boldsymbol{\rho}^i$ from $\boldsymbol{\rho}^j$: $p_{ij}(t) \ = \ Pr(Y(t) = \boldsymbol{\rho}^i | Y(t-1) = \boldsymbol{\rho}^j)$. These transition probabilities are independent of the time instant, *i.e.*, $p_{ij}(s) = p_{ij}(t)$ for all $\boldsymbol{\rho}^i, \boldsymbol{\rho}^j \in S$ and for all $s, t \geq 1$. Therefore, $Y(t) = \boldsymbol{\rho}^t{}_{t\geq 0}$ is a time-homogeneous finite Markov chain. Let $P = \{p_{ij}\}$ be the transition matrix of the process $Y(t) = \boldsymbol{\rho}^t{}_{t\geq 0}$. The entries of the matrix P satisfy $p_{ij} \in [0, 1]$ and $\sum_{j=1}^{|S|} p_{ij} \ = \ 1\ \forall i \in S$. So

P is a stochastic matrix. For GAEM to converge to a global maximum, it is required that P be a primitive matrix. We investigate the operators that make the matrix P primitive. The probabilistic changes of the individuals during GAEM operation can be captured by the transition matrix P, which can be decomposed into a product of stochastic matrices, $P = C * M * R * L$, where C, M, R, and L describe the intermediate transitions caused by crossover, mutation, replacement, and local search (EM learning) respectively. Each element of the matrices C, M, R and L represent the transition probability of generating a new individual $\rho_i^{t+1'}$ from a parent individual $\rho_i^{t'}$ on the application of crossover, mutation, replacement, and local search (EM) mechanisms respectively. Since every positive matrix is primitive, it is therefore enough to find the conditions which make C and M positive, and R and L column-allowable.

Proposition: Let C, M, R and L be stochastic matrices, where C and M are positive[1] and R and L are column allowable. Then the product $P = C * M * R * L$ is positive.

Crossover: An individual $\rho_i^{t''}$ can be obtained from the individual $\rho_i^{t'}$ on application of the crossover operator. Each state of S is mapped probabilistically to another state, so C is stochastic.

Mutation: The matrix M is positive if any individual $\rho_i^{t'''}$ can be obtained from an individual $\rho_i^{t''}$ on application of the mutation operator. The mutation is done by taking into account the constraint on CPT as discussed in Section 3.

Replacement: The matrix R is column allowable, *i.e.*, there is at least one positive entry in each column of the matrix. The replacement process does not alter the CPTs of the individual. This matrix takes into account the probability of survival of each individual in the current population, which depends on the fitness (log likelihood) value of the individual. So there is only one positive entry, r_{ij} when $i = j$. The survival probability can be bounded as shown: $r_{ii} \geq \frac{f(\rho_i^t)}{\sum_{i=1}^{N} f(\rho_i^t)} \geq 0, \ \forall i \in S$. Even though this bound changes with the generations, it is always strictly positive. Hence, R is column allowable.

Local search: The matrix L is column allowable, *i.e.*, there is at least one positive entry in the matrix. The local search is done by applying the EM algorithm, which is a deterministic procedure. When an individual ρ_i^t undergoes EM learning, it will always converge to the same new individual $\rho_i^{t'}$. So there is only one positive entry per column, $l_{ij} = 1$.

Theorem: Let $X(t) = \max\{f(\rho_i^t) | i = 1, \ldots, n_p\}$ be a sequence of random variables representing the best fitness within a population $\boldsymbol{\rho}^t$ at generation t. Let the matrices C and M be positive and R and L be column allowable as described above. Then,

$$\lim_{t \to \infty} \Pr(\{X(t) = f^*\}) = 1,$$

where $f^* = \max\{f(\rho_i^t) \, | \rho_i^t \in S\}$ is the global optimum.

Proof Sketch: It is proved that a canonical genetic algorithm, which maintains the best solution found so far, converges to the global optimum (Rudolph, 1994). Under the hypothesis of the theorem, the transition matrix P of GAEM is primitive as discussed above. From the pseudocode in Figure 1, it can be seen that the best solution found in the generation is maintained.[2] Thus, the theorem follows from Rudolph (1994, Theorem 6). □

1. The crossover and mutation probabilities are assumed to be greater than zero ($p_c > 0$ and $p_m > 0$).
2. The best solution is different for different replacement mechanisms.

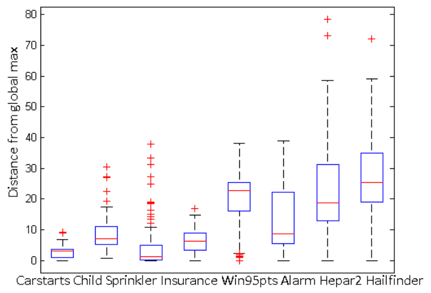

Figure 2: Box plots (or box and whisker diagrams) reflecting the spread of log-likelihoods for different BNs. For each BN, the plot is for $n_p = 100$ runs, at convergence, of the traditional EM algorithm.

5. Experimental Results

5.1 Methods and Data

We performed experiments on several BNs of different sizes and structures, see Figure 2. Most of the experiments were done on Carstarts with 18 variables, Alarm with 37 variables, Hepar2 with 70 variables, and Win95pts with 76 variables.[3] Through Gibbs sampling, we generated 500 samples with 19, 7, 35, and 38 hidden variables for the Alarm, Carstarts, Hepar2, and Win95pts BNs respectively. For Alarm and Carstarts, we chose hidden variables such that traditional EM was known to need many iterations to converge (Saluja et al., 2012). For Win95pts and Hepar2, we randomly hid 50% of the variables. The hidden variables are fixed for a BN, and do not vary between experiments.

We implemented GAEM by using the libDAI graphical models C++ library (Mooij, 2010), which contains an EM implementation oriented towards parameter estimation in Bayesian networks, and the Boost multithreading library to process the EM runs in parallel, on a multi-core computer. In experiments, we set the maximum number of iterations that the EM algorithm can undergo to $\omega = 1000$ iterations, and the log-likelihood difference tolerance (*i.e.*, if the difference in log-likelihood between two iterations is less than this tolerance, then we say the EM run has converged) to $\epsilon = 0.00001$. We generated EM runs by randomly choosing starting points for each BN.

5.2 Search Space Analysis

To better understand the structure of the BN parameter search spaces, we first study the spread of LL values for the traditional EM algorithm at convergence. In this experiment, we used eight BNs: Carstarts, Child, Sprinkler, Insurance, Win95pts, Alarm, Hepar2, and Hailfinder. For each BN, we executed $n_p = 100$ traditional EM runs and used a sample size of 500. Distances are calculated using d_i = LL* - LL$_i$, where LL* is the maximum LL (max_LL) seen among the 100 EM runs and LL$_i$ is the LL of the i-th EM run. Figure 2 shows the spread of distances for the experimental BNs.

The red line in Figure 2 denotes the median; everything above or below it represents the distance of 50% of the EM runs. The size of the box denotes the spread of the LL distances around the median. For Carstarts, Child, Sprinkler, and Insurance, the median is close to the max_LL. In Win95pts, the spread above the median is low, implying that 50% of EM runs are closer to the max_LL. Thus, we consider Carstarts, Child, Sprinkler, Insurance, and Win95pts as easy search spaces. For Alarm, Hepar2, and Hailfinder, the spread above the median is high and 50% of the EM runs are away from the max_LL. Thus, we consider these BNs as hard search spaces.

3. http://www.bnlearn.com/bnrepository/

5.3 Role of Replacement

We now study the impact on GAEM of four different replacement mechanisms: traditional (GAEM-TRAD), deterministic crowding (GAEM-DETER), probabilistic crowding (GAEM-PC), and ALEM-based (GAEM-ALEM). We investigate Alarm and Carstarts, representing the hard and easy search spaces respectively. We carefully optimized GAEM input parameters (such as n_g, n_p, p_m, and p_c) in pilot experiments.[4] For Alarm, we use $n_g = 100$, $n_p = 4$, $p_m = 0.1$, and $p_c = 0.1$. For Carstarts, we use $n_g = 200$, $n_p = 2$, $p_m = 0.05$, and $p_c = 0.5$. The results, shown in Figure 3, are taken as an average over 10 independent GAEM runs.

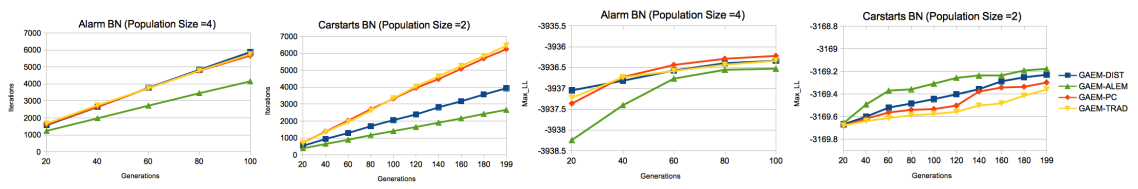

Figure 3: Role of replacement techniques on Alarm and Carstarts BN. The two leftmost panels show n_t as a function of n_g. The two rightmost panels show max_LL as a function of n_g

From the two leftmost panels of Figure 3, we see that GAEM-ALEM uses few iterations for both Alarm and Carstarts. From the two rightmost panels of Figure 3, we see that GAEM-PC produces a better solution quality for Alarm, while GAEM-ALEM produces a better solution quality for Carstarts. From this experiment, we conclude that GAEM-ALEM is faster than other techniques. Though for Alarm the solution quality of GAEM-ALEM is worse than that of other replacement techniques, GAEM-ALEM still beats the traditional EM algorithm (see Table 1).

5.4 Processor Time Comparison

The goal of this experiment is three-fold. First, we want to measure the solution quality (max_LL) obtained by GAEM versus EM. Second, we want to compare the processor time (CPU time) taken by GAEM and EM. Third, we want to understand the speed-up in terms of the total number of iterations taken by GAEM (denoted as n_{GAEM}) versus traditional EM (denoted as n_{EM}). We consider Alarm and Carstarts representing the hard and easy search spaces, respectively. For Alarm, we let $p_c = 0.1$, $p_m = 0.1$, $n_p = 4$, and $n_g = 50$, resulting in a total of $4 \times 50 = 200$ EM runs. For Carstarts, we let $p_c = 0.5$, $p_m = 0.05$, $n_p = 2$, and $n_g = 100$, resulting in a total of $2 \times 100 = 200$ EM runs. For apples-to-apples comparison, we transfer the concept of generation from GAEM to traditional EM. In GAEM, we have n_p EM runs in the first generation and use the learned individuals as parents in subsequent generation, also with n_p EM runs. Similarly, in traditional EM, we start with n_p EM runs in the first generation and use a new set of n_p EM runs in subsequent generations.[5] We tried traditional replacement (GAEM-TRAD), PC based replacement (GAEM-PC) and ALEM based replacement (GAEM-ALEM) and varied the sample sizes from 500 to 3000 for both BNs.[6]

For Alarm, Table 1 shows that the solution quality for GAEM-TRAD is higher for most of the sample sizes compared to GAEM-PC and GAEM-ALEM. But all the three replacement methods

4. The results of these pilot experiments are not reported here due to limited space.
5. We keep the total number of EM runs as 200 in both BNs because we assume that a new user who wants to compare GAEM with traditional EM would allot time based on the time taken by traditional EM per EM run.
6. The results for GAEM-DETER is found to be similar to GAEM-PC in terms of the number of iterations and solution quality hence it is not shown in the Tables 1 and 2.

	Carstarts				Alarm			
Samples	**GAEM-TRAD**	**GAEM-PC**	**GAEM-ALEM**	**EM**	**GAEM-TRAD**	**GAEM-PC**	**GAEM-ALEM**	**EM**
500	-3169.40	-3169.55	**-3169.31**	-3169.96	-3936.93	**-3936.69**	-3937.31	-3937.44
1000	-6924.56	-6924.56	**-6924.54**	-6924.80	**-16047.15**	-16048.18	-16048.6	-16050.20
1500	**-8646.85**	**-8646.85**	**-8646.85**	**-8646.85**	**-22977.85**	-22977.89	-22979.56	-22981.7
2000	-13743.87	-13744.50	**-13743.84**	-13743.85	-31217.09	**-31217.02**	-31217.90	-31218.10
2500	-14504.44	-14504.46	-14504.43	**-14504.40**	**-40550.40**	-40550.97	-40552.73	-40556.00
3000	-18888.15	**-18887.61**	-18887.84	-18887.90	**-51210.45**	-51211.13	-51217.46	-51220.50

Table 1: Comparing max_LL for the GAEM-TRAD, GAEM-PC, and GAEM-ALEM methods with the traditional EM method for Carstarts and Alarm. The highest max_LL are in bold. The total number of iterations taken to reach the max_LL is shown in Table 2.

	Carstarts			Alarm		
Samples	**GAEM-TRAD**	**GAEM-PC**	**GAEM-ALEM**	**GAEM-TRAD**	**GAEM-PC**	**GAEM-ALEM**
500	2049 (4.1)	2757 (3.1)	1397 (**6.0**)	3062 (4.6)	3172 (4.5)	2322 (**6.1**)
1000	1227 (4.6)	1765 (3.2)	1016 (**5.6**)	1659 (3.5)	1631 (3.6)	1386 (**4.2**)
1500	624 (**1.5**)	624 (**1.5**)	624 (**1.5**)	2515 (4.2)	2522 (4.2)	1940 (**5.5**)
2000	1282 (4.4)	1231 (4.6)	989 (**5.8**)	1097 (3.6)	1107 (3.6)	997 (**4.0**)
2500	688 (2.0)	683 (**2.1**)	684 (**2.1**)	2507 (3.8)	2499 (3.8)	1774 (**5.4**)
3000	1214 (5.2)	1208 (5.2)	977 (**6.5**)	4082 (4.0)	4002 (4.1)	2353 (**7.0**)

Table 2: Comparing total iterations for GAEM-TRAD, GAEM-PC and GAEM-ALEM methods with traditional EM method. Speedups are shown in parentheses, with the highest speedups in bold. These results corroborate the CPU-time experiments in Figure 4.

have a higher solution quality than EM. For Carstarts, all three replacement techniques achieve better solution quality than traditional EM (see Table 1) in all cases except for sample size = 2500, where the difference in log-likelihood is very small (0.0002%).

We now consider the number of iterations until convergence, as well as the speed-up, for the replacement methods. The speed-up is calculated as $s_X = \frac{n_{EM}}{n_X}$ where X reflects the replacement method. In Table 2, for Alarm, we can see a speed-up of $3.5 \leq s_X \leq 4.6$ (as shown in the parentheses) for all sample sizes using GAEM-TRAD and GAEM-PC techniques and a much higher speed-up $4.0 \leq s_{ALEM} \leq 7.0$ for GAEM-ALEM. For Carstarts, GAEM-TRAD and GAEM-PC show a speed-up $1.5 \leq s_X \leq 5.2$ for all sample sizes (see Table 2). But GAEM-ALEM has a higher speed-up $1.5 \leq s_{ALEM} \leq 6.5$. These results suggest that GAEM can achieve a better solution quality in a shorter time, measured by the number of EM iterations, compared to the traditional EM for given set of random starting point configurations.

Figure 4 shows the results of comparing processor time for Carstarts and Alarm. All GAEM-based methods take less time than traditional EM for both BNs across all sample sizes. For both Alarm and Carstarts, we can see that the processor time consumption for GAEM-PC and GAEM-

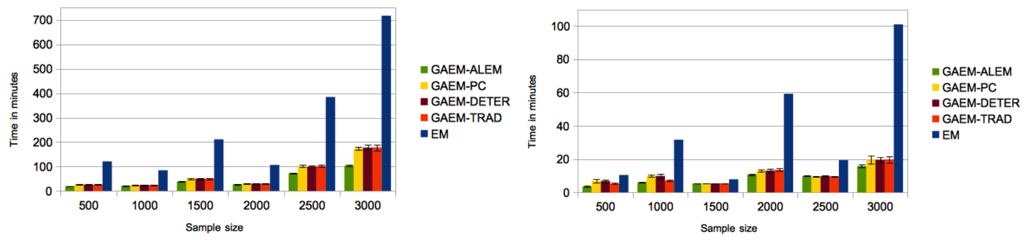

Figure 4: Processor time comparison for EM and GAEM on Alarm (left) and Carstarts (right) for 200 EM runs. The corresponding max_LL for these configurations are shown in Table 1.

DETER is almost the same. Among the methods, GAEM-ALEM is the fastest. For Carstarts, GAEM-ALEM is the most suitable since there is an improvement both in solution quality and compute time. For Alarm, GAEM-ALEM shows a decrease in the total number of iterations which in turn shortens CPU time. The solution quality, though poorer than for GAEM-TRAD and GAEM-PC, is better than for traditional EM. We conclude that GAEM-ALEM obtains better solution quality (LL) in less time, relative to traditional EM, for both BNs.

6. Conclusion

In this work, we develop a Genetic Algorithm for EM (GAEM) for parameter learning from incomplete data in Bayesian networks. From experiments, we find that GAEM achieves better log-likelihoods in most cases compared to the traditional EM algorithm. A key component of GAEM is a novel replacement technique, GAEM-ALEM. Using GAEM-ALEM, poorly performing EM runs, in terms of their log-likelihoods, are discarded early during the progress of EM optimization. In experiments, GAEM-ALEM produced a speed-up of 1.5 to 7, along with better solution quality, compared to the traditional EM algorithm. These results suggest a general capability of GAEM to produce better solutions in shorter time relative to traditional EM. We also prove the global convergence of GAEM theoretically. For future work, we will explore other evolutionary methods and compare their performance to that of GAEM. We will also investigate other ways of characterizing the structure of the BN parameter search spaces.

References

D. M. Chickering. Learning equivalence classes of Bayesian Network structures. *Journal of Machine Learning Research*, 2, 2002.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, 27(1):94–128, 1999.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.

C. B. Do and S. Batzoglou. What is the expectation maximization algorithm? *Nature Biotech.*, 26: 897–899, 2008.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *Eighteenth National Conference on Artificial Intelligence*, pages 132–139, 2002.

J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

M. Jamshidian and R. I. Jennrich. Acceleration of the EM algorithm by using quasi-Newton methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 59(3):pp. 569–587, 1997.

W. Jank. The EM algorithm, its stochastic implementation and global optimization: Some challenges and opportunities for OR. *Perspectives in Operations Research*, pages 367–392, 2006.

K. Krishna and M. N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, Jun 1999.

S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 26(4):487–493, Jul 1996.

P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana. A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Inf. Sci.*, 233:109–125, June 2013.

S. W. Mahfoud. Crowding and preselection revisited. In *PPSN 2*, pages 27–36, Amsterdam, 1992. North-Holland.

X. L. Meng and D. B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):pp. 267–278, 1993.

O. J. Mengshoel and D. E. Goldberg. Probabilistic crowding: Deterministic crowding with probabilisitic replacement. In *GECCO-1999*, volume I, pages 409–416, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

J. M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, Aug. 2010.

J. W. Myers, K. B. Laskey, and K. A. DeJong. Learning Bayesian networks from incomplete data using evolutionary algorithms. GECCO'99, pages 458–465, 1999.

F. Pernkopf and D. Bouchaffra. Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1344–1348, Aug. 2005. ISSN 0162-8828.

L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

E. B. Reed and O. J. Mengshoel. Scaling Bayesian network parameter learning with expectation maximization using MapReduce. *Proc. of Big Learning: Algorithms, Systems and Tools*, 2012.

G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Trans. on Neural Networks*, 1994.

A. Saluja, P. Sundararajan, and O. J. Mengshoel. Age-layered expectation maximization for parameter learning in Bayesian networks. *AISTATS-2012*, pages 984–992, 2012.

B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 45:279–299, 2001.

Q. Zhao, V. Hautamäki, I. Kärkkäinen, and P. Fränti. Random swap EM algorithm for Gaussian mixture models. *Pattern Recognition Letters*, (16):2120–2126, 2012.